Delft University of Technology

Multiscale Pattern Recognition of Transport Network Dynamics and its Applications
A bird's eye view on transport

Krishnakumari, Panchamy

DOI
10.4233/uuid:81f93c75-0b8a-413e-85a0-ca616fd533b2

Publication date
2020

Document Version
Final published version

Citation (APA)
Krishnakumari, P. (2020). *Multiscale Pattern Recognition of Transport Network Dynamics and its Applications: A bird's eye view on transport*. [Dissertation (TU Delft), Delft University of Technology]. TRAIL Research School. https://doi.org/10.4233/uuid:81f93c75-0b8a-413e-85a0-ca616fd533b2

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Multiscale Pattern Recognition of Transport Network Dynamics and its Applications

A bird's eye view on transport

Panchamy Krishnan Krishnakumari

This doctoral project received financial support from the SETA project funded by the European Union’s Horizon 2020 Research and Innovation program.

SETA

Cover photo: ESA/NASA

Multiscale Pattern Recognition of Transport Network Dynamics and its Applications

A bird's eye view on transport

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof.dr.ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 27 February 2020 at 15:00 o'clock
by

Panchamy Krishnan KRISHNAKUMARI
Double Master of Science in Information and Communication Technology
KTH Royal Institute of Technology, Sweden
Delft University of Technology, the Netherlands
born in Kollam, India.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	Chairman
Prof. dr. ir. J.W.C. van Lint	Delft University of Technology, promotor
Dr. O. Cats	Delft University of Technology, promotor

Independent members:

Prof. dr. M. Bell	The University of Sydney
Prof. dr. F.C. Pereira	Technical University of Denmark
Prof. dr. F. Viti	University of Luxembourg
Prof. dr. ir. P.F.A. Van Mieghem	Delft University of Technology
Prof. dr. ir. S.P. Hoogendoorn	Delft University of Technology
Prof. dr. ir. A. Verbraeck	Delft University of Technology, reserve member

TRAIL Thesis Series T2020/5, the Netherlands TRAIL Research School

TRAIL
P.O. Box 5017
2600 GA Delft
The Netherlands
E-mail: info@rsTRAIL.nl

ISBN: 978-90-5584-263-6

Copyright © 2020 by Panchamy Krishnan Krishnakumari

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without written permission of the author.

Printed in the Netherlands

For all girls who were told they couldn't.

Acknowledgements

It is fascinating how a small-town girl with no big aspirations, one who never left her home for 22 years, ended up obtaining a master's degree from Europe and is now at the final stages of obtaining her Ph.D. There's an old proverb in Africa that says it takes a whole village to raise a child. Here, I would like to take the opportunity to thank my village.

I met Hans almost 5 years ago as a student assistant and there was no prelude that it would lead to this. With a specialization in medical imaging, I had no reason to pursue a substantial career in the transport field, especially an academic career. However, he saw potential in me and my work even when I didn't and, for that, I am extremely grateful. Oded was brought in as my daily supervisor a bit later in my Ph.D. and it made me realize how lucky I am for his supervision as I know how it would have been without his guidance. With Hans and Oded, I had a perfect supervisory team where we complemented each other. Hans brought along enthusiasm, vision and his brilliant mind. Oded brought structure into my chaotic work, an abundance of knowledge and his invaluable guidance. I want to thank both of you for all the professional and emotional support you have provided me with through these years and for reigning me in when needed so that I was not drowning in work.

My Ph.D. started with an ambitious collaborative work with Ludovic and Clelia and this helped me navigate the transportation field as I was just a novice in this field. This work remains one of the cornerstones of my thesis. The collaboration with Hai and his students, Nam and Tin, showed me how I could use my background in computer science in this domain. Special thanks to all my other collaborators - Rafael, Tamara, Nam, Tin, Alan - and the students I have worked with - Theo, Faye, Nicolas. These seemingly random collaborations have truly enriched my knowledge and aided me in slowly converging my thesis into what it is today.

I would like to thank my doctoral committee members for investing their valuable time in reviewing my thesis - Francesco Viti, Francisco Pereira, Mike, Piet, Serge, and Alexander. I have admired your works and have had the opportunity to meet all of you during my Ph.D. It is a great honor that you are part of my journey.

There is nothing that promotes a good work environment than the colleagues you enjoy working with. It has been my absolute pleasure to work with an amazing team at our Dittlab - Ding, Tin, Huong, Ehab, Shahad, Yezen (half-human), Zahra, Parviz, Leonie, Justin, Lodewijk, Sanmay, Kristel, Guopeng, Simoen, Kai and Peter. The many outings, dinners, birthdays and celebrations made us bond beyond the walls of 4.02 room and I am sure we will continue to be in each other's lives even in the future. Special thanks to Tin and Ehab for being the jokers of Dittlab and making it a fun place to work at. I have to thank Sanmay for all our discussions on music, food, and politics. They were always a welcome distraction for me. I am also grateful to Leonie for our coffee breaks (Tijd voor koffie). You are such a caring and warm person, and it was fun to work and watch Lodewijk grow up with you. I also want to convey my thanks to our visitors - Clelia, Etienne, He, Loic, Nicolas, Nam, Rafael, Alan, Kota, Juan, and Julian - who always made for a dynamic and interesting working environment. Above all, there is one person that was always next to me, figuratively and literally, for the past 4 years at Dittlab - Ding. You were there for me when I needed to share my grievances or needed food. I am particularly glad that we get to finish our Ph.D. journeys together.

I have a great appreciation for the amazing and diverse group of people in our department. Niels, for all the talks we had at the department and during TRB. Paul, always great conversations and an amazing conference travel buddy. Yan, you are a bundle of joy whenever I see you. Danique, for all the great floor talks we had. Jishnu and Freddy, for being there when I missed talking in Malayalam. Priscilla, Moreen and Dehlaila, for all the trouble you went through for my never-ending contract problems. Conchita, for helping me during the final stages of my thesis preparations. Special thanks to Guilia, Maria, Marie-Jette, Joelle, Irene, Martijn, Tim, Niharika, Malavika, Nikola, Nejc, Konstanze, Alexandra, Pablo, Alphonse, Menno, Xavi, Arjan, Vincent, Yufei, Solmaz, Rafal for all the conversations and fun. I also would like to acknowledge the people at the graduate school for all the help with finalizing my thesis.

It seems like four years went by quite fast. However, I then remember the people that I had lost and realized that time had not been kind. I want to take this opportunity to remember the people who couldn't be here to share in my happiness - Gomathi ammumma, KC appoppan, Anantham ammumma, Ravi chittapan, Giya ammumma, Sjacky, Muthassi, and Boris. You will always be in my memory.

I wouldn't have been able to survive these 7 years, away from India, if it weren't for my strong support system. Thanks to my friends - Li, Sevil, Saira, Elena, Elisabeth, Kevin, and JB - for cheering me on and supporting me from all corners of the world. Special thanks to Biju mamman for his support throughout my entire life. This wouldn't have been possible without you. Arjan and Cynthia for welcoming me into the family and making the Netherlands feel like home. Parvathy, Balu, and Aadi for bringing a bit of home to the Netherlands. Special thanks to my sisters, Parvathy and Pournami, for all the love and support. Instead of calling me a 10 pointer all the time, they will now get a new salutation to pull my leg. And to Jerry -for bringing unexpected love into my life. Coming home to you at the end of the day made my Ph.D. experience a lot sweeter.

Finally, all the gratitude in the world is not sufficient for my biggest cheerleaders - my parents, Krishnakumar and Krishnakumari. I want to thank them for not being disappointed in me and my sisters for being girls and for giving us every opportunity a man would have had. Thank you for instilling in us the importance of education. My father, for not listening to people, sometimes even us, when it came to our education and pushing us to be better than what society wanted us to be. My mother, for not trying to change her three unruly and disobedient girls to fit the mold. They allowed us to be different and for that, I am thankful. This thesis is as much their achievement as it is mine!

Panchamy Krishnakumari,
Delft, January 2020.

Contents

Chapter 1

Introduction

The three major traffic-related social costs are traffic accidents, congestion, and environmental damages, which include both hard economic costs and intangible human costs. Based on the methods used to estimate these costs, they can vary between different countries. Even a developed country like the Netherlands scores relatively poorly with respect to these costs; the cost of road crashes, in particular, was about 2% of the GDP ($\approx$ 14 billion euros) in 2015 [1]. We need different tools to either mitigate or even prevent these costs such as traffic and demand management, incident management, transport planning, and freight scheduling.

These tools require clear insights into network dynamics, both demand, and supply. This starts with monitoring and then analyzing this information for prediction, optimization and long term planning for redesigning policies, services or infrastructure. Such analysis techniques may be automated and made available through decision support systems for traffic managers or service operators. In this thesis, we focus on operational decision support and the information and insights needed for such tools ranging from network representations, traffic and demand data to understanding network dynamics.

There are several key ingredients needed for an efficient transportation decision support system, as illustrated in figure 1.1. The first key ingredient and cornerstone of a transportation model is a proper graph representation of the underlying road or public transport infrastructure network. Then, we need the information that is transferred through the network. For a transport system, this information can relate to the number of vehicles being transported (flow) or the speed at which they are traveling. This information can be obtained through different data sources such as loop detectors, vehicle movement traces, surveys or travel diaries.

There are different methods for estimating such information from the available data and network representation through data assimilation. This information includes demographics and land use, origin-destination matrices, link flows, speeds and density. There is a multitude of purposes for such information, ranging from transport planning and design, policy evaluation, and monitoring, to uses beyond the mobility domain itself, e.g. asset management, city planning, etc. These variables can also be used to completely and uniquely describe the dynamic evolution of the transportation system in order to understand the dynamics of the network. There are many applications for understanding why the traffic behaves in the way it does. One of the most extensively studied applications is using the

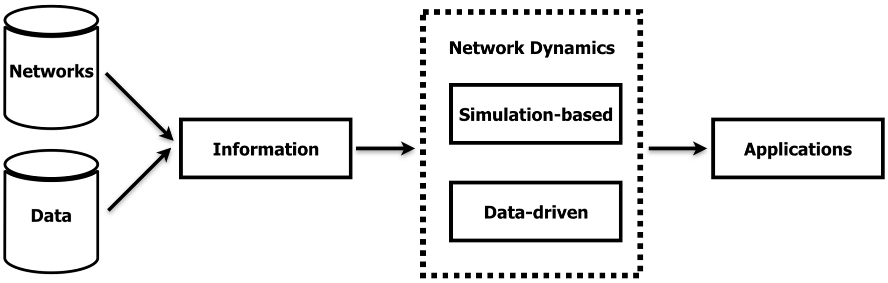


Figure 1.1: Simplified schematic representation of a transportation decision support system

traffic dynamics insights for traffic prediction, which is useful for re-routing traffic through less congested paths, for regulating traffic lights' cycle in order to better accommodate the traffic flow or for building context-aware navigation apps. Note that an ideal decision support system is not linear, as illustrated in figure 1.1. There are various feedback loops from dynamics to information and also from applications back into the information process [2].

1.1 Simulation-based vs. data-driven approaches to analyze network traffic dynamics

Understanding traffic dynamics has been one of the main research topics in the transportation sciences since the early 1950s. A large number of different methods have been developed by researchers for this purpose from all over the world as advanced techniques emerge and better data sources become available. There are many reviews of these methods, with different ways to categorize them [3–6]. [3] reviewed short-term traffic forecasting studies up to 2003 based on the determination of scope, modeling and conceptual output specification whereas [4] categorized the methods as naive, data-driven and model-based approaches. In this thesis, we broadly classify the methods into simulation-based and data-driven approaches.

Simulation-based approaches rely on mathematical modeling to mimic the complex dynamics of traffic systems. There are numerous studies on modeling traffic dynamics and propagation in single-dimensional traffic systems (corridor-level) from the field of transportation, physics and mathematics. An overview of these modeling approaches can be found in [7]. Some of the main approaches can be categorized as car-following models [8, 9], gas-kinetic models [10, 11], cellular automata [12], first-order traffic flow models [13] and higher-order traffic flow models [14, 15].

However, literature on network-level traffic dynamics is limited especially in the context of large-scale urban networks. Most of the previous studies have focused on microsimulation of link-level traffic dynamics or visually analyzing the congestion propagation. The idea of a macroscopic fundamental diagram (MFD) [16, 17], along with the empirical evidence of its existence [18–20], provided a breakthrough in modeling network dynamics. It was found that details at the individual link level are not needed to describe the congestion dynamics of cities but can be instead defined based on homogeneous regions of a city. The main characteristic of traffic is that congestion propagates both in space and time with some finite speed and is spatially correlated to the adjacent roads. However, such homogeneous regions are defined only in two dimensions, speed propagating in space of a single time period. The time variability in traffic dynamics is addressed sequentially by iterating the algorithms for each time step without directly incorporating time into the two-dimensional network.

On the one hand, from the literature, we know that approaches based on traffic and transport simulation models provide the ideal solution for decision support, as we can do what-if analysis with such models. However, there is a downside to simulation-based approaches for network dynamics. A full-fledged simulation-based solution for city networks is a highly complex and labor-intensive solution. This is because simulation-based approaches require many inputs that need to be derived from data – consistent graphs that represent the road infrastructure, boundary conditions and initial state estimates, inputs such as OD flows, route choice patterns, public transport schedules, and finally, parameters of the mathematical traffic flow models (microscopic/macroscopic/mesoscopic) for driving and traveling behaviour. Thus, a simulation-based approach is only as good as the quality of each of these inputs.

On the other hand, data-driven approaches predict traffic conditions by estimating the current or future traffic state from historical patterns, without detailed descriptions of intrinsic network dynamics based on traffic flow mechanisms. Thus, the prediction is not determined by consistent propagation over a network graph but through statistical modeling with generic mathematical models. Hence it is much less labor-intensive, less meta-data hungry and more robust to missing and faulty input data. The downside here, evidently, is that data-driven models are only as good as the degree to which the training data is representative of the traffic dynamics in a network. However, in the age of big data, the question is not really about data availability but rather how to effectively utilize it.

Literature in data-driven approaches is mainly based on different types of machine learning methods, an overview of which can be found in [6, 21, 22]. Based on the properties of the input data and the prediction parameters, the predictions can be broadly classified as linear and non-linear. For both linear and non-linear models, the prediction problem boils down to finding the optimal parameter set and defining the criteria for finding the optimal set of weights. The methods to assign or tune these parameters can be broadly classified as supervised, unsupervised and reinforcement learning methods. Some of the short-term prediction data-driven methods are linear regression [23–25]; ARIMA family of models [26, 27]; Bayesian methods [28–31]; dimensionality reduction methods [32, 33]; decision trees [34, 35]; k-nearest neighbors [36, 37] and neural networks [38–43].

Most of these methods consider traffic as single-dimensional data (time-series) where they build a data model for each link of the network, which restricts scalability. Recent graph-based neural network methods try to incorporate space to overcome this [44, 45]. However, the use of such black-box and non-linear methods do not provide insight into the network dynamics, either due to their inherent limited explanatory power or their inability to produce a unique solution to a problem. These disadvantages also hold for studies in the public transport domain where most of the data analysis is done at line-level and not for the whole network. With the increasing availability of smart card data, there are many promising research avenues for understanding network dynamics of public transit networks such as delay prediction, disruption detection and occupancy prediction [46]. To summarize, the failure to incorporate space and time jointly in representing network dynamics, limitations of the black-box methods and limitations on the scalability of the current approaches for large-scale urban networks calls for new data-driven approaches.

1.2 Research objective

Given the drawbacks of the current approaches in understanding the network dynamics, we can now formulate the research objective of this thesis as follows:

To design efficient data-driven methods for describing and understanding the traffic dynamics in large-scale metropolitan networks.

In order to refine our research objective, we identify several requirements that need to be fulfilled by any approach intended to improve the current methods:

- The approach should be *data-oriented* with a minimal number of parameters.
- The approach should incorporate dynamics over both *space* and *time*.
- The approach should provide a significant *computational* gain with respect to current approaches.
- The network traffic states derived from the methods should be *interpretable*.
- The approach should be *scalable* for networks at multiple levels of scale and for different modes.
- The network traffic states derived from the data-driven methods should be able to *adapt* or *evolve* as additional data emerge.

1.3 Research questions

To date, no such approach has been feasible because of the identified open issues of network traffic dynamics and a lack of knowledge on how to fulfill the requirements of the research objective. Therefore, we have focused on the following key questions:

1. One of the main challenges of any network-level study is the sheer number of dimensions involved in representing the traffic dynamics of a city. Thus, complexity reduction needs to be achieved wherever possible; it starts with the transportation network, which leads to one of our main key questions: *How can we reduce the complexity of the transportation network without compromising its key topological characteristics?* **[Chapter 2]**
2. The most important drawback of network-level analysis is the failure to integrate space and time dimensions while looking at the traffic dynamics. Most of the methods investigate either spatial correlations or temporal correlations but do not incorporate both of these dimensions simultaneously, which leads to the next question: *How can we incorporate spatio-temporal relations in representing the network traffic states?* **[Chapter 3]**

3. Data-driven methods for network traffic propagation are generally based on time-series data. We need to identify spatio-temporal features that can be used to represent a traffic state, which can then be used for understanding the traffic dynamics. *How can we define traffic states based on high-level physical attributes derived from data?* **[Chapter 4]**

4. In the network-level dynamics literature, MFD is the most common phenomenon that has been used to define the traffic dynamics of a city. Given that high-level spatio-temporal features can be used to represent traffic states, can we identify more of such features with inspiration from human vision? *How can we use concepts based on human vision to expand the pool of physical attributes to define a network traffic state?* **[Chapter 5]**

5. The most researched application of network-level dynamics is traffic predictions. Our assumption is that zooming out to high-level features can be used for applications other than for prediction, such as revealing linear/non-linear relationships between different spaces in traffic such as demand and supply. This leads to a key question: *How can we reveal correlations between spatiotemporal demand and supply patterns of a network using data?* **[Chapter 6]**

6. Scalability is one of the main issues for network-wide studies. The computational complexity of both spatiotemporal and time-series analysis increases significantly with the increase in network complexity. The ideal solution would be using scale-invariant high-level features to define the network traffic states. *How can we extend data-driven methods for multiscale networks?* **[Chapter 7]**

7. For these methods to be extended to other transport modes such as public transport, active mode, it should be possible to estimate the respective 3D traffic state. This is especially challenging for public transportation systems since they include both infrastructure and service networks. This leads to a key question: *How can we define a network state for a public transportation network?* **[Chapter 8]**

In this thesis, we limit the research to those networks for which we have sufficient data, both describing the networks and the related dynamic processes. Practically, this implies that our focus is mostly on road networks, except for the application in chapter 8. We show that some of the methods introduced in this thesis can be applied to networks at multiple levels of scale – both corridor-level (chapter 4) and network-level (chapter 7). Corridor-level refers to a single road stretch, whereas network-level refers to a large urban city network that might contain inner-city roads, highways, etc. All the methods and frameworks proposed in this thesis have been validated with either real or simulated data.

1.4 Research approach

Humans are the most sophisticated pattern recognizers in the world. When we are children, we learn to recognize visual patterns such as faces, animals, and plants using examples. We learn to identify color, color differences, edges, corners, intensity, etc., and we use these features to recognize complex patterns. The existence of MFD, a core concept of network dynamics, reveals that if you zoom out enough, regular patterns will emerge. This insight opens up many possibilities, as this is how humans learn patterns as well. We zoom out, find the high-level features and associate the examples we have seen with that object. And we use the same process irrespective of the type of object we want to visually recognize – faces, animals, numbers. Given that these visual features can recognize complex patterns, our assumption is that these features can also be used to recognize traffic patterns. We use the existence of MFD as the first feature and introduce the concept of human vision to further define and extract high-level features to understand the mobility patterns. In this dissertation, we draw on the human ability to recognize objects using data (examples), coupled with these physical attributes, or features to identify the complex patterns of a network. This new approach combines the field of pattern recognition – with a focus on computer vision - with the traffic domain. Incorporating the physical attributes related to human vision to recognize objects has been studied extensively in medical imaging, which combines computer vision with pattern recognition. However, using such attributes for recognizing complex traffic patterns is a new avenue of research.

To fully demonstrate the potential of our research, we discuss a novel data-driven OD estimation solution that incorporates these high-level features of traffic dynamics to unravel the unknown relationship between demand and supply space. Furthermore, we show that our method, together with coarsening, is scalable by applying the proposed method for nation-wide travel time predictions. Finally, we also pave the way to introduce these methods into other modes by proposing a new estimation method to represent the spatio-temporal network dynamics of a public transport network by taking inspiration from our analysis of the road traffic network.

1.5 Contributions

To address the identified problems, our main contribution to the field of traffic dynamics is a data-driven approach to retrieve interpretable features of network traffic states. This approach aims at improving the computational efficiency of dealing with large-scale networks using understandable high-level features. This paves the way for decrypting and making sense of these features so that we can further generalize them for different types of networks. The aim is to provide the tools and building blocks for using the available traffic data to their full potential so that in the age of big data, we can learn what kinds of patterns we are looking for. Our research offers the following contributions under different topics:

1.5.1 Tranport networks

- A new flexible heuristic method to substantially reduce the complexity of transport networks without significant loss of information. The method allows for coarsening subject to multiple objectives, such as road similarity, dynamic speed or travel time similarity, and more. **[Chapter 2]**

- An open-source implementation of this multi-scale network coarsening heuristic that can be readily used by both researchers and practitioners. **[Chapter 2]**

- A new compact construction to represent spatio-temporal data that is mapped on a graph using 3D maps. **[Chapter 3]**

- A new post-treatment method for clustering techniques to ensure the topological connectivity of the resultant clusters. **[Chapter 3]**

1.5.2 Feature selection for transport

- A new technique of representing congestion dynamics using custom feature vector that incorporates relevant features based on domain knowledge. These feature vectors can be easily extended to include contextual information. **[Chapter 4]**

- A new method to define traffic states using a high-level physical feature - shapes, which is a dominant feature used by humans to distinguish between objects. **[Chapter 4]**

- A new method to represent traffic as images and extract meaningful traffic states using models pre-trained on natural images. **[Chapter 5]**

- New insights that feature used by computer vision to recognize objects can successfully distinguish different traffic patterns as well. **[Chapter 5]**

1.5.3 Traffic patterns

- A new method to successfully compress multiple days into a handful of representative consensus patterns that are sufficient to explain the essence of the city dynamics. **[Chapter 3]**

- A new scalable framework that includes various complexity reduction methods such as coarsening and custom feature vectors to extract traffic patterns for a large-scale network and subsequently use it for travel time predictions. **[Chapter 7]**

1.5.4 Traffic demand

- A new data-driven framework for OD matrix estimation with only two behavioral assumptions and that does not require an equilibrium assignment or network loading model. The framework was also extended to be scalable for large networks. **[Chapter 6]**

- A new supervised learning method to estimate production and attraction patterns from 3D supply patterns. **[Chapter 6]**

1.5.5 Public transport

- A new estimation method to decompose passenger delay from individual trajectory into their corresponding network elements. Thus, the passenger delay dynamics of the transit network can be represented compactly which has many applications such as delay predictions and disruption detection. **[Chapter 8]**

- A new method to reveal recurrent patterns in the passenger delay of the public transport network. **[Chapter 8]**

1.6 Thesis outline

The chapters of this thesis are based on articles that are either published or are at time of writing under review. The text is completely identical to the published work. Consequently, the reader may encounter some degree of repetition between chapters. An overview of the thesis is presented in figure 1.2, with each box representing individual chapters. The chapters in this thesis are structured as follows:

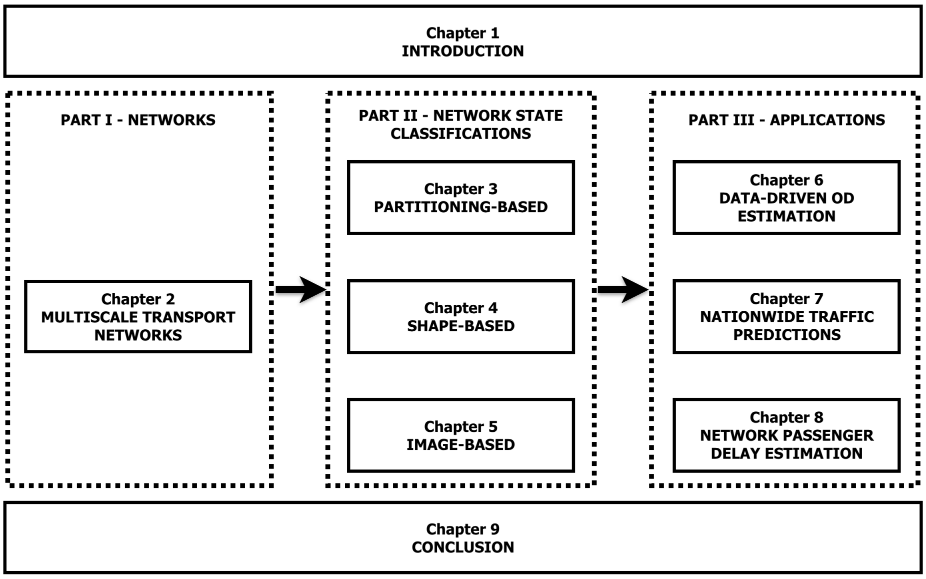


Figure 1.2: Outline of the thesis

Part I addresses the network complexity of large-scale transportation networks. It contains a single chapter, Chapter 2, which presents the heuristic method for automatically generating multiscale transportation networks without compromising key topological properties. It addresses a problem that is rarely discussed in the transportation literature, but that in our view is going to become increasingly relevant in the age of big data, where reducing the network complexity could easily determine the viability of the research for real-world applications.

Given efficient tools to reduce the network complexity, we can use the traffic variables to represent the network state. However, the dimensionality of the data can still be high depending on the space and time aggregation of the data. Thus, Part II explores different methods from fields such as graph partitioning, data point clustering and computer vision to extract the essence of the network traffic dynamics. Chapter 3 introduces the concept of 3D spatiotemporal maps to represent the network traffic states of a day for an entire city where both space and time are incorporated directly. These 3D maps are clustered using different partitioning techniques to define a day based on 3D homogenous speed zones instead of individual speed measurements. Consensus learning is used to produce a global pattern based on these 3D zones that fit multiple days, uncovering the day-to-day regularity. Chapter 4 investigates extracting shape from the spatiotemporal speed maps of highways and use this high-level feature to represent the traffic dynamics. In Chapter 5, the spatiotemporal maps are encoded as images and a full-fledged pre-trained deep-learning neural network is applied to the images to determine whether they reveal meaningful traffic states.

Part III is dedicated to the applications of looking at such network patterns at a higher abstraction level that goes well beyond road traffic. Chapter 6 shows how these 3D supply patterns can reveal an unknown correlation with demand patterns. This relationship, along with minimal assumptions, is used to estimate OD matrices in a data-driven framework. In Chapter 7, the shape-based approach is extended for network-wide analysis to reveal regularity between daily network patterns. This is compared with the partition-based approach to evaluate both of the methods' performance with respect to travel time prediction. The method is also applied to the entire Dutch road network to evaluate its performance on scalability. Chapter 8 describes how similar 3D network patterns can be estimated for public transport networks. For this, we decompose the passenger delay into its corresponding network elements by constructing a solvable system of equations from the passenger and transit vehicle trajectories.

Based on the aforementioned studies, Chapter 9 then presents the conclusion of the thesis, including the key findings, contributions, and recommendations for future research.

Part I

Networks

Chapter 2

Multiscale transport networks

Graphs at different scales are essential tools for many transportation applications. Notwithstanding their relevance, these graphs are created and maintained manually for most applications, in both research and practice, which is time-consuming and error prone. In this chapter, we develop a heuristic method for automatically generating multiscale graph representations without significantly compromising their topological properties. The method is demonstrated on the open street map network of Amsterdam with four different application cases. To support further research, an open-source implementation of the algorithm is made available.

This chapter is based on the following published paper:

Panchamy Krishnakumari, Oded Cats, and Hans van Lint. "Heuristic Coarsening for Generating Multiscale Transport Networks." IEEE Transactions on Intelligent Transportation Systems (2019). https://doi.org/10.1109/TITS.2019.2912430
Open-source code: https://github.com/Panchamy/Heuristic-Coarsening/wiki

2.1 Introduction

Directed graphs are vital tools in many areas of transportation science and practice. Particularly for the design and study of ITS; accurate graph representations at the appropriate level of detail are of quintessential importance. There are many readily available detailed directed graph representations. These representations are based on structured, reusable and standardized geographic and dynamic data such as open street map (OSM), and dedicated maps maintained by public administrations and road and rail operators. However, multiscale representations of these networks are more difficult to come by, despite their relevance.

Multiscale graph decomposition has been studied extensively in different fields such as scientific computing, gaming, Very Large Scale Integration (VLSI) system design, to name a few, using methods based on random walks, diffusion maps, spectral graph theory and various coarsening schemes [47–49]. In transportation, studies involving graph decompositioning focus mainly on graph partitioning problems for speeding up shortest path routing [49–52], and applications in the context of traffic assignment and/or equilibrium sensitivity analysis [53–57]. However, there are many other transport applications that may benefit from *consistent* network representations at different levels of scale, obtained from either detailed graph data (e.g. OSM) or coarse schematics. Examples include multiscale modeling and simulation [58, 59]; traffic estimation and prediction [60–64]; and even public transport service network analysis [65] to name but a few. In fact, there are very few areas within transportation science, where no schematic graph representation of either the physical or service network is needed. In practice today, such simplified schematic representations are often created and maintained manually, which is time consuming and error prone.

Given the wide range of applications for transportation network analysis, automation of the process of generating such coarser graphs from whatever data available offers scientists and practitioners large benefits in terms of effort spent. This calls for the development of a generic simple solution for generating and maintaining a set of mutually consistent and accurate directed graphs on the basis of the available geographic data.

Definition 1 *A multiscale graph is a set of increasingly coarser graphs G_i, G_{i+k}, ..., k = 1,2,3; representing the same transport infrastructure (or service network).*

We propose that a consistently coarsened graph G_{i+1} with respect to some finer base graph G_i should match the following criteria:

- *G_{i+1} has considerably fewer links and nodes than G_i*
- *G_{i+1} preserves important global topological characteristics of G_i (connectivity, shortest path distribution, diameter, total network length, centrality)*
- *G_{i+1} preserves important domain specific link and node attributes encoded in (or defined on) G_i*
- *G_{i+1} preserves consistent and accurate local (dynamic) topological attributes of G_i such as the shortest paths between origins and destinations (at approximately the same locations)*

Note that where we use the words ”preserve” (certain properties), one may also read ”gracefully degrades”, in the sense that in some cases, some degradation of information *density is inevitable when cutting out nodes and/or links. We return to this point in more detail in the validation experiments we provide.*

To this end, we propose a heuristic coarsening technique based on topological and/or data-driven information of the directed graphs. A constrained version of this coarsening approach using data-driven parameter is briefly noted in [63]. Here, we present a more detailed and generic framework that supports more widespread application. What makes our approach different from existing coarsening techniques tailored for specific transport applications—e.g. routing and assignment, which we discuss below—is that it provides a generic and flexible tool to simplify large transport networks into consistent coarser ones for many applications, ranging from topological analysis, modeling, simulation or visualisation, to name just a few. In our research lab this method has significantly reduced the effort in generating graphs for these common research tasks, and to the best of our knowledge no such generic method has been reported in the transportation literature and/or made available in code. We demonstrate the framework for four such applications on the large scale network of Amsterdam city. We use readily available topology information like the length, type, node-density, or other physical attributes of the graph to assign the weights and define the coarsening rules. The detailed graph representation and the physical attributes are obtained from Open Street Map (OSM), an open-source geographic data source. To support the research community in using and further developing efficient tools for graph coarsening we offer an open-source version of the code that implements our framework.[1]

The chapter is organized as follows: Section 2.2 first overviews the basics of network coarsening, using related work in (mostly) disciplines other than transportation. In section 2.3 we then discuss the proposed coarsening framework and the algorithms that will be applied to transportation networks. In section 2.4 we discuss the (Amsterdam) data; and the methods and performance indicators to assess how well our approach succeeds in generating consistent coarsened graph representations of the Amsterdam, the Netherlands. We quantitatively and qualitatively discuss the results in section 2.5 and conclude the chapter in section 2.6.

2.2 Related works

Within transportation, a limited number of studies report explicit algorithmic work on graph coarsening. In [56] and [54] a bush-based approach is proposed for replacing a regional network with a smaller one, containing all of the sub-network, and zones. Artificial arcs are created to represent “all paths” between each origin and sub-network boundary node, under the assumption that the set of equilibrium routes does not change. Similarly, [57] and [55] present method(s) for network aggregation under Stochastic User Equilibrium (SUE), using sensitivity analysis, in which the measure for assessing the resulting coarse network representation is based on how well perturbations in either demand or supply characteristics (i.e. changes in the OD matrix and/or changes in the link cost functions respectively) affect the result of the assignment. These methods are insightful, but based on a huge set of assumptions specific (and relevant) to the assignment problem, but not to other transportation problems. This hinders their relevance and transferability to other application domains. A

[1] https://github.com/Panchamy/Heuristic-Coarsening/wiki

second and related class of transportation problems for which graph coarsening plays an important role is speeding up shortest path routing algorithms [49–52, 66]. Bast et al. [67] gives an extensive overview of the multilevel methods for routing in transport networks. They conclude out that there are not many studies available within the transportation domain that discuss how—for a much broader range of applications other than assignment and speeding up routing problems—the topological characteristics of multiscale graphs differ with respect to the original fine-scaled graph. There is, however, a rich body of work available in other domains. Here, we present an abridged overview on coarsening research that is directly relevant for this work.

Multilevel methods were introduced during the 1990's to improve efficiency and quality of combinatorial optimisation problems [48]. Multilevel based algorithms try to solve complex problems by creating a hierarchy of problems that represent the original problem with fewer degrees of freedom. This process is coined coarsening. These hierarchies at different scales can be sequentially projected back to reconstruct the original problem space, known as uncoarsening. The coarsening and uncoarsening stages together constitute the multilevel framework. There are a couple of papers that provide an overview of multilevel techniques [68, 69]. In this work, we are only interested in the coarsening phase of the framework. Coarsening can be broadly classified into two types - strict and weighted coarsening. In strict coarsening, nodes are aggregated together to form a single node in the "coarsened space". The nodes in the coarsened space are called aggregates [48]. In weighted coarsening, each node is divided into fractions and these fractions can belong to different aggregates in the coarsened space [70]. More details on the principal differences between these two methods in graph terms can be found in [69].

Multilevel algorithms have been used in many disciplines including games [71, 72], mechanical engineering [73], infectious disease spread studies [74] and graph optimisation problems [69]. The graph partitioning problems and graph optimisation applications within transportation that focus on speeding up shortest path routing algorithms [49–51, 66, 67, 75–77] typically use strict coarsening for generating the hierarchies. That most multilevel methods for transport networks use hierarchical techniques makes sense, since road networks are inherently hierarchical. This was first fully exploited in the highway hierarchies (HHs) [50] method. The highway hierarchies contains two main building blocks - edge reduction and node reduction. Edge reduction preserves the edges in the middle of long distance paths and node reduction contracts nodes of degree one and two (i.e. nodes that only connect one or two adjacent links).

A simpler version of HHs are so-called contraction hierarchies (CHs), introduced by Geisberger et al. [49, 78], which are among the most effective (shortest route) speedup techniques. In general, coarsening techniques work by replacing edges in the graph with so-called shortcuts. In CHs, the shortcuts are added iteratively by contracting nodes following a given order of importance. The node ordering eliminates one of the major drawbacks of classical methods - the unpredictability of the contraction results. The main reason for this can be attributed to the random choice of nodes for the coarse level graph in classic methods [48]. Edge reduction is used in HHs to minimise the explosion of average node degree in the coarsened network but in CHs, this shortcoming is eliminated using a more sophisticated node contraction. Node contraction in CHs adds shortcuts only if shortest paths are preserved in the coarse scale after each node contraction. However, checking if the shortest path is preserved is time consuming. There are various solutions to speed up this process including limiting the space for shortest path search [49], using GPU [79] and customizable contraction hierarchies [72].

All these studies are based on graph methods that have not (yet) been explored in the traffic domain other than for routing applications. In this chapter, we seek a (heuristic) approach for network coarsening that can be used (insofar possible) in most transportation applications where graph coarsening might be useful. This method should offer a generic mechanism to assess the quality of the procedure based on topological information and/or data available in the application at hand. Based on the simplicity and success of CHs, we propose a heuristic approach with some of the building blocks of CHs—node ordering and node contraction. In [63], we briefly show how a constrained version of CHs can be easily used for network complexity reduction for traffic predictions. In the current contribution we further develop, formalize, apply and test the proposed approach to provide a more generic heuristic framework based on CHs that can be deployed in various applications.

2.3 Heuristic coarsening framework for multiscale graph generation

The general idea of coarsening is that, given graph G with n nodes, a more compact graph with a smaller number of nodes can be found which yields a good representation of the original graph. The multiscale graph G_{i+1} is constructed from the previous finer scale graph G_i by collapsing together the nodes and edges that have similar matching criteria. The matching can be computed in different ways, for example, by using aggregates [48]; by considering dominant route flows [80]; or based on node density [81]. In this work, the matching is based on the edge difference or variance of the edge weights. On top of the building blocks of CHs, we also use pruning to further reduce the network. This section will detail the steps required to derive these multiscale graphs. The coarsened graph can be constructed using the following four steps [48]. Note that each step may be detailed according to application-specific requirements or constraints.

1. Assign weights to the links in the directed graph;
2. Prioritize the nodes so that they can be removed in a strict order for generating the next coarsened level;
3. Determine contraction and pruning decision rules based on the edges weights, and;
4. Determine the new weights of the links for the coarse graph(for potentially a next iteration).

Notation : We use the standard notations used in graph theory as detailed in Table 2.1, illustrated using the example network shown in Figure 2.1. Here, the graph $G = (V,E)$ is a weighted directed graph where V is the set of nodes and E is the set of ordered pairs of edges or links. The edge $(u,v) \in E$, in Figure 2.1, is an incoming link with respect to node v where v is the target node and u is the source node. (v,w) and (v,x) are the outgoing links of node v where v is the source node and w and x are the target nodes. Arbitrary edge weights of the example network are also indicated in Figure 2.1.

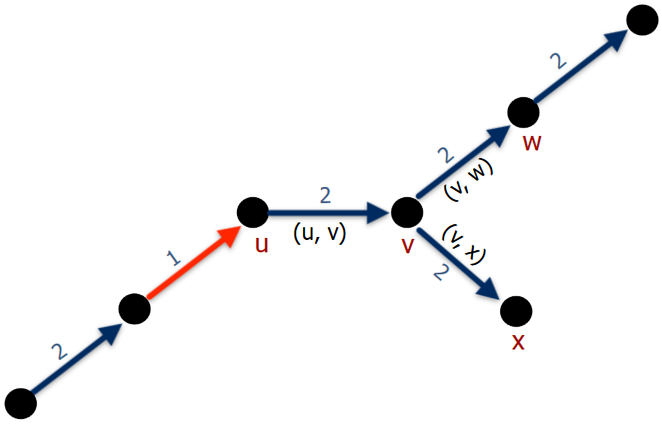


Figure 2.1: Example Network

Table 2.1: Graph Notations. Examples are based on Figure 2.1

.

$G_i(V,E)$	Fine network with set of nodes V and set of ordered pair of edges or links E
$G_{i+1}(V',E')$	Coarse network with the updated nodes V' $\subset$ V and the updated edge set E'
w_{uv}	weight of edge (u,v). eg: $w_{uv} = 2$
$N(v)$	neighboring links of node v. eg: $N(v) = \{(u,v),(v,w),(v,x)\}$
$N^-(v)$	incoming links of node v. eg: $N^-(v) = \{(u,v)\}$
$N^+(v)$	outgoing links of node v. eg: $N^+(v) = \{(v,w),(v,x)\}$
$\delta(v)$	$\|N(v)\|$, degree of node v which is the total number of incoming and outgoing links of node v. eg: $\delta(v) = 3$
$\delta^-(v)$	in-degree of node v. eg: $\delta^-(v) = 1$
$\delta^+(v)$	out-degree of node v. eg: $\delta^+(v) = 2$

2.3.1 Step 1 - Assigning edge weights

Edge weights are an essential element in solving graph problems such as coarsening, partitioning, etc. The weight can correspond to link length, width, type characteristics such as the link flow, inductance (for electric applications) or speed (for transport applications). We propose a generic weight measure, w_{uv} for the link (u,v) in the form of a weighted average over the application-relevant edge weights:

$$w_{uv} = \sum_{i=1}^{n} \beta_i w_{uv}^i \tag{2.1}$$

where n is the number of attributes, β varies typically between 0 and 1 and reflects the influence of these attributes on the generic edge weight, and w^i_{uv} is the i^{th} attribute of the link (u,v). Clearly, the value of β may differ based on the application.

2.3.2 Step 2 - Ranking the nodes

The order in which the nodes are removed is important for graph coarsening for computational reasons (only) [48]. In general, the seed nodes (nodes in the original graph considered for collapsing) are chosen randomly. In this work, we use a deterministic approach based on node ordering such that the nodes from the priority queue are contracted across the network in a uniform way, rather than contracting nodes randomly. For example, nodes can be ordered based on geographical scale (e.g. metropolitan areas; cities; neighbourhoods); traffic hierarchy function (freeways; motorways; main arterials; etc); spatial subdivision types such as grid-based [82] and polygon-based (e.g. clustering based on postal codes).

To illustrate this process, we use node degree (i.e. the number of edges connected to a node) as the decision rule for prioritising the nodes. The more neighbours the node has, the higher the rank, and the node will be contracted later. The underlying assumption is that a node that connects a lot of edges is likely to be more important for the transportation network and flow distribution—at least locally. Thus, the nodes are contracted by increasing order of node degree. Suppose, $(u,v) \in E$ where $u,v \in V$ then the rank of the nodes u and v will satisfy the following condition:

$$r(u) > r(v) \ , if \ \delta(u) > \delta(v) \tag{2.2}$$

where $\delta(u)$ and $\delta(v)$ are the degree of node u and v respectively. Thus, based on the contraction rule, v will be contracted before u. Node contraction affects the priorities of other nodes. Therefore, the priority queue is rebuilt after each node collapse. Since this process can become computationally expensive, we have implemented an iterative approach instead of re-evaluating the priorities, which is more efficient and provides robust results. In the iterative approach, we evaluate the priority once at the beginning of the iteration and collapse the nodes according to this queue. The neighbours of the nodes are updated at the end of the iteration. The iteration ends when all the nodes are visited at least once for collapsing consideration. The method converges when the iteration provides the same result as the previous iteration.

2.3.3 Step 3 - Defining the contraction and pruning rules

Once the nodes are sorted in increasing order, the contraction rules based on edge weights for collapsing them are defined. When a node is collapsed, its neighbouring links are joined together to form new links. Figure 2.2 presents some examples of different cases of node contraction. If the node collapse results in the same or even a larger number of links than before its collapse as shown in case (6) in Figure 2.2, there is no reason to collapse that node. Collapsing nodes without any regulation can lead to explosion of average node degree in the coarse level graph [49]. Therefore, a criterion $c_1(v)$ (Table 2.2) is set to decide if the contraction of a given node will contribute to a reduction in network complexity. Note that in case the application requires a coarse graph with fewer nodes but the number of links is not a priority, this constraint can easily be adjusted accordingly.

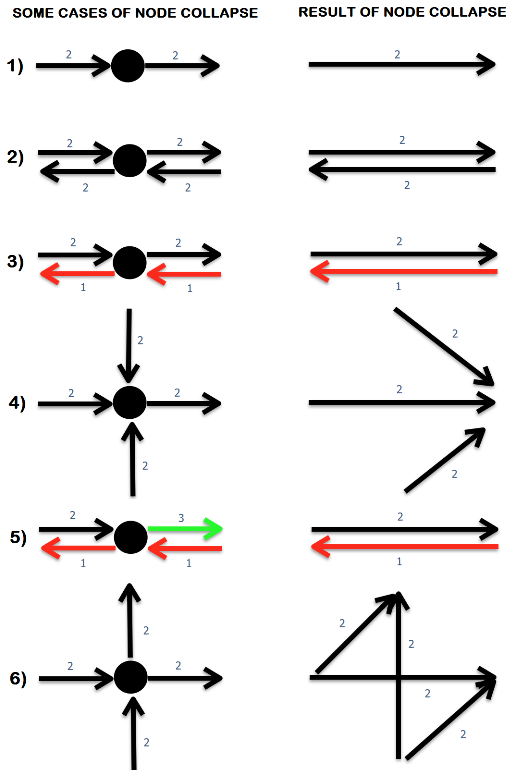


Figure 2.2: Examples of node collapse

The edge difference is used to define the next rule $c_2(v)$ for inclusion or exclusion of that node for contraction (Table 2.2). This rule is checked for each of the incoming-outgoing link pairs of the given node v. A lower threshold, ρ, implies a tighter constraint on the node collapse. The edge difference or variance of the edge weights w_{iv} and w_{vj} of node v, defined in equation (2.3), is used as the matching criterion.

$$\begin{aligned} \sigma^2(w_{iv}, w_{vj}) = |w_{iv} - \mu|^2 + |w_{vj} - \mu|^2, \text{where} \\ \mu = \frac{w_{iv} + w_{vj}}{2}, \end{aligned} \tag{2.3}$$

This is based on the idea that nodes should not be collapsed if they serve as the connection between two inherently different links. For example, a node that is connecting a highway and a city road is topologically important and results in small edge difference. If the ρ is set to 0 then this node would not be collapsed. If the ρ is set higher then the nodes that connect links with a smaller weight difference will be collapsed. For example, a node that connects a highway and a service road is hierarchically and topologically informative and hence should not be collapsed. Setting a proper ρ can prevent this. In the case (5) in Figure 2.2, if the ρ is set to 0, the node will not be collapsed as the edge difference is not 0. (i.e. there is a change in hierarchical level)

Table 2.2: Decision Rules

Node Collapse Rules

$$c_1(v) = \begin{cases} 1, & \text{if } \delta(v') < \delta(v) \text{ where } v \in V \text{ \& } v' \in V' \\ 0, & \text{otherwise} \end{cases}$$

$$c_2(v) = \prod \begin{cases} 1, & \text{if } \sigma^2(w_{iv}, w_{vj}) \leq \rho \\ 0, & \text{otherwise} \end{cases} \forall \begin{array}{r} i \in N^-(v) \\ j \in N^+(v) \end{array}$$

Node Deletion Rules

$$c_3(v) = \begin{cases} 1, & \text{if } \delta^+(v) = 0 \text{ or } \delta^-(v) = 0 \\ 0, & \text{otherwise} \end{cases}$$

$$c_4(u,v) = \begin{cases} 1, & \text{if } u = v \\ 0, & \text{otherwise} \end{cases} \forall\ (u,v) \in E$$

The most expensive computation for most of the methods mentioned in the literature, including CHs, relates to checking whether the shortest path is preserved after each node collapse [49]. This condition is not included in our heuristic approach under the premise that if the node collapse is performed according to the proposed method, there will only be minimal deterioration in the shortest path, which is acceptable for most applications. *In section 2.5, we will explicitly examine the validity of this assumption.*

Collapsing nodes can only reduce the complexity of the network to the highest edge difference threshold. To further reduce the network, pruning can be performed. Pruning refers to removing unimportant (in an application-specific sense) nodes or links from the network, instead of collapsing them. Depending on the application, pruning can be allowed or disabled. In this work, pruning is used for removing dead-ends (nodes without either incoming or outgoing links) and self-loops in the graph. Examples of these two cases are illustrated in Figure 2.3. Given that pruning is allowed, two conditions are defined to identify the dead ends and self-loops - $c_3(v)$ and $c_4(u,v)$, respectively.

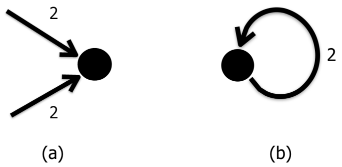


Figure 2.3: Example cases of pruning (a) Dead-ends (b) Self-loops

2.3.4 Step 4 - Assigning weights to new links

Assigning weights to the new links of the coarsened graph is the final step in the multiscale graph generation algorithm. The new edge weight is a function of the weights of the edges that are joined to make the new edge. Suppose the node v in Figure 2.1 satisfies both criteria $c_1(v)$ and $c_2(v)$, then the incoming-outgoing link pair (u,v,w) is joined to form a new directed link (u,w) and the weight of this link is determined as follows:

$$w_{uw} = f(w_{uv}, w_{vw}) \tag{2.4}$$

Depending on the edge weight, this function may represent any mathematical (e.g. logical or statistical) operation on the original weights. For example, if the edge weights represent the link length, the logical choice is a summation function. The same holds true if the edge weights represents costs or travel time. A common edge weight in different application domains is link capacity. To combine different link capacities, a minimum function is employed as illustrated in the examples in Figure 2.2. However, for traffic assignment applications, a minimum function might cause a reduction in overall network capacity. For this application, a stricter constraint with respect to pruning and edge difference combined with a maximum function might be more appropriate.

The pseudo-code for the heuristic coarsening is given in Algorithm 1. A step-by-step node collapse for the example network with pruning disabled is illustrated in Figure 2.4. The nodes are ranked based on their node degree. The new weights are computed using a minimum function. Figure 2.4(a) shows the graph with the ranked nodes. Since pruning is disabled, the node with degree 1 cannot be collapsed because of the initial stopping criterion $c_1(v)$. Figure 2.4(b) shows the result of the collapse of degree 2 nodes with the ρ set at 0. Lastly, the higher degree nodes that satisfy both the conditions are collapsed as shown in Figure 2.4(c).

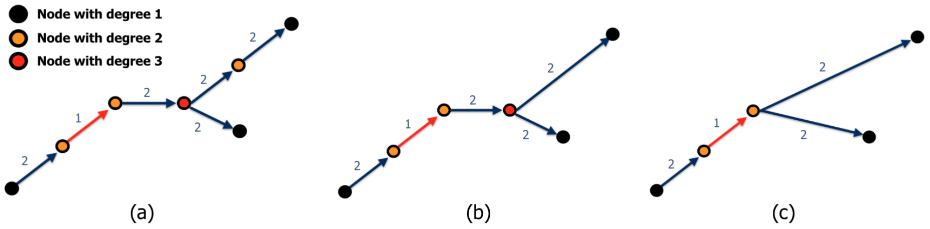


Figure 2.4: Node collapse results in the example network with ρ *0. (a) Nodes are marked based on their rank. (b) A node with degree 2 is collapsed. (c) A node with degree 3 is collapsed.*

2.4 Experimental setup

In this study, we present four application cases of the coarsening scheme. The applications illustrate various aspects of the algorithm and how restrictions can be added for different purposes. We study in detail whether the coarsening results satisfy the requirements of multiscale graphs proposed in Section 2.1 using several verification measures. Note that we do

Algorithm 1: Heuristic Coarsening Approach

```
Function
    Input  : Node list $V$, edge set $E$,iterations $M$, weights $w$, pruning and $\rho$
    Output: Coarsened edge set $E'$
    $E' \leftarrow E, i \leftarrow 0, iter \leftarrow 0, flag \leftarrow 1$
    while flag = 1 and iter < $M$ do
        /* Step 2 - Node ordering                                              */
        $E \leftarrow E', V' \leftarrow$ sorted $V$
        while $i \neq |V'|$ do
            $v \leftarrow V'[i]$
            /* Step 3 - Contraction rules                                      */
            if $c_1(v) = 1$ then
                Find $N^-(v)$ and $N^+(v)$. Eg: $(u,v),(v,w),(v,x)$
                Pair up $\{N^-(v), N^+(v)\}$. Eg: $(u,v,w),(u,v,x)$
                if $c_2(v) = 1$ then
                    $E' \leftarrow E' - [(u,v),(v,w),(v,x)]$
                    $E' \leftarrow E' + [(u,w),(u,x)]$
                    /* Step 4 - Assign weights to new links                    */
                    $w_{uw} = f(w_{uv}, w_{vw})$, $w_{ux} = f(w_{uv}, w_{vx})$
            $i \leftarrow i + 1$
        /* Step 3 - Pruning rules                                              */
        if pruning is True then
            foreach $v \in V'$ do
                if $c_3(v) = 1$ then
                    $E' \leftarrow E' - N(v)$
            foreach $(u,v) \in E'$ do
                if $c_4(u,v) = 1$ then
                    $E' \leftarrow E' - (u,v)$
        $V' \leftarrow$ update neighbors of $V'$
        $iter \leftarrow iter + 1$
        if $|E| = |E'|$ then
            $flag \leftarrow 0$
```

not claim these four cases provide conclusive evidence that under all application constraints the requirements in Section 2.1 are met. In this section, we explain the application cases; describe the data used; how the weights are assigned for the case study networks; and the verification measures.

2.4.1 Application cases

Before describing the four cases, let us briefly mention that for each of these we need to set two general parameters associated with our method: ρ (*threshold*) and *pruning*. For *pruning*, there are only two possible values; either enabled (1) or disabled (0). The ρ value corresponds to the restriction on the edge weight difference and in most applications, the ρ values are bounded as the edge weights are bounded. In this work, we demonstrate the coarsening for two instances of ρ values - minimum and maximum ρ for all the applications. This will define the upper bound and lower bound of the coarsening results for a particular application case. So, for each application there are four scenarios - pruning $[0,1]$ and ρ $[minimum, maximum]$.

Application I - Maximum network reduction possible without any restrictions

For the first application, we coarsen a large scale network with the simple aim of reducing the network complexity as much as possible. This objective may, for example, arise when visualizing properties of the network in time-critical applications (websites, mobile apps, etc). Clearly, the trivial maximum possible reduction of a network is to reduce it to a single node. However, the aim of our coarsening is to reduce the number of nodes as much as possible while reducing the number of links according to the constraint $c_1(v)$. Further reduction of coarsened graph by relaxing this constraint will lead to an explosion of links. Therefore, the maximum possible reduction of a network, in our case, corresponds to the maximum reduction of links and this is bounded by two parameters - pruning and ρ.

Application II - Network reduction restricted based on node type

The second application is a constrained version of the first where we try to coarsen the network while preserving all of the intersections. This case may, for example, arise when constructing a network model for traffic simulation with a focus on developing or ex ante evaluation of (coordinated) intersection control algorithms, or conversely, on driving behavioural models for conflict negotiation. An intersection is a node representing any kind of discontinuity such as a crossing, converging or diverging links, etc. In graph terms, we consider an intersection as any node with more than 1 outgoing link and 1 incoming link and also with $\rho = 0$ as edge difference is a form of discontinuity. The $c_1(v)$ is adjusted to represent this constraint as:

$$c_1(v) = \begin{cases} 1, & \text{if } \delta(v') < 2 \text{ where } v \in V \ \& \ v' \in V' \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

This aims at maximum network reduction while preserving the information of all discontinuous nodes.

Application III - Network reduction restricted based on area

The third application case pertains to having different scales within the same network, for example in case the study area is in fine detail whereas the area outside the study boundary is less detailed, which is particularly useful for hybrid modeling. By using an additional constraint for the nodes $c_5(v)$, we can create a subset of nodes as the exception node list to achieve this. This subset of nodes can be created manually or by defining a polygon boundary for the study area. In our case, we use a rectangular boundary for the study area defined as $[x_{min}, y_{min}, x_{max}, y_{max}]$, thus the constraint $c_5(v)$ is defined as:

$$c_5(v) = \begin{cases} 0, & \text{if } x_{min} < x_v < x_{max} \ \& \ y_{min} < y_v < y_{max} \\ 1, & \text{otherwise} \end{cases} \tag{2.6}$$

where x_v and y_v are the co-ordinates of the node v. Thus, we can create an exception list of nodes that are prohibited from being collapsed or deleted in that rectangular area. This is also useful for Dynamic Traffic Assignment applications such that certain origin-destinations can be added to the exception list so that they are not removed.

Application IV - Network reduction based on data driven parameters

The fourth and the final application is data driven coarsening. The difference between this and the first application is that now the edge weights used for the coarsening are aggregates of dynamic quantities. This can be useful for real time predictions for large scale networks where the complexity increases with the size of the network, as time-dependent networks are used for this purpose [63]. We used speed per link as the weights for coarsening the network for this application. The new weights of the links after node collapse are found using a mean function instead of a minimum function used as is done for the other applications.

2.4.2 Data

The real-world large-scale network of Amsterdam is used in the experiments (Figure 2.5). The Amsterdam network was extracted from the open-source open-street map(OSM) and contains 30 757 links and 34 935 nodes. Assigning weights to the links of the directed graph of Amsterdam is the first step of the heuristic graph coarsening. Given the limited availability of (open-source) data for *all* these links, we define the weight of an edge *(u,v)* with nodes *u* and *v* simply as:

$$w_{uv} = 1/t_{uv}$$

where t_{uv} corresponds to a value that depicts the type of the road network, which is readily available. Here, we use $\beta = 1$ in (2.1) because of the lack of additional meta information about the relative importance of these road types. In OSM, the type of the link refers to the standardized classification of the roads defined in OSM data source such as primary-link, secondary-link, access-ramp, etc, which is often used as a proxy for free-flow speed. There are 36 tags in OSM to define the type of the road segment. Each of the ordinal road classification tags is transformed into a numerical scale ranging from 1 to 36 based on the link importance of each tag described in [83] and this is assigned to t_{uv}.

For the Amsterdam network, there are 22 links which are tagged as 'road'. Since only 22 links are not properly tagged, the performance of the method is not significantly hampered. Another drawback of OSM network is that not all the nodes in the graph representation are correctly-noded [84]. This might lead to the graph being weakly connected with multiple connected components. For the Amsterdam network, there are 6759 such components with 90% of them having no more than four links. For most of the applications, these small "islands" are not that important for the study and pruning can be enabled to remove them.

For the first application, we use the Amsterdam network given in Figure 2.5. The edge weight is the numerically mapped road type, thus the minimum ρ is 0 and maximum is

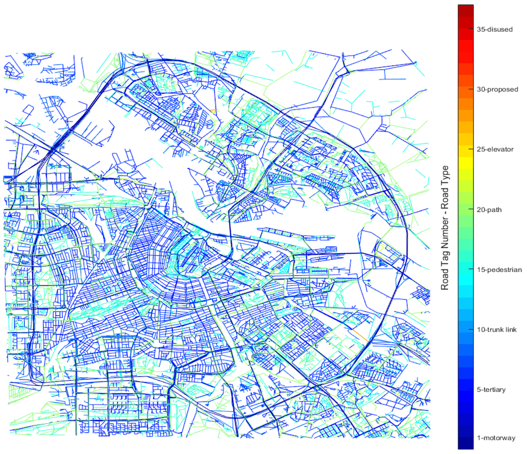


Figure 2.5: Amsterdam network with 30 757 links and 34 935 nodes

$\sigma^2(w_{min}, w_{max})$, where w_{min} is 1 and w_{max} is $\frac{1}{36}$ edge weights respectively. The same network is used for the second application case related to preserving the intersections. The rectangular study area (center of Amsterdam) boundary in relation to the larger Amsterdam network is shown in Figure 2.6 which is used for the third application.

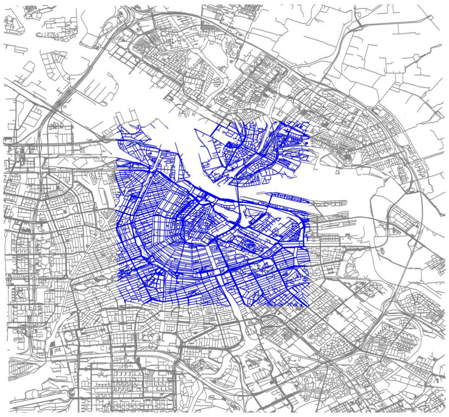
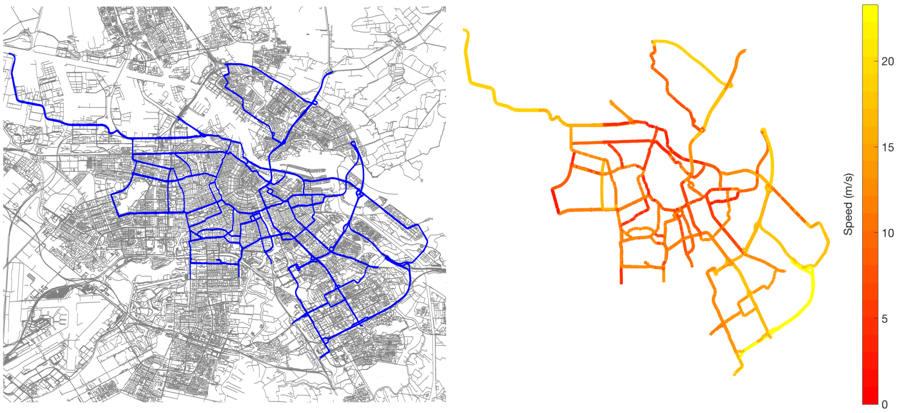


Figure 2.6: Application III - Study area (in blue) in relation to the Amsterdam network

In the fourth application case, we use travel time data from a license plate recognition system in Amsterdam to derive the speed per link. There were 314 pairs of start and end camera observations for the whole of Amsterdam network so the whole network is not completely utilized. The sub-network within the recognition system coverage is shown in Figure 2.7(a). The sub-network has 7512 links and 6528 nodes and it is a single connected component. The data preparation and conversion of travel time to speed per link is described in detail in [63]. The traffic state of Amsterdam at time 16:00 for one particular day is shown in Figure 2.7(b). The speed per link is used as the link weights of the sub-network.



(a)

(b)

Figure 2.7: Application IV - (a) Sub-network of Amsterdam (in blue) (b) Speed per link at time 16:00 for a particular day for the sub-network

All the applications have four scenarios as explained before with pruning $[0,1]$ and ρ $[minimum, maximum]$. The ρ for applications I to III corresponds to the change in functional road class of links and the change in speeds for application IV. To preserve the change in speed or road class, the minimum ρ is set to 0. The maximum ρ was set at 10000 (an arbitrarily high value) for applications I, III and IV whereas for application II the maximum ρ is also set at 0. This is because of our definition of intersection which prohibits coarsening nodes that have discontinuities in edge difference. The maximum number of iterations for all the applications was set at 10000, although the process stopped in all cases well before that.

2.4.3 Evaluation metrics

The multiscale graphs generated using our node collapse and pruning for the four applications are evaluated based on the five criteria proposed in Section 2.1, we use several graph measures and the computational performance for creating these graphs is also considered. All runs were done on a 64-bit computer with Intel Xeon CPU E5-1620 v3 processor of 3.5GHz, 16.0GB RAM and no GPU. In this section, we revisit the criteria for multiscale graphs and detail the four graph aspects used to quantify these.

Network reduction

The reduction in the network complexity is computed based on the reduction in the number of links. The network reduction, $r^{G_{i+1}}$, for a coarsened graph G_{i+1} in relation to the original graph G_i is defined as:

$$r^{G_{i+1}} = \frac{|E| - |E'|}{|E'|} * 100\ \% \tag{2.7}$$

where $|E|$ and $|E'|$ are the number of links in the original graph and coarsened graph, respectively.

Global topological characteristics

The desirable property of collapsing nodes and edges is that the topological characteristics of the graph are preserved, except when pruning is enabled which alters the topology of the network. For evaluating the global topological characteristics of the coarsened graph, we use five metrics: connectivity, trip length distribution, diameter, total network length and centrality.

Graph connectivity is measured using the number of connected components in a network. The connected components are found using depth first search based algorithm [85].

Trip length distribution is equal to the shortest path distribution (in distance) given that only the shortest path is considered between origins and destinations. The influence of coarsening on the shortest path between two arbitrary nodes is important since large shifts in shortest paths between origins and destinations imply fundamental changes in network topology relevant for many applications in transportation, ranging from simulation, to planning and to applied ITS tailored at providing information of the network state to travelers, operators and service providers. Checking this requirement is particularly interesting as this check is not built in the method itself. Assuming that there are N number of nodes in the coarse network, there are $N \times N$ OD (origin-destination) pairs in that graph and hence, $N \times N$ shortest paths. By comparing the distribution of these $N \times N$ shortest path cost of these ODs in the coarse and fine scale, we can observe the effect of coarsening on these shortest paths. The shortest path is found using Dijkstra's algorithm for weighted directed graphs [86] and the weights can be distance or travel time based on the application.

Diameter is the shortest path between the two most distant nodes in the network and is measured for a given network as maximum of all calculated trip length between all the origin-destination pairs in that network.

Total network length is the total length of the transport network (in km) and measured by summation of the length of all links in the network.

Centrality characterizes the (global) importance of a node in a network. In this work, we use betweenness centrality $g(v)$ as defined in [87, 88]:

$$g(v) = \sum_{s \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.8}$$

where σ_{st} is the number of shortest paths going from s to t and $\sigma_{st}(v)$ is the number of shortest paths from s to t through node v. Betweenness centrality can be considered as a proxy for flow throughput as a node is said to have high centrality if a large number of the shortest paths in the network go through that node. Thus, a similar betweenness centrality distribution needs to be maintained for the different scales as this implies similar hierarchy among nodes at different scales. This distribution is known to follow a power law for most transport networks, defined as:

$$g(v) \sim v^{-\eta} \tag{2.9}$$

where η is the power law exponent [87]. The value of power law exponent is typically in the range $2 < \eta < 3$ for scale-free networks, although not in all cases [89]. Our assumption is that the exponent value should not degrade significantly between different scales, at least for coarsening without pruning. So, if a network is originally scale-free, it is desired to maintain this property also for the coarsened network. Other than the power law exponent, it is also important to maintain the quality of the power law fit. In this work, we use the R^2 value as the goodness-of-fit measure for the regression model which varies between 0 and 1. A value of 1 implies a perfect fit to the data.

Domain specific characteristics

Depending on the application, the domain specific attributes of the network are either static link attributes such as functional road class, speed limits, capacities, geo-information or accurate aggregates/averages of dynamic quantities such as average flows, average speeds, travel times at a particular time for the network. These attributes define the link weights of the network which is then used for coarsening. In the algorithm, we have a hard constraint $c_2(v)$ (see Table 2.2), in which the ρ determines to what degree the attributes are preserved while coarsening. If ρ is 0 in $c_2(v)$,

$$|w_{iv}-\mu|^2+|w_{vj}-\mu|^2=0 \implies w_{iv}=w_{vj} \tag{2.10}$$

Thus, for links (i,v) and (v,j), the links are coarsened only if the link weights are the same and hence the link weights are fully preserved for ρ = 0 and by mathematical induction, it holds for all links in the network and for different ρ values.

Local topological characteristics

While the trip length distribution shows the global trend of shortest paths in that network, it does not show if the same shortest path is maintained in the coarsened graph as the original graph. To observe this, we use the shortest path deterioration of each OD pair st of $N \times N$ OD pairs for a given coarse scale graph G_{i+1} defined as:

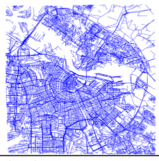
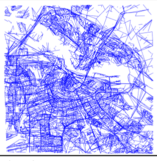
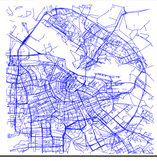
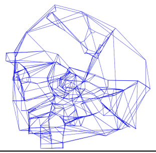
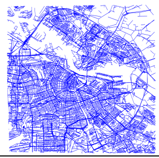
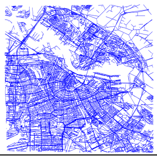
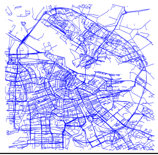
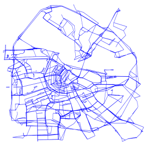
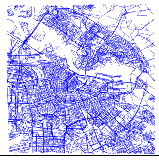
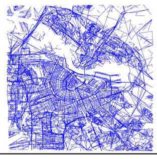
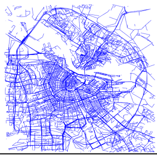
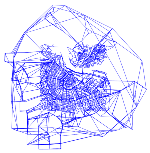
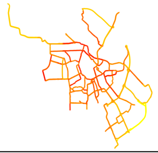
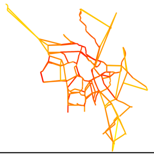
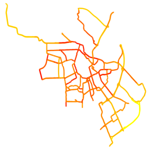
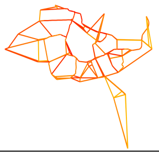
$$D_{st}^{G_{i+1}} = \frac{c_{st}^{G_{i+1}} - c_{st}^{G_i}}{c_{st}^{G_i}} * 100\ \% \tag{2.11}$$

where $c_{st}^{G_{i+1}}$ and $c_{st}^{G_i}$ is the shortest path cost between nodes s and t in the coarsened graph G_{i+1} and the original graph G_i respectively.

2.5 Results and discussion

In this section, we present the results of the four applications and their corresponding scenarios. We evaluate application I in depth in relation to the evaluation metrics since this application aims at the maximum possible reduction of network size without any restrictions using our coarsening framework. Hence, it is expected to manifest the maximum deterioration of the topological properties and thus offers the most conservative performance assessment. We discuss the other applications more briefly. Table 2.3 shows the coarsening results for the four applications and their scenarios.

Table 2.3: Coarsening results of the four applications.

		Scenarios			
		pruning = 0 ρ = minimum iterations = 1	pruning = 0 ρ = maximum iterations = maximum	pruning = 1 ρ = minimum iterations = 1	pruning = 1 ρ = maximum iterations = maximum
Application I	Results				
	Network reduction	21%	47%	59%	96%
	Connectivity	6759	6759	1103	1
	Network length	3746 km	4945 km	2268 km	1213 km
	Diameter	38.80 km	38.69 km	38.80 km	29.52 km
	η, R^2	3.29, 1.00	3.08, 0.99	2.98, 0.99	2.13, 0.90
	Computation time	10 - 15 minutes			
Application II	Results				
	Network reduction	17%	26%	57%	88%
	Connectivity	6759	6759	1139	7
	Network length	3617 km	3617 km	2201 km	1036 km
	Diameter	38.80 km	38.80 km	38.80 km	38.80 km
	η, R^2	3.25, 1.00	3.09, 0.99	2.99, 0.99	2.26, 0.96
	Computation time	10 - 15 minutes			
Application III	Results				
	Network reduction	14%	32%	43%	68%
	Connectivity	6759	6759	2379	1588
	Network length	3716 km	4580 km	2577 km	1782 km
	Diameter	38.80 km	38.69 km	38.80 km	32.13 km
	η, R^2	3.10, 0.99	3.08, 0.99	2.94, 0.99	3.19, 0.99
	Computation Time	6 - 10 minutes			
Application IV	Results				
	Network reduction	38%	74%	40%	85%
	Connectivity	1	1	1	1
	Network length	311 km	366 km	309 km	249 km
	Diameter	51.27 km	48.39 km	51.29 km	19.03 km
	η, R^2	1.95, 0.81	1.66, 0.89	1.94, 0.81	1.49, 0.75
	Computation time	30 - 78 seconds			

For application I, the node coarsening leads to a network edge reduction of $21-47\%$ without pruning giving the same number of connected components as in the original network (6759 components). Thus, the connectivity is preserved without pruning. Since the approach is iterative, the 47% reduction is achieved after 12 iterations. This implies there are 12 multiscale graphs available with a reduction within the range of $21-47\%$ respectively compared to the complete Amsterdam network as shown in Figure 2.8. *With* pruning, the edge reduction ranges between $59-96\%$, resulting in a more compact network of 1103 components for the minimum ρ and a single component for the maximum ρ after 15 iterations as shown in Figure 2.9. (Note that only pruning has an effect on the number of components, node collapsing has no influence on the network connectivity.) The computation time for coarsening is 10 to 15 minutes with and without pruning respectively. We return to the results of the other applications listed in Table 2.3 further below.

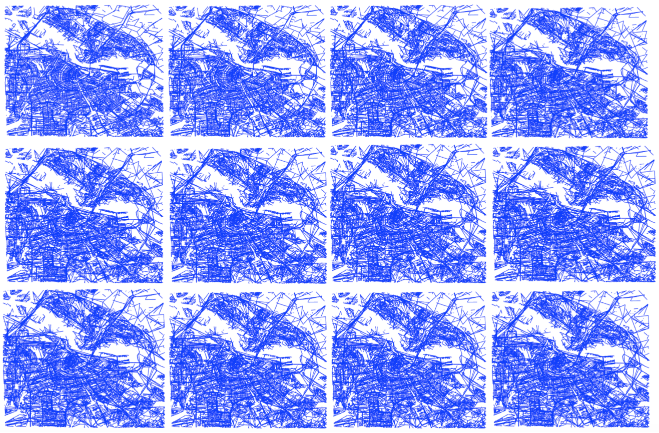


Figure 2.8: Evolution of the Amsterdam network during 12 iterations for scenario 2 of application I (without pruning), with network reduction ranging from 21% to 47%.

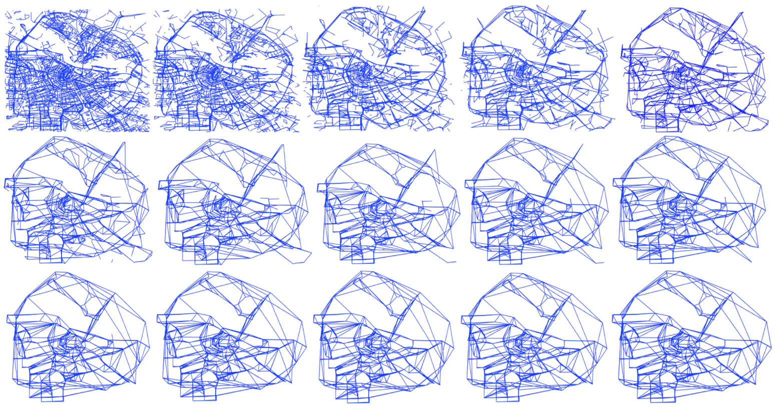


Figure 2.9: Evolution of the Amsterdam network during 15 iterations for scenario 4 of application I (with pruning), with network reduction ranging from 59% to 96%.

We first examine some global topological characteristics of the multiscale graphs resulting from application I. First, the trip length distribution of the network. There are 427 nodes remaining in the final coarsened graph (application I, scenario 4). Therefore, there are 427 × 427 origin-destination (OD) pairs leading to 182 329 shortest paths. The link length is used to estimate the shortest path cost for application I. The trip length distribution of these shortest paths in the original and coarsened network are shown in Figure 2.10. As can be seen, the trip length distribution of the original and coarsened graph is quite similar, even for the maximum network reduction. We have also investigated the complete trip length distribution of the original and coarsened graph and found that while the shape of the distribution remains similar, some of the ODs have deflated shortest path lengths, especially due to node removal. This is also evident from the network diameter reported in Table 2.3 for all applications.

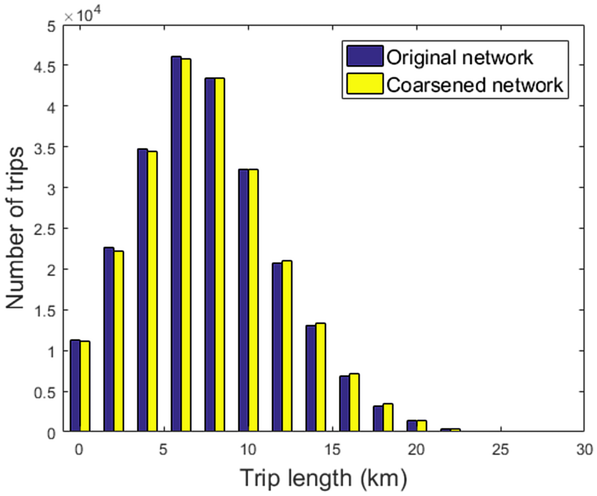
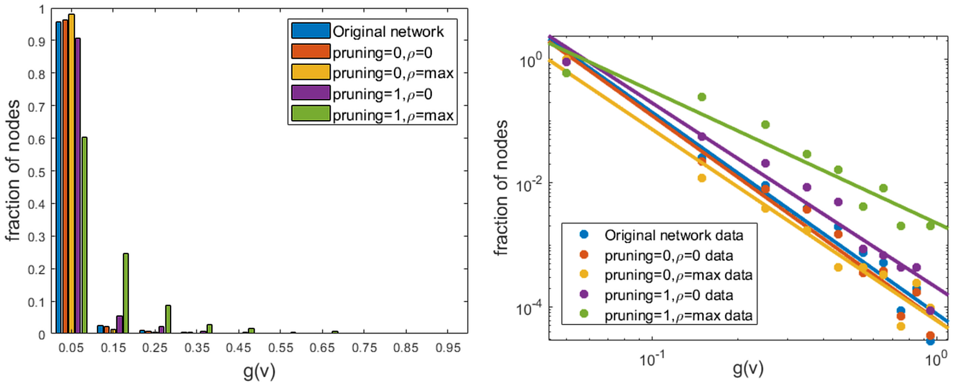


Figure 2.10: Trip length distribution of the OD pairs in the Amsterdam network for application I.

The diameter of the original network is 38.92km. From Table 2.3, it can be seen that the diameter of the network decreases for all scenarios. The slight decrease in diameter when pruning is disabled is unexpected as we expect it to be unchanged. But, this is because of shortcuts which are created due to node coarsening that are duplicates of links already existing in the network. Consequently, there are two separate links connecting the same nodes with different link length. Since the minimum of these length will be used for computing the shortest paths that traverse this link, there is a decrease in the diameter.

The other global characteristics considered is the total network length. The total network length of the original network is 3622km. From Table 2.3, it can be seen that the network length increases significantly for coarsening without pruning while decreases for coarsening with pruning. The decrease in network length is expected as links are removed due to pruning. However, the increase in network length is due to the shortcuts created during coarsening leading to direct connections between nodes as shown in example cases (2) and (6) in Figure 2.2.



(a) *(b)*

Figure 2.11: (a) Influence of coarsening on node centrality distribution; (b) Log-log plot of node centrality with the data as dots and the corresponding line showing the power law fit.

We also consider node centrality characteristics of the multiscale graphs, which exhibit similar distribution as compared to the original graph as illustrated in Figure 2.11(a). This distribution of the multiscale graphs follows a power law and the corresponding power law parameters that define the relationship between centrality and the number of nodes as $g(v) \sim v^{-\eta}$. Estimated power law parameters for the relation of the betweenness centrality versus the number of nodes are shown in Figure 2.11(b). The original Amsterdam network has a power-law with exponent $\eta \approx 3.25$ with $R^2 = 1.00$ whereas for coarsening without pruning has $\eta \approx 3.29 - 3.08$ with approximately the same R^2 as shown in Table 2.3. The power-law exponents for coarsening with pruning is $\eta \approx 2.98 - 2.13$.

Thus, these multiscale graphs have similar exponent as that of a scale-free network by employing pruning which is significantly different than that of the original network. However, the exponents without pruning are similar to those of the original network and this is also evident from Figure 2.11(b). The degradation of the exponent in case of pruning is probably because of the high number of disconnected components in the network. This assumption is confirmed in application IV which have a single component. For application IV, the original network has a power-law exponent of 2.43 with $R^2 = 0.80$ whereas the coarsened networks have exponents less than 2.00 while the R^2 remains similar. Thus, all the multiscale networks seem to follow a power law with similar goodness-of-fit as the original network, however the scale-free property of these networks are inconclusive using just the power law exponent. The exponent degradation between all scenarios in all applications is approximately 1, which seems marginal.

Figure 2.12 shows the influence of coarsening on the length of the shortest paths in the graphs for application I. The trip length distribution of the multiscale graphs were shown to be similar; however the actual shortest path has to be studied in detail to check if it is maintained within the different multiscale graphs. We used relative shortest path deterioration to check this local characteristics for application I. As can be seen from the figure, the effect of coarsening on the shortest path is minimal. With more than 95% of the nodes removed, there is only a maximum deterioration of 0.025% in the shortest path cost. Thus, even extreme pruning only has a marginal influence on the shortest path, which is an important finding for the application scope. For most of the traffic applications, a 0.025% deterioration in distance is negligible.

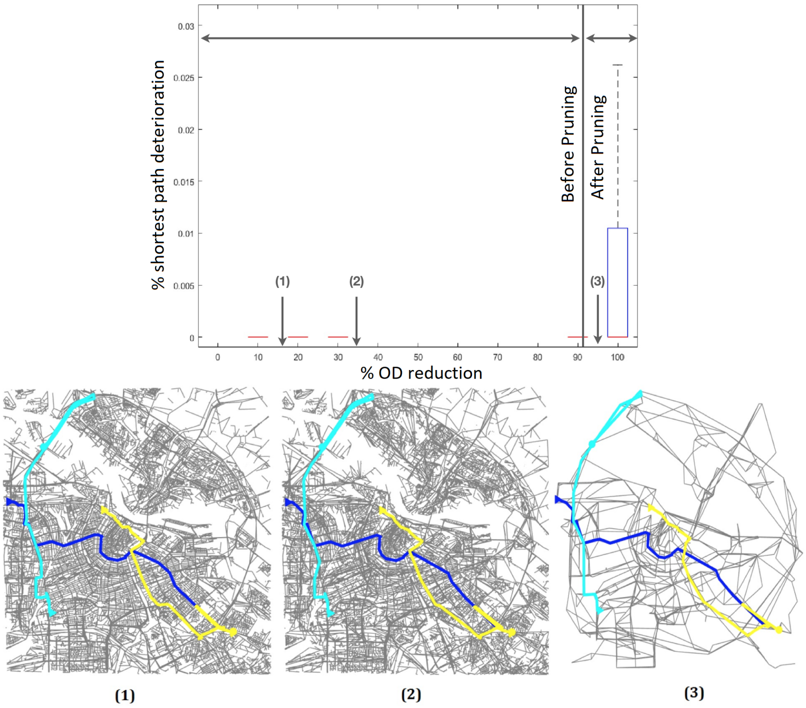
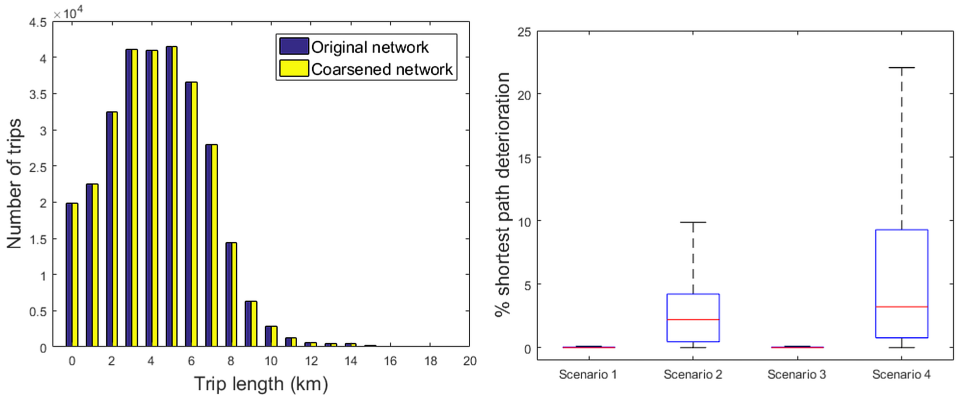


Figure 2.12: Boxplot of shortest path deterioration versus the share of nodes removed (OD reduction) for all iterations in all scenarios for application I. (1), (2) and (3) represent the multiscale graphs at 16%, 34% and 93% OD reduction respectively. Same three shortest paths with the largest deterioration are shown in (1), (2) and (3) for illustration.

Table 2.3 summarizes the results for all the applications. The node coarsening for the preservation of intersection application provides a network reduction of $17-26\%$ without pruning after 7 iterations and $57-88\%$ with pruning after 17 iterations. The node coarsening for the study area application provides the least network reduction of $14-32\%$ without pruning after 8 iterations and $43-68\%$ with pruning after 13 iterations. The higher number of components (see Table 2.3) and lower network reduction compared to the previous applications are because of the unaltered study area which corresponds to 30% of the network size and comprises of 1500 components. This also corresponds to the low degradation of the values for the characteristics such as the diameter, network length, power law exponent and fit compared to application I as reported in Table 2.3. All the centrality distribution of multiscale graphs follows the power law with similar R^2 value as those of the original network. The node coarsening results for the data driven application case provides a network reduction of $38-74\%$ without pruning after 10 iterations. With pruning, it provides a reduction of 40 − 85% after 86 iterations.

For application IV, travel time is used to compute the shortest path. The travel time deterioration for the coarsened graph and the original graph for application IV for different scenarios are shown in Figure 2.13(b). There are 538 × 538 OD pairs for this application resulting in 289 444 shortest paths. There is a maximum deterioration of 22% in travel time. However, the shortest path distribution based on distance as the cost, shown in Figure 2.13(a), has an approximately zero deterioration, i.e. very minimal undesired alterations. This is because the distance of the link is preserved when coarsening whereas the link weights (average speed) are skewed because of averaging when $\rho > 0$. The travel time is preserved better for smaller ρ (scenarios 1 and 3) with a maximum deterioration of 0.1%.



(a) *(b)*

Figure 2.13: Shortest path deterioration based on distance and travel time for application IV

2.6 Conclusion

In this chapter, we propose a generic heuristic coarsening technique for generating multiscale graphs. Compared to existing graph based methods popularized in Experimental Algorithms, we presented a heuristic method that is directly applicable and versatile for many transport applications. The method comprises of two main building blocks: node ordering and node contraction. Importantly, the explosion of the average node degree in the coarse network is eliminated by a simple decision rule, which can also be easily relaxed. By setting an edge difference variance threshold ρ, the graph can be generated at the required coarsened level. The method is demonstrated on the Amsterdam city network for four applications. The method was able to successfully reduce the Amsterdam network by up to 96% of its original size at a computation time of no more than 15 minutes with a limited loss of information as indicated by the preservation of key network characteristics. For offline use (e.g. transportation planning, simulation model generation), this performance is considered satisfactory.

An initial verification study analysing some of the topological measures provide some important observations. For global and local topology preservation applications, pruning must be disabled as it might remove a node or link that is necessary for the shortest path and connectivity. However, for applications that require a more compact network, pruning is useful. With more than 95% of the nodes removed, we found a maximum deterioration of just 0.025% in the shortest path of the ODs. This confirms the assumption that it is not necessary to check if the shortest path is preserved after each node collapse. We also showed that all the multiscale Amsterdam networks fit the power law with similar goodness-of-fit as the original network with a maximum deterioration of approximately 1 for the power law exponent. One of the key finding from the four application cases was that there is at least a minimum reduction of $14-38\%$ in the network complexity with the strictest constraints. This decrease can have significant impact on computational efficiency especially when dealing with time-dependent networks.

From the results we conclude that the algorithm offers an effective coarsening solution for many transport applications. The method can be used as a preprocessing step for many network-wide applications - either to reduce the network complexity or to generate multiscale representations of these networks. Application include a range of ITS applications from design to control such as hybrid modeling, traffic assignment, real time predictions, visualisations, etc. Most of the operations on large-scale networks require hardware capabilities or high-level optimisations to be viable for real-time or even exploratory studies. This simple open-source algorithm is an initial step to remove some of these complexities at network-level before adding high dimensional information on the graph. This is especially important in the age of big data, where data availability may no longer be the problem, but rather the efficient capabilities to use the data may prove a new bottleneck for modelers and analysts.

Part II

Network state classifications

Chapter 3

Partitioning-based classification

In the previous chapter, the framework for reducing network complexity was discussed. Now, we can add the traffic variables on top of the network to represent the network traffic state. However, the complexity of the network becomes prohibitive when the data is added on top of it. In this chapter, the concept of 3D spatiotemporal maps is introduced to represent the network traffic states, and clustering is proposed as a technique to compress the high-dimensional data.

In the first part of the chapter, these 3D spatiotemporal maps are clustered using different partitioning techniques from different domains to build homogeneous regions, which theoretically should have a well-defined MFD. However, within the scope of this thesis, we see these regions mainly as a dimensionality reduction technique. The 3D regions can be used to define the traffic state of a whole network for a single day instead of the thousands of time series otherwise needed for each link. In the second part of the chapter, a systematic approach is proposed to gather days with similar 3D regions and use consensus clustering methods to produce a unique global pattern that fits multiple days, uncovering the day-to-day regularity.

My personal contributions to this body of work are mapping the data to a geographic information system network, coarsening the network to reduce the complexity at the city scale, developing the new post-treatment methodology that ensures that the 3D regions are connected and estimating the travel time through the 3D spatiotemporal maps.

This chapter is based on the following published papers:

Clelia Lopez, Panchamy Krishnakumari, Ludovic Leclercq, Nicolas Chiabaut, and Hans van Lint. ”Spatiotemporal partitioning of transportation network using travel time data.” Transportation Research Record 2623 (2017): 98-107. https://doi.org/10.3141/2623-11

Clelia Lopez, Ludovic Leclercq, Panchamy Krishnakumari, Nicolas Chiabaut, and Hans van Lint. ”Revealing the day-to-day regularity of urban congestion patterns with 3D speed maps.” Scientific Reports 7 (2017): 14029. https://doi.org/10.1038/s41598-017-14237-8

3.1 Spatiotemporal partitioning of transportation networks

3.1.1 Introduction

Graph partitioning is a common challenge in different fields, such as transportation [90, 91] and image segmentation [92]. Partitioning a heterogeneous network, such as an urban network, into homogeneous zones can be extremely useful for many applications. We distinguished, at least, three applications that can be improved by using 3D homogeneous regions: (i) traffic control, (ii) macroscopic traffic modeling and (iii) route guidance.

1. For traffic control purpose, different traffic management schemes can be identified for regions of a heterogeneous network [93, 94]. Traffic signal control is computationally expensive for a large network [95], especially if the scheme is required to be generated in real-time [96]. Developing a scheme at a regional level is computationally more plausible than at a link level. A time adaptive scheme can be considered from the 3D regions.

2. The Macroscopic Fundamental Diagram (MFD) is more likely to be well-defined in a homogeneous network [19, 97–100]. Refine the equilibrium analysis by zone is a promising approach to investigate the effect of route choice behavior [101]. Partitioning methods make it possible to define sub-regions within a network which is essential for a multi-reservoir modeling approach such as the MFD modelling framework [99, 102]. This macroscopic scale model facilitates the development of traffic management strategies.

3. Tour planning, trip advisors and dynamic route guidance can be refined by network partitioning results. The routing models can calculate the least cost routes including the day-to-day 3D regions, i.e. the travel time objective function $f(t,x)$ is refined by the space time properties of the 3D regions. An investigation can be done to identify the best time to start a journey by minimizing the total route time through spatio-temporal network.

To this end, [90] investigates the performance of $k-$means in partitioning urban networks by considering spatial locations of the road as new features in the data. [91] elaborates a second method based on the definition of a similarity matrix between observations and the application of the Ncut algorithm [92]. Such regions are defined in 2D, i.e. only measurements of a single time period are considered to identify homogeneous areas. The question of traffic dynamics is only addressed by iterating the algorithms for each time step without direct connections between the 2D clusters (quasi-stationary approximation). It should be noted that cluster compactness is a central property in these studies because the main targeted application is traffic control.

To perform network partitioning, both the network topology and the link speeds for all time periods are needed. However, real data is often flawed and incomplete, especially for data collected by classic urban measurements. Therefore, data preparation is needed to create a validated dataset for partitioning. The aim of the data preparation is (a) travel time outlier removal, (b) coarsening the large scale network to improve the computation time, and (c) speed estimation including an extension for missing data. In this chapter, we introduce methodologies that address these issues. We use real data gathered from Amsterdam urban area. The network topology is derived from both cameras and Geographic Information System (GIS) data from Open Street Maps. In the Amsterdam case, not all speeds for every time period are available. This necessitates the development of algorithms to impute these missing data. We discuss the estimation of missing data and how to minimize the problem of missing data in section 3.1.2. Based on a complete (albeit partially imputed) record of all data, we can now use the Amsterdam network to assess the performance of different partitioning techniques.

In this chapter, we investigate two classes of clustering techniques an extension of Ncut based on a new expression of the similarity matrix, called snakes [90, 91] and classical methods such as DBSCAN and Growing Neural Gas (GNG) from the data-mining field. These clustering techniques are used to construct 3D clusters of road links based on their average speed. We want to ensure that all clusters contains a single connected component (CC) in order all links be reached within the same clusters. This requirement is central for application like travel time estimation or macroscopic traffic modeling. Clustering consists of finding the optimal decomposition of the graph into connected clusters with lowest internal variance of the speed while it retains a reasonable 3D compactness. However, the classical methods that are used in this chapter DBSCAN and GNG produce unconnected clusters. Hence, a new post-treatment method is introduced in this work to generate connected clusters from unconnected clusters in 2D and 3D.

In section 3.1.2, we describe the methodology for building both a validated network and speed dataset for partitioning. The construction of the 3D connected clusters using the three clustering techniques along with the post-treatment and cross evaluation criteria are described in section 3.1.3. Section 3.1.4 provides the results of this preliminary evaluation and section 3.1.5 concludes the chapter with the major findings of the chapter and future recommendations.

3.1.2 Data Preparation

Network Reconstruction

In our study, raw data are the individual travel times from the city of Amsterdam. The data includes the ID and location of the start camera and end camera, recorded individual passing times and travel times for 41 days. There are a total of 314 unique pairs of start and end camera observations in the data for all the days. The first step to build a network from this dataset is to create a route table. For this purpose, we need to locate the camera on the Amsterdam GIS network obtained from OpenStreetMaps which consists of 147059 links/edges as shown in Figure 3.1(a). The mapping of the camera coordinates to the GIS network is required since these coordinates may not be located in the road, hence location correction is needed. This snapping is done by finding the point-to-segment distance to find the nearest location in the network. Once the coordinates have been snapped to the network graph, we find the path between the start and end camera location. There are various path finding algorithms available in the literature. For this work, the shortest path was found using the algorithm employed in OSRM (OpenStreet Maps Routing Machine) which is based on contraction rules [103]. This algorithm is fast and robust for real time applications as well. There were 314 pairs of start and end cameras leading to 314 shortest paths. Mapping these to the GIS Amsterdam network provides a network with 7512 links as shown in Figure 3.1(b).

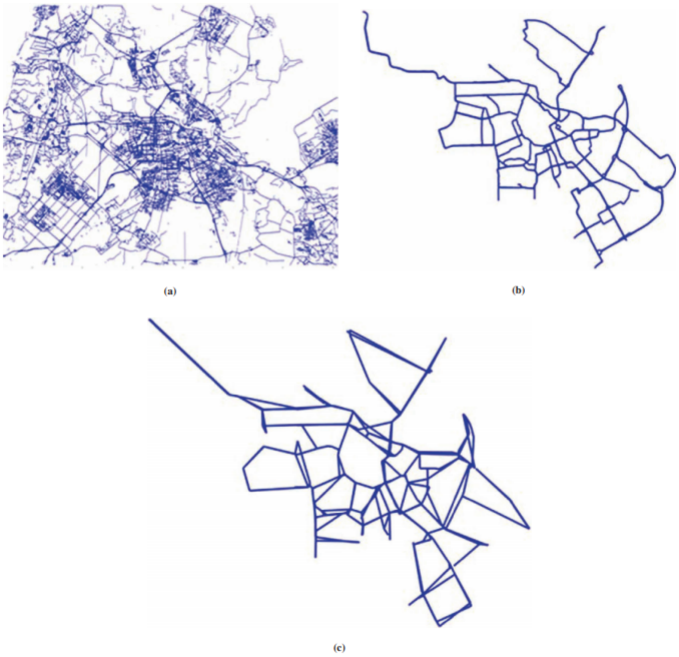


Figure 3.1: Camera data to validated network (a) Amsterdam GIS network with 147059 links, (b) Shortest paths mapped to the network - 7512 links (c) Coarsened network with 411 links.

Network Coarsening

After mapping the shortest path to the Amsterdam GIS network, there are still 7512 links, which is quite large. Hence, we employ network coarsening techniques to remove nodes that satisfy certain criteria. The idea behind coarsening is that a multiscale graph G_{i+1} can be constructed from the previous fine scale graphs G_i collapsing together the nodes that have similar matching criteria. The matching criteria can differ according to the application. In this chapter, the matching criterion relates to the differences in edge weights. In our case, these weights represents the speed of each link/edge. If the edges have the same weight, the node that connects the edges will be collapsed/removed. The network coarsening in this chapter is based on a constrained version of contraction hierarchies [103].

The construction of the coarsened graph in this work is based on three steps: (a) the nodes are prioritized or ranked for contraction, i.e. a node with a lower rank will be removed before a node with a higher rank, (b) the contraction rules are determined based on the edge difference, and (c) the new weights of the link for the coarse graph are calculated. In this work, the contraction rules are governed by the edge weights and the connectivity of the network. The weights for each link/edge is the estimated speed of each link. The weight w_{uv} link (u,v) is the speed of the link s_{uv}. As we only require speed for one time slice for assigning the weights, a peak period time slice (16:00) was chosen. This is because the network will exhibit the most variance during peak period with most links having different speeds at this time. If we chose a non-peak period, these variances will be smoothed out. A more detailed description of node ranking and the contraction rules are given below.

Node Ranking

The order in which the nodes are removed is important for graph coarsening. There are different node ranking or ordering techniques as introduced in relation to contraction hierarchies [103]. In this work, the nodes are ordered based on their densities. The more number of neighbors the node has, the higher the rank is, and the lower the chance that the node will be removed. A dense node might connect a lot of edges and might be important for transportation purpose application. Given a graph $G = (V, E)$ node set V and an edge set E. Suppose, $(u, v) \in E$ where $u, v \in V$ then the rank of the nodes u and v will satisfy the following condition:

$$r(u) > r(v), \text{ if } n(u) > n(v) \tag{3.1}$$

where $n(u)$ and $n(v)$ are the number of neighbors of node u and v respectively. Thus, based on the contraction rule, v will be contracted before u. The neighbors of the node are found by determining all the incoming and outgoing links from the node. A link (u, v) is said to be incoming w.r.t node v if v is the target node of the link and outgoing if v is the source node. Once the neighbors are found for all the nodes, the nodes are sorted based on their ranks in ascending order with the lowest rank first.

Contraction Rules

Once the nodes are sorted according to the rank, the contraction rules are used to remove them. First, a criterion has been set to decide if the given node is eligible for contraction to decrease the complexity. The criteria are that if the node removal results in the same or more number of links than before removal, then the node is not removed. Table 3.1 presents an overview of different cases of node contraction. The link color represents weights; different color implies different weights. For example, in case (5) of Table 3.1, removing the nodes results in 4 new links which is the same number of links as that before node contraction. Hence, this node will not be considered for removal. It was also observed that contracting nodes with more than four neighbors, in the majority of the case, leads to 5 or more links. Therefore, an initial constraint has been imposed on the node contraction only to contract nodes with neighbors less than or equal to 4 links as given below:

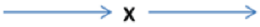
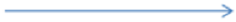
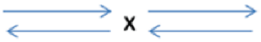
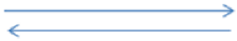
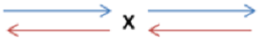
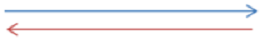
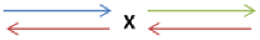
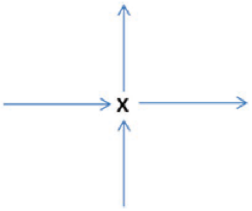
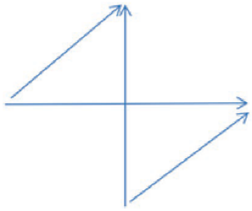
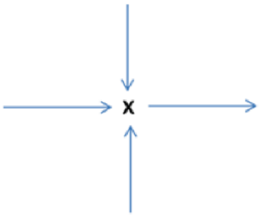
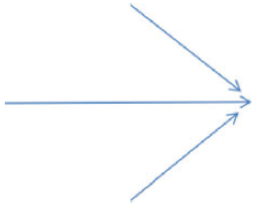
$$c_1(v) = \begin{cases} 1, & n(v) \leq 4 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

A given node v will be contracted only if $c_1(v)$ is equal to 1; where $c_1(v)$ is the criterion. When the initial criterion has been satisfied for a given node v, edge difference is then used as the next rule for inclusion or exclusion of that node for contraction.

$$c_2(u, v, w) = \begin{cases} 1, & w_{uv} - w_{vw} \leq \text{threshold} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Here, w_{uv} and w_{vw} are the weights of the links (u, v) and (v, w) respectively. In this work,

Table 3.1: Examples of contraction rules

Case	Some Cases of Node Removal	Results of Node Removal
1		
2		
3		
4		na
5		
6		

NOTE: na = not applicable.

the weight is the estimated speed for each link. In our work, we set the threshold to 0, so that only if the links have the same speed, nodes can be contracted.

Based on these two criteria, we will now discuss the rulesets for node contraction. Given the nodes that have been ranked, the steps for coarsening are as follows:

1. Set the node with the lowest rank as v.
2. Check if $c_1(v)$ is satisfied.
3. If it is satisfied, we find all the incoming and outgoing links of node v. If $c_1(v)$ returns 0, go to step 9.
4. Pair up all the incoming and outgoing links while considering the direction. For example, in case (6) of Table 3.1, if (u,v), (x,v) and (y,v) are the incoming links and (v,w) the outgoing link, the pairs are (u,v,w), (x,v,w) and (y,v,w).
5. U-turn is not considered in this work, thus (u,v,u) is not considered as a valid pairing as shown in case (2) of Table 3.1.
6. Check if c_2 criteria is satisfied for all the valid incoming-outgoing pairs.

7. If one of the pairings does not satisfy the condition, assign the next node in the rank list as v and repeat from step 2.

8. If c_2 is satisfied for all pairs, then we remove the node from the node list and update the node ranking as new links are formed. Repeat from 1.

9. Stop the iteration.

In the case (4) of Table 3.1, the node cannot be contracted as the edge difference did not satisfy the threshold criteria. The node coarsening is applied to the mapped Amsterdam network with 7512 links. As explained before, the peak period was estimated to be around 4pm and hence the speed at this time slice is used for the network coarsening. The new coarsened network contains 411 links as shown in Figure 3.1(c).

Speed Estimation

This section described the methodology to estimate the link speed based on individual travel times recorded by the cameras. First, a cleansing process will remove outliers. Second, the speed will be estimated for each time period.

The moving average process has been used to remove travel time outliers. There are usually alternative paths for a given Origin-Destination (OD). It can be k-shortest paths, representing the common route choices. We defined travel times significantly higher than the distribution as outliers. To remove these outliers, we propose two approaches: (1) treat the travel times higher than the third quantile as outliers. In order to keep the dynamics, distributions can be split by periods. The disadvantage of this approach is that distribution is sensitive to the number of observations. Thus, outliers can be smoothed into time periods with few travel times. (2) The approach that we used for removing the outliers is based on the moving average process. We define the moving average $\tilde{\tau}$ as

$$\tilde{\tau_n} = \frac{1}{k}\sum_{i=0}^{k-1}\tau_{n-i} \tag{3.4}$$

where τ_n is the n^{th} realized travel time. Outliers are defined by $\tilde{\tau_n} + \Delta\tau$, where $\Delta\tau$ is the travel time window. In our study, we set $\Delta\tau$ to the standard deviation of the peak demand. $\Delta\tau$ has be tuned by a graphical inspection of the effects of $\Delta\tau$ on the number of outliers. We considered only the upstream window to remove travel time outliers as the downstream window is not relevant for our study, i.e. we do not consider that fast travel times could be outliers. The travel time window is refined after two iterations. Figure 3.2 gives an example with the x axis being the observed time and y axis the travel time. The red curve is the moving average and the black curve represents the upstream travel time window.

Individual travel times have been aggregated by period. We consider the mean of individual travel times for a given OD at a given period.

An OD denoted by od_i is characterized by a succession of links $L_i = (l_1, l_2, ..., l_n)$, i.e., represents a path. The first shortest path has been used to keep speed estimation zones compact. During a given period t, the travel time of od_i is $tt_{(i,t)}$. Then, it is easy to estimate the speed of the path i by $s_i = \frac{\text{dist}(L_i)}{tt_{i,t}}$. It can happen that a given link l_k belongs to different ODs paths, i.e. $l_k \in (od_1, od_2, ..., od_m)$. For this case, its speed is defined as the mean of the corresponding speeds of the paths.

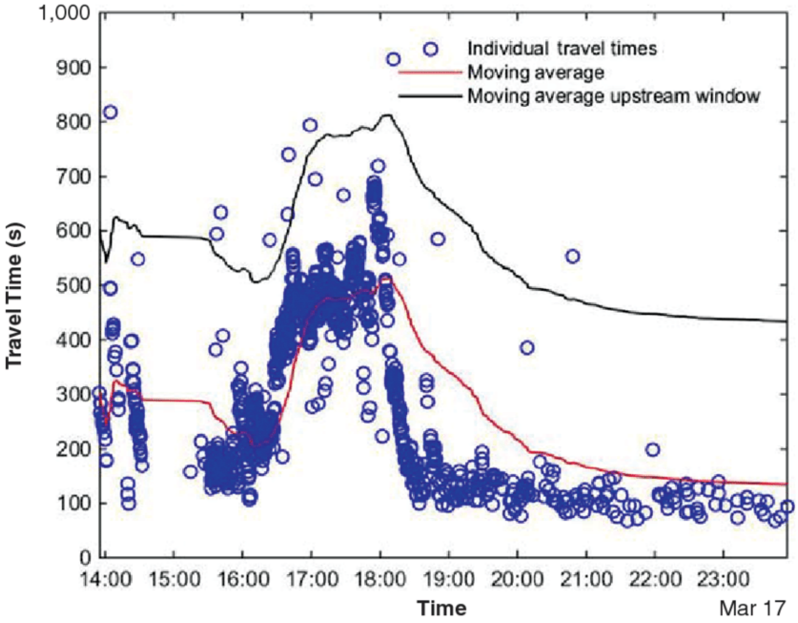


Figure 3.2: Travel times exceeding the travel time window (black curve) are considered as travel time outliers

When no data is available to measure $tt_{(i,t)}$, we distinguished three approaches to estimate the speed: (1) set links without data to the limited speed, which does not consider traffic lights. We assume that setting limited speed at links without data will increase the variance of the zones. (2) Compute new speed value through the speeds of a specific neighborhood. We assumed that the average speed of a given specific neighborhood will smooth the speed. Thus, congestion pockets can be less homogeneous and location zones can be less identifiable in space and time. The phenomenon is more common with link belonging to the boundaries of the zones, i.e. the link without data is being connected with links from different zones. (3) Duplicate speed from an identified link. We assumed that the duplication of speed minimize the speed variance of zones. We use the third approach by using a cost function. Let $G = (V, E)$ be the weighted graph where N_v and N_e are the numbers of vertices and edges respectively, A is the directed adjacency matrix of the network, C is the directed weight matrix and D is the cost paths matrix. A represents relationship of the finite graph G. In our study, G is the graph of the 3D network, i.e. a repetition of the same network at N_t time slices. Through the time, a vertex v_i is connected to itself at time steps $t+1$ and $t-1$ where $t \in (1, N_t)$. Its edge weight is set to the period duration. Through the space, we denote the edge weight by c_{ij} ,where i and j are connected links. Directions are considered to set the edge weight. For a given edge, if vertex j named v_j is downstream to v_i then the edge weight $c_{ij} = \frac{\text{length}(l_j)}{s_j}$, else if vertex v_j is upstream to v_i then the weight cost $c_{ij} = \frac{\text{length}(l_j)}{w}$, where w is the backward wave speed and it is set to 5 m/s in this study. If no data is available for v_j which is downstream to v_i, then $s_j = \max(s) + \frac{\max(s)}{2}$. Dijkstras algorithm [104] is used to estimate the shortest path between two vertices i and j in G The cost of the shortest path is denoted by D_{ij}. Thus, the link without data will be assigned the speed of the most relevant link. The most relevant link is identified as the link that minimizes the cost function. For computation efficiency, the process is performed strictly for links without data and a time window is used to constrain the search for the relevant link.

Experimental setup

The data preparation process - the coarsening methodology and the speed estimation of the link - considers a weighted directed network. The partitioning methods used in this chapter requires strongly connected graph, i.e. a directed path exists for every pair of vertices. A real network is strongly connected when a vehicle can reach any link from any starting point. For both fine and coarse resolutions networks, this constraint is not true. Thus, direction is not a convenient attribute to partition the Amsterdam network.

Our study application focuses on only one-day data. This given day is a common weekday. The analyzed period is from 7am to 5pm. It is a time window of 8 hours, one third of a day representing the morning peak demand. After analyzing the travel time data, it was found that not all of the ODs are used on all days. Only used ODs have been considered for reconstructing the network based on the shortest path finding in the network with 411 links. They are mapped to create a new coarsened network as explained in Section 3.1.2. From the used ODs, around 16% of data are missing.

3.1.3 Spatio-temporal Partitioning Techniques

Normalized Cut based on snakes similarities

One of the contributions of this chapter is to adapt the existing methodology of snakes for a spatio-temporal network, i.e. a repetition of the same network at numerous time slices. A snake [91] is composed of a sequence of links, which iteratively grows by adding adjacent link that are similar to itself. The connectivity is ensured by a link addition constraint which considers links strictly belonging to the neighborhood of the snake. Let S_i be the snake initialized by the link l_i here $S_{ik} \in S_i$ is the subset containing the first k elements. For a fixed time, the neighborhood of S_{ik} of a given snake is defined as the links spatially connected to it. We consider this neighborhood both in space and time. The link l_i at time t is denoted by $l_{(i,t)}$. We make duplicates of link $l_{(i,t)}$ at instants $t-1$ and $t+1$, denoted by $l_{(i,t+1)}$ and $l_{(i,t-1)}$ respectively. The snake similarities are defined as follows [91]

$$w_{ij} = \sum_{k=1}^{N} p^k \text{intersect}(S_{ik}, S_{jk}) \tag{3.5}$$

where the weight coefficient p is fixed to $p \leq 1$. Let W be the snakes similarities matrix with $W(i,j) = w_{ij}$. NCut [92] is a measure of dissociation based on this W. The complexity of NCut is NP-compete.

One of the main inconveniences of snake is its heavy computational cost. The complexity to run a snake is $O(n)$ for the best case and $O(n^2)$ for the worst case, where $n = N_l$ and $n = N_l * N_t$ for a spatial snake and a spatio-temporal snake respectively. The complexity of running n snakes is $O(n^2)$ for the best case and $O(n^3)$ for the worst case. The growing snake algorithm search its neighbors iteratively. The neighborhood size depends on the growing snake pattern and the network topology. In particular, the neighborhood of a snake growing in a corridor is equal to 2 (the upstream and the downstream neighbors of the current snake). It is much smaller than the neighborhood of a snake growing in a strongly connected network, i.e. a network where every nodes have a link with every other nodes. These both examples correspond to the best and the worst cases. Their associated complexity are respectively $O(n^2)$ and $O(n^3)$. Any realistic situation is between these boundaries. For example, we empirically investigated the complexity of the algorithm in our case study. We show that it is $O(n^{\frac{22}{10}})$. It can be seen that the similarities decrease exponentially with the weight coefficient p. A sensitivity analysis was done to investigate the performance of snakes for different lengths. Results show that the quality of NCut partitioning is independent of the snake length. Nevertheless, the snake length - k - has to be set at a minimal threshold to keep the connectivity. Thus, a too short snake cannot discover the entire topology of the network in both space and time. The short snake length may also provide clusters results where a cluster contains links that are not all connected with each other. Therefore, we set the snake length to 38% of the spatio-temporal size of the network. In addition, the weight coefficient is set to $p = 0.8$ in our study.

Partitioning based on Data Points Clustering

The two other classic clustering methods that are considered for this study are: (i) GNG [105] and (ii) DBSCAN [106]. We represented the 3D network into a data set containing four variables, link coordinates with their corresponding speed and time measurement (x,y,t,s). The four quantitative variables have been normalized. After normalization, we multiply the speed column by a fixed coefficient equal to 3 to be sure that speed is the predominant variable over spatial and temporal coordinates during clustering.

(i) DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based method. DBSCAN has two user-specified parameters. The radius parameter $\varepsilon > 0$ specifies the radius neighborhood and the MinPts parameter specifies the density threshold of dense regions. For our study, parameters have been set to $\varepsilon = 0.01$ and $MinPts = 50$.

(ii) GNG is an Artificial Neural Network variant of Neural Gas [107]. GNG begins with two neurons and the network grows during the execution of the algorithm. GNG has been adapted for clustering through a two-step process: running GNG and reconstructing data point clusters based on GNG centroids. The user-specified parameters are the number of centroids N, the maximum number of iteration m,L, the adaptation threshold ε_b, ε_n, the neighborhood size α, δ, the time T, which have been set as $N = 10$, $m = 20$, $L = 50$, $\varepsilon_b = 0.2$, $\varepsilon_n = 0.005$, $\alpha = 0.5$, $\delta = 0.995$, $T = 50$.

Post-treatment

The clustering results provide clusters which are not connected as shown in Figure 3.3(a). The homogeneous zone partition needs to be a single connected cluster. Therefore, post-treatment is needed on the clusters that is obtained from DBSCAN and GNG to obtained connected clusters with minimum inter-cluster speed variance. There are three steps for the post-treatment algorithm and these are:

- Identifying the CCs in each of the cluster
- Assigning the biggest CCs as the initial clusters

- Assigning all the other CCs to the initial clusters

Given that there are N number of clusters from the data point clustering methods, there might be more than one CC for each cluster as shown in Figure 3.3(a). Ideally each cluster should contain a single CC. In order to achieve this, all CCs within each cluster are identified. Then, these CCs are sorted according to the CC size. Given that the target cluster size is M, the biggest M CCs from different clusters are chosen as the initial cluster. This process is illustrated with a simple example in Figure 3.3. In Figure 3.3(a), there are 2 CCs in blue cluster, 1 CC in red, 1 CC in green and so on. It can be clearly seen that there are 2 CCs for the blue cluster.

Assuming that there are a total of X CCs from all the clusters, there are $(X-M)$ clusters that still need to be assigned to one of the initial cluster. The rest of the $(X-M)$ CCs are merged with the M initial cluster and the following two parameters are found for each pair.

$$c(x,m) = \begin{cases} 1, & x \cap \text{m is connected} \\ 0, & x \cap \text{m is not connected} \end{cases} \tag{3.6}$$

$$v(x,m) = \begin{cases} \text{variance}(s_x, s_m), & x \cap \text{m is connected} \\ \infty, & x \cap \text{m is not connected} \end{cases} \tag{3.7}$$

where $x \in (X-M)$ CCs, $m \in M$ initial CCs, s_x and s_m are the speed of each cluster. Once the $c(x,m)$ and $v(x,m)$ have been calculated for all the CC pairs, the $(X-M)$ CCs are sorted in decreasing order according to the $\sum c(x,m)$ for all $m \in M$ CCs. The x CCs with the largest $\sum c(x,m)$ is selected and it is merged with the initial cluster m that has $c(x,m) = 1$ and have the smallest variance $v(x,m)$ among all the $m \in M$. Once merged, this cluster is removed from the $(X-M)$ CCs and $c(x,m)$ and $v(x,m)$ is updated for all $(X-M)$ CCs as the initial clusters are updated. This process is repeated until all the $(X-M)$ CCs are assigned to the M initial clusters.

Figure 3.3 shows an example of post-treatment results in 2D and 3D. Figure 3.3(a) shows the cluster before post-treatment and Figure 3.3(b) shows post-treatment with the same number of clusters as the input for a single time slice. Figure 3.3(d) shows the result of post-treatment in 3D with same number of clusters as the input which is shown in Figure 3.3(c). There is no difference in post-treatment methodology between 2D and 3D. The only difference is in calculating the 3D adjacency matrix for finding the 3D CCs and the connectivity. The 3D adjacency matrix is defined by creating bi-directional links between the time slices.

Evaluation Metric

The three clustering techniques have been compared using three indicators in this study: (i) Total Variance normalized (TNn), (ii) Connected Clusters Dissimilarity (CCD), and (iii) time computation. (i) TV is the original indicator of the snake similarities partitioning and is defined as $TV = \sum\limits_{A \in C} N_A * \text{var}(A)$ [90]. TV has been normalized using the following equation:

$$TVn = \frac{1}{N} \frac{\sum_{A \in C} N_A * \text{var}(A)}{S^2} \tag{3.8}$$

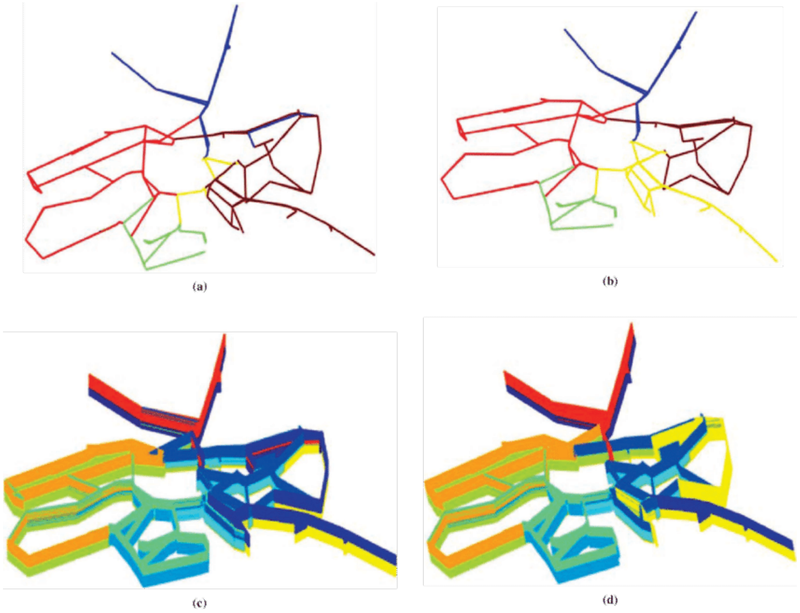


Figure 3.3: Post-treatment results for data point clustering methods (a) Unconnected 2D clusters before post-treatment (b) Connected 2D clusters after post-treatment (c) Unconnected 3D clusters before post-treatment (d) Connected 3D clusters after post-treatment.

This indicator is based on the assumption that a given cluster is composed of links characterized by similar speeds. The speed variance is highlighted. (ii) The second metric used is the CCD. The criterion is the dissimilarity between a given cluster and its neighboring cluster, i.e. clusters touching the given cluster. CCD is defined as follows:

$$\mathrm{CCD} = \frac{\sum\limits_{i=1}^{n} \sum\limits_{k=1+i}^{n} \delta_{ik} |\bar{x_i} - \bar{x_k}|}{\sum\limits_{i=1}^{n} \sum\limits_{k=1+i}^{n} \delta_{ik}} \tag{3.9}$$

$$\delta_{ik} \begin{cases} 1 & \text{if k and i are connected clusters} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

(iii) The time computation indicator evaluates the computational cost of the algorithms. The complexity has to be considered for another size of network and number of time slices. Two different field of partitioning methods have been compared: (1) NCut from the graph theory based on the snakes similarities and (2) DBSCAN and GNG from data point clustering. The basic data point clustering methods are faster but a post-treatment process is required. The computational cost of the post-treatment is heavy because it checks the connectivity of the previous results and iteratively updates the clusters. The time computation evaluation includes both field partitioning methods and all the internal processes.

3.1.4 Results and Discussion

In this section, results from the three methods are analyzed and compared - NCut based on snakes similarities, DBSCAN with post-treatment and GNG with post-treatment. The comparison focuses on two conceptually different fields to partition a transportation network. Figure 3.4 illustrates the partitioning results from three methods under a fixed number of clusters set to 2. Both the data point clustering methods do not produce the same spatiotemporal zones by the post-treatment algorithm. The zones shapes present a reasonable 3D covers, i.e., a given zone is roughly compacting by space and time. We use three indicators to evaluate the three methods. TVn and CCD measure the quality of clusters representing the homogeneous zones and the compactness respectively. The time computation quantifies the computational cost applied to our case study. Figure 3.4(d,e) shows the TVn and CCD for a systematic number of clusters from 2 to 9. It can be observed that GNG is the best method that minimizes the TVn. NCut is the best method that maximizes the CCD. However, the time computation for GNG is found to be the fastest.

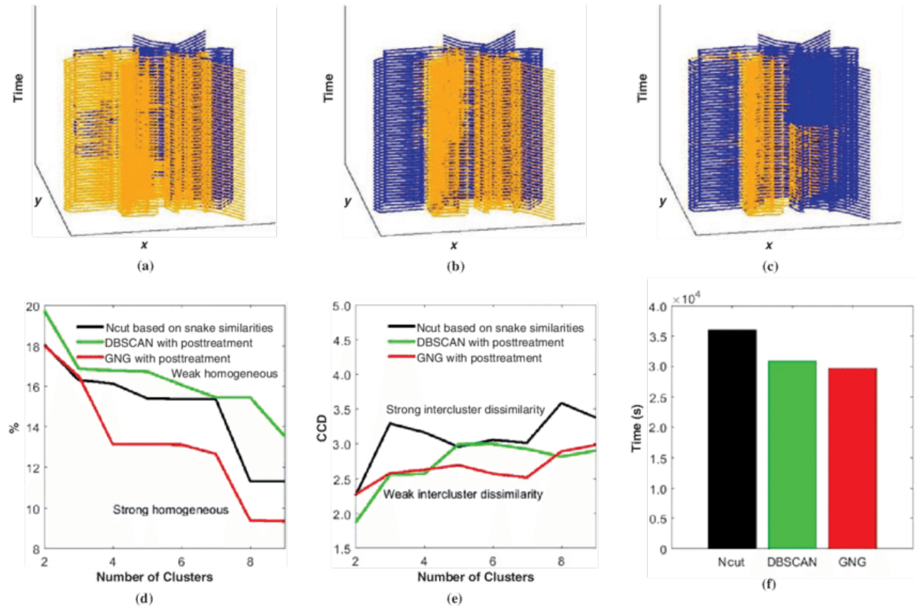
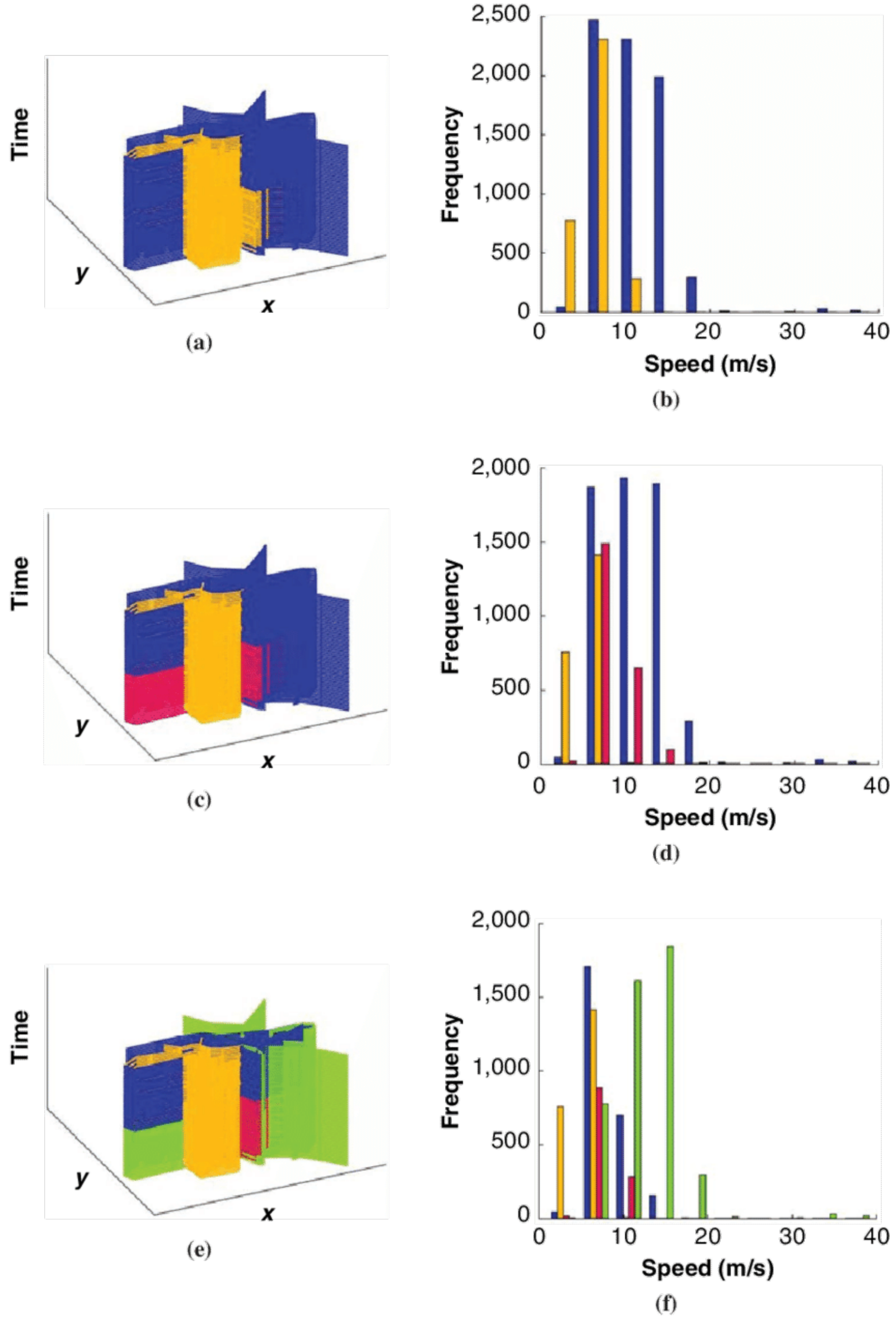


Figure 3.4: Amsterdam network (7am to 5pm) where 3D clusters $(n = 2)$ obtained with (a) NCut based on snakes similarities, (b) DBSCAN with post-treatment, (c) GNG with post-treatment; and a comparison of the three partitioning methods by (d) TVn, (e) CCD and (f) the computation time

Figure 3.5(a,c,e) illustrates the GNG partitioning with different number of clusters ranging from two to four. Figure 3.5(b,d,f) are histogram plots of links speed for each cluster. Note that the maximal speed is around 40 m/s, corresponding to highway. The validated Amsterdam network used in this work contains both provincial roads and highways. The histogram validates that each cluster has speed variance as low as possible. For example, in Figure 3.5(d), most of the links in the blue cluster have speed of 5 to 10m/s. Only a few of



(a) (b) (c) (d) (e) (f)

Figure 3.5: (a,c,e) 3D network visualization of the GNG partitioning into two to four spatio-temporal zones and (b,d,f) its corresponding histograms

the links in the blue cluster have 0 to 5m/s compared to the orange cluster. This observation remains valid for all the three cases shown here proving the hypothesis that the cluster results from post-treatment provides clusters that are connected and that minimizes the speed variance.

3.1.5 Conclusion and Future Work

This research describes a generic methodology to prepare data for transportation application. We reconstructed the network from GIS based on the coordinates of cameras and their corresponding recorded travel times. We also introduced coarsening techniques to reduce computational complexity. The speed has been estimated from incomplete and flawed individual travel times. The validated network along with the estimated speed has been used for partitioning.

Two different concepts of spatio-temporal partitioning of a transportation network has been compared in this work. We implemented methods belonging to different fields: (i) unsupervised learning and (ii) graph theory. (i) Clustering approach considers links under a network as data points. This hypothesis allows us to implement simple and efficient clustering algorithms but it requires post-processing for contiguity, i.e. all clusters should be composed of a unique CC. (ii) We considered transportation network at numerous time slices as a graph. In this case, connectivity is considered through graph topology.

This chapter has presented three methodologies to partition a network both in space and time, which is demonstrated in a coarsened Amsterdam city network. The partition criterion is the speed. The comparison between the three techniques has been evaluated by two metrics: TVn and CCD. Preliminary results show that none of the partitioning method is better w.r.t both metrics. TVn measures the homogeneous zones which can be spread throughout the 3D network keeping the connectivity. CCD indicator focuses on the compactness of the zones but can forsake the homogeneity. The choice of a spatio-temporal partitioning method is a compromise among both criterions.

There are various future directions that can be pursued. The methodology can be improved to be more generic and computationally efficient. A more extended comparison study needs to be implemented. Also, we have only looked at one day in this work and the methodology can be iterated for several days. Another application that is promising for creating these spatio-temporal speed regions is that it can be used to derive future travel times. These claims still need to be researched and validated.

3.2 Revealing the day-to-day regularity of congestion patterns

3.2.1 Introduction

Studying human mobility in large cities is critical for multiple applications from transportation engineering to urban planning and economic forecasting. In recent years, the availability of new data sources, e.g. mobile-phone records and global-positioning-system data, has generated new empirically driven insights on this topic. A central question at large spatial and temporal scales is which (dynamic) components of human mobility can be considered as predictable and thus suitable for explanatory and predictively valid mathematical models, and which part is unpredictable. Earlier studies of human trips shows that traveled distance can be described by random walks and more precisely as Lvy-flights[108]. Latter studies partly amend this theory by recognizing some regularity features in peoples' trips. Individuals obviously frequently move between specific locations, such as home or work[109]. Such patterns are also regular in time[110, 111] meaning that the most frequent locations are likely to be correlated with daily hours and dates. Regularity can also come from decomposition by transportation modes[112]. Human mobility can be studied at the microscopic level, i.e. through person trajectories, but also at the macroscopic level, for example by estimating commuting flows between different regions (origins to destinations) or on the different links of a transportation network[113, 114]. Such collective mobility patterns can be explained for example by distances between regions[115, 116], trip purposes[117] and road attractiveness related to road types, e.g. freeways, or locations, e.g. in major business districts[118]. Predicting commuting flows often requires local data for calibration[119] meaning that results cannot easily be transferable to other regions or cities. Recent findings[120], however, show that a scale-free approach corresponding to an extension of the radiation model can successfully be applied to commuting flow estimation. This means that some regular patterns can be observed also at the macroscopic level.

In this chapter, we aim to pursue the investigation of regularity in macroscopic mobility patterns not by focusing on the commuting flow distributions; but on the resulting level of service of the transportation (road) network, i.e. on congestion patterns. Along with commuting flows, congestion patterns vary both within days and from day-to-day at large urban scales. It is common knowledge that some regularity happens as congestion is usually observed during peak hours on the most critical links of the network. In contrast to commuting flows, congestion patterns are more easily observed using real data as they only require speed information in the different network links. Nowadays, such information is easily accessible through different sensing technologies that are massively deployed in many cities. However, in large networks with speed data on hundreds (or thousands) of links over a large number of time periods, studying regularity and identifying distinct network congestion patterns is not an easy task to undertake: the challenge is to see the forest (regular large-scale traffic patterns) for the trees (many local pockets of queuing and congestion spillback processes). Here, we propose a new concept to address this challenge. We synthesize within days link speed data and simplify day-to-day comparisons, by means of so-called spatio-temporal speed cluster maps. Such 3D speed maps consist of a joined partition of space (road network links) and time (the different observations) into homogeneous clusters characterized by a constant mean speed. More precisely, such a partitioning should fulfill the following criteria: (i) all clusters should contain a single connected graph component meaning that all links are reachable within a cluster, (ii) the internal speed variance for all clusters should be minimized - the *intra-cluster homogeneity criterion* and (iii) the difference in speed between neighboring clusters should be maximized - the *inter-cluster dissimilarity criterion*.

Clustering is a common problem in different fields of engineering such as data mining[121] or image segmentation[122]. Two recent and significant contributions in transportation for our work are (i) the application of the k-means algorithm[123] to partition urban networks by considering spatial locations of the road as new features in the data and (ii) the definition of a similarity matrix between observations and the application of the Ncut algorithm[124]. These works result in 2D clusters, covering a spatial portion of transportation networks for a given time period. To obtain a picture of the traffic dynamics over different time periods, the algorithms are simply iterated for each time period without connecting the 2D clusters. Note that usual clustering works in transportation also include compactness as a requirement for clusters. This is because the main application is perimeter control. In this chapter we present an algorithm that directly unravels traffic dynamics over both space and time. We favor connectivity - requirement (i) - rather than compactness for clusters, which makes more sense in 3D. To this end, we first determine which clustering method is the most efficient to cluster all time-dependent link speed observations into 3D speed maps, where we consider the intra-cluster homogeneity and inter-cluster dissimilarity criteria as well as the computational times to determine the optimal number of clusters. Second, we apply consensus learning techniques [125, 126] to summarize multiple 3D speed maps from a training set of days, into a single common pattern. Interestingly, such a meta-partitioning operation can be fulfilled with a very small number of groups. This means that the day-to-day regularity of daily congestion patterns can be easily revealed based on such a classification. Finally, we will show that using a single consensus pattern for each class of 3D congestion maps is sufficient to accurately estimate *in real-time* travel times in the city. This means that addressing congestion patterns directly at the whole city scale for all time intervals reveals a meaningful and accurate global picture of the city traffic dynamics that can be used as an efficient alternative to classical methods that process much more data at local and short-term scales.

3.2.2 Results

Our case study corresponds to most of the major street network of Amsterdam city excluding the freeways, see figure 3.6(a). Whereas the original mapping of the inner city network contains over 7512 links, it is coarsened in this chapter to 208 links and 214 nodes. Such an operation basically merges all successive links in the same direction between two intersections into a single one and disregards the internal links in the original mapping for intersections, see the method section. Mean speed information is available every 10min between 7am and 3pm for all 208 links during 35 days. This information is derived from license plate recognition systems at different critical points of the network. The methodology to derive link speed data from passing times, coarsen the network, and reconstruct missing data has already been published[127]. It should be noticed that all the methods elaborated in this chapter can be applied to any set of time-dependent link speed data combined with the related connected graph (contiguous time intervals for the same network link should be connected by an edge) whatever the initial sensing method is.

Clustering results for individual days

So, the initial data for a particular day is an undirected graph in which links are connected in space with their upstream and downstream neighbors following the road network, and in time by their immediate neighbors, i.e. the previous and the next time intervals for a given link. Each link is characterized by a spatial (x,y) position, a time and a speed value. Link directions are not considered during the clustering process because changes in traffic volume propagate forward while congestion propagates backward and we want to capture both phenomena. To obtain the 3D speed map related to such data, we first benchmark different clustering algorithms from the literature. We choose to oppose the most recent development in clustering for transportation networks, i.e. the Ncut algorithm with snake similarity also referred to as S-Ncut[124] (see supplementary S1 for more details) with two simpler clustering algorithms, the k-means [128] and DBSCAN [129] algorithms, see the method section. The main difference between these, is that S-Ncut uses network topology when calculating the similarities between observations; whereas the two other methods simply use normalized Euclidean distances (regardless of topology) to balance both space, time and speed values. Note we weigh speed three times more heavily ($\alpha = 3$) compared to space and time (vicinity) since our objective is to obtain clusters with a narrow speed distribution, see supplementary S2 for more rationales about the choice of α. The quality of the clustering results is assessed for a given number of clusters n through two indicators that relate to the *intra-cluster homogeneity* and the *inter-cluster dissimilarity* criteria respectively: the total within cluster variance (TV_n) and the connected cluster dissimilarity (CCD_n).

$$TV_n = \frac{1}{\sum_{i=1}^{n} n_i} \frac{\sum_{i=1}^{n} n_i s_i^2}{s^2} \;; \quad CCD_n = \frac{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \delta_{ik} \sqrt{n_i n_k}\, |\bar{x}_i - \bar{x}_k|}{\sum_{i=1}^{n} \sum_{k=1+i}^{n} \delta_{ik} \sqrt{n_i n_k}} \tag{3.11}$$

where n_i is the number of links in cluster i, $\bar{x}_i$ and s_i are respectively the mean and the standard deviation of link speeds for cluster i, δ_{ik} is equal to 1 only if clusters i and k have a common border and s is the standard deviation of link speeds for the whole network. Since we also impose that each cluster should contain a single connected graph component, clustering results should be post-processed, see supplementary S3. Note that S-Ncut results, even though the method includes topological considerations to calculate similarity between observations, also require post-processing, see supplementary S1. Post-processing has very little impacts on TV_n and CCD_n values for S-Ncut. It deteriorates TV_n values and to a lesser extent also CCD_n values for DBSCAN and k-means methods, see supplementary S3. This is not surprising as these two methods only account for proximity (distance between links) and not for connectivity within a cluster. In the end, what is important to assess the quality of a method is to compare TV_n and CCD_n values after post-processing when we are sure that *connectivity* - requirement (i) - is verified.

Clustering results after post-processing are presented for a randomly selected day among the 35 available in figures 3.6(b-f). The evolutions of TV_n in figure 3.6(b) and CCD_n in figure 3.6(c) are comparable for all three methods, although k-means can be identified as the best method to minimize TV_n, and DBSCAN appears slightly more efficient in maximizing CCD_n. DBSCAN also appears to provide more stable (i.e. monotonically decreasing) results for increasing cluster numbers than the other two. However, the TV_n and CCD_n values are not sufficiently different to provide conclusive evidence that one method is better than the other two. What can be concluded is that the S-Ncut algorithm has much higher computational times than the other two, which disqualifies the method since clustering has to be repeated for multiple different days. Both k-means and DBCAN are over 20 times faster than S-Ncut on the same computer, see figure 3.6(f). Finally, figures 3.6(b-c) highlight that improvements to TV_n and CCD_n values tend to significantly reduce when the number of cluster exceeds 9 to 10. This means that for this particular day, the optimal number of clusters can be fixed to 9. The resulting 3D speed map is presented in figure 3.6(d). A 3D video is also visible on the data repository website, see additional information. In figure 3.6(e) a slice at time *t*=9am is shown to illustrate the clustering results in detail. Note that links from the same cluster may look not connected because of the slicing but they are of course connected through time links and different time periods.

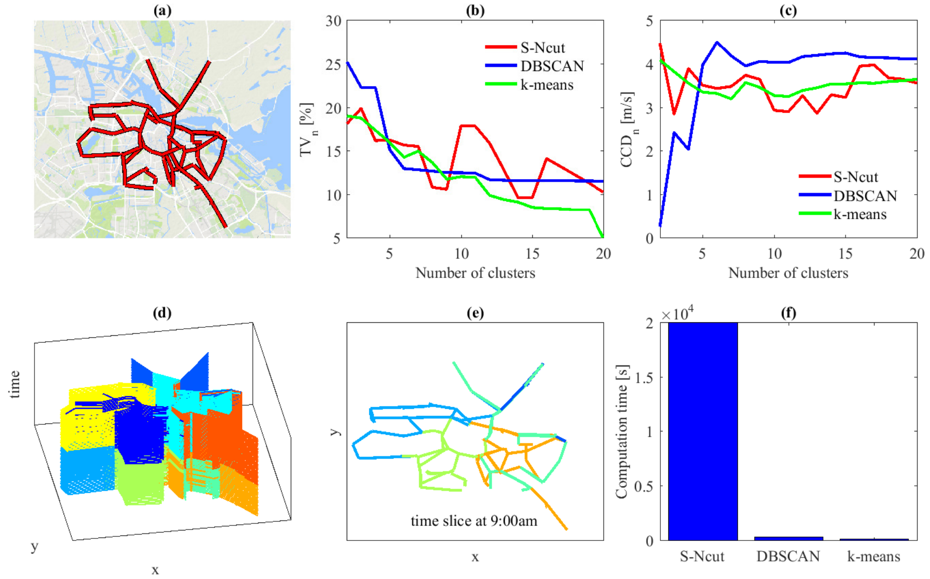


Figure 3.6: Link speed 3D clustering for one particular day. (a) Sketch of Investigated network - Amsterdam city (NL) - MapData @2017 Google (b-c) Evolution of the total variance (TV_n) and the connected cluster dissimilarity (CCD_n) with respect to the number of cluster for different clustering methods (d) Resulting 3D speed maps for 9 clusters (e) Slide of the 3D speed map for time period t=9am (f) Computational times for different clustering methods and a targeted number of clusters equal to 9. Graphs (b,c,f) show that the clustering algorithms that do not consider the graph topology, i.e. the k-mean and the DBSCAN, blast the S-Ncut in terms of computational times with analogous TV_n and CCD_n results. DBSCAN appears very stable when the number of cluster exceeds 6. Selecting 9 clusters looks optimal for this dataset and network configuration.

Figure 3.7 now presents the clustering results for all 35 days. Figure 3.7(a) shows that S-Ncut and k-means generally outperform the DBSCAN method with lower TV_n values.

The score on CCD_n values is much less decisive. However, when reducing the number of clusters to 9, and testing all methods with this same number of clusters, k-means clearly outperforms the other methods over all 35 days. Interestingly, when comparing figure 3.7(b) to figure 3.7(a), one can observe that for this relatively low number of clusters (9), using k-means results in TV_n values that are very close to the best results obtained with any of the other two methods for larger number of clusters. Figures 3.7(c) and (d) provide a direct comparison of the three methods with respect to minimizing TV_n and maximizing CCD_n for $n = 9$. The k-means method generates a distribution of TV_n values for all days that is significantly better (lower) than both other methods. The distribution of CCD_n with k-means is not the best (the highest), but it is very close to what is obtained with the best methods for this indicator, i.e. the S-Ncut see Figure 3.7(d). Since k-means is the most economical method in terms of computational cost, we can conclude that it must be favored to obtain 3D speed maps in this case. Furthermore, the results provide evidence to fix the optimal number of 3D clusters to 9 for our case study.

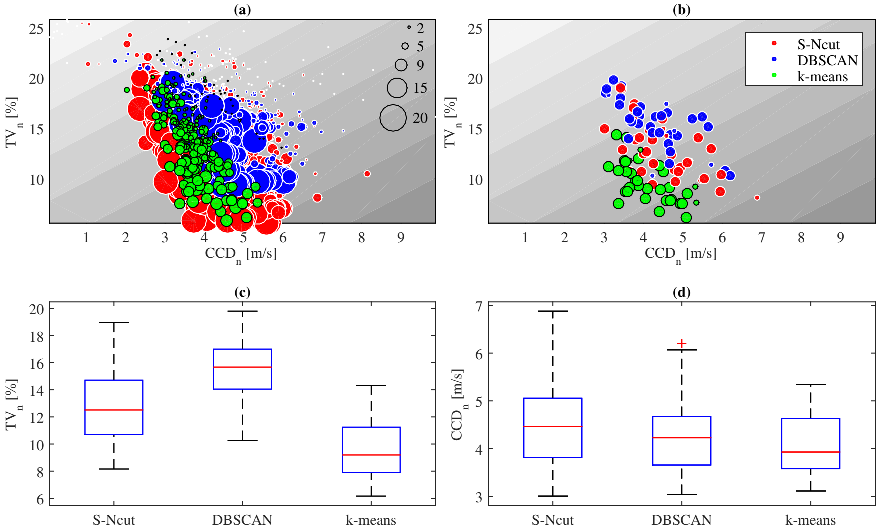


Figure 3.7: Clustering results for all 35 days. (a) Clustering efficiency with respect to the number of clusters (b) Clustering efficiency for a number of clusters equal to 9 (c-d) TV_n and CCD_n values (respectively) for all days and 9 clusters. (a) shows that S-Ncut provides the best results compared to the other two methods when the number of cluster is large (above 15). However, when the number of clusters is reduced to 9 (b), k-means provides in general the lowest TV_n values while leading to similar CCD_n values than S-Ncut and DBSCAN. This is confirmed by (c) and (d) that show TV_n and CCD_n distribution for all methods when the number of cluster is 9. More interestingly, by comparing (b) and (a), it appears that k-means with only 9 clusters usually lead to close results compared to S-Ncut with a significant higher number of clusters. So, we define as 9 the optimal number of clusters for all days.

Classification of multiple days to identify consensual congestion patterns

Now our objective is to find commonalities in the 35 daily congestion patterns, and, ideally, summarize these with a fewer number of ”consensual” patterns. To this end, we first have to define a common link network for all the 35 days, see supplementary S4. This is necessary because some links may have insufficient observations on particular days to be assigned with a significant value. The procedure has 3 main steps as outlined in figure 3.8(a). In step 1 we obtain 3D speed maps related to each daily pattern, by running the k-means algorithm with 9 targeted clusters over all the 35 days of the dataset. After this, each observation, i.e. a couple composed by a link and a time period, is assigned a cluster ID i. Each day k can then be synthesized into a single ordered vector of all observations π_k, whose values are the cluster ID. To compare two different days π_k and π_l and assess if their 3D speed maps have similar shapes, we use the normalized mutual information (NMI) indicator. It has been designed to assess the proximity between two clustering results[125, 130].

$$NMI(\pi_k,\pi_l) = \frac{I(\pi_k,\pi_l)}{\sqrt{H(\pi_k)H(\pi_l)}} = \frac{H(\pi_k)+H(\pi_l)-H(\pi_k,\pi_l)}{\sqrt{H(\pi_k)H(\pi_l)}} \tag{3.12}$$

where $I(\pi_k,\pi_l)$ is the mutual information between π_k and π_l, which measures the mutual dependence between two random variables [125], $H(\pi_k)$ is the entropy of π_k and $H(\pi_k,\pi_l)$ is the joint entropy of π_k and π_l. Calculating the NMI for all day couples allows us to define a similarity matrix. We can then classify the whole set of days using the Ncut algorithm [122], see step 2 in figure 3.8(a). More specifically, we apply a classical cross-validation approach by randomly splitting our 35 days into a training set of 28 days and a validation set of 7 days and considering 12 replications in total. The purpose of the validation set will be explained later. We test a partition of the 28 training days into 2 and 4 groups for all replications of the training set. It appears in all cases that 4 groups lead to better results, see supplementary S5. All four groups appear homogeneous with high mean NMI values inside a same group (usually higher than 0.6) and low differences between the maximum and the minimum NMI values (usually below 0.24). When looking at the day labels (Monday, ...) within the four groups, no clear pattern appears. The major conclusion at this stage is that the 28 days can be classified into only 4 groups, which exhibits close 3D speed map shapes. We are now going to adjust the cluster shapes of the days belonging to the same group to obtain a unique consensual shape that can be applied within the group.

The consensus clustering problem consists in identifying the most representative partition from a group of partitions [125, 131]. The best of K (BOK) algorithm [126] can be used to determine the median partition m (the 3D speed map shape of a single day in our case) that maximizes the total similarity TS with all the other days belonging to the same group:

$$TS = \sum_{k=1}^{a} NMI(\pi_m,\pi_k) \tag{3.13}$$

where a is the number of days in the targeted group, π_m is the vector resulting from the initial clustering (3D speed map) for the median partition, and π_k the same vector but for each other day of the same group. The median partition can be further improved (increasing TS) by moving some of its elements from one cluster to another, i.e. changing the cluster ID of some elements in the vector. To realize such an optimization, we apply the one element move (OEM) algorithm [126]. It consists in randomly changing the label of one element of the vector and assess if such a change improves the TS value. The algorithm stops when TS has not been improved for a while. Determining the consensus shaping for all 4 groups corresponds to the final step 3 of the data processing, see figure 3.8(a). Figures 3.8(b) and (c) illustrate the difference between the original cluster shape of a particular day and the consensual shape resulting from the processing of all days in the same group. Figures 3.8(d) and (e) respectively show the variations of the TV_n and CCD_n values when comparing the consensus cluster shape with the original one for all the training days and all replications. It appears that the TV_n values significantly deteriorate (increase by more than 2%) for only 15% of the days while the CCD_n are significantly worse (decrease by more than -0.5 m/s) for only 20.8% of the days. Even for the days that see a significant change in the clustering quality, the final values related to the consensual shape remain always acceptable. This means that the consensual shape is relevant to describe in a unique and common manner the congestion patterns of the same group of days. Since classification into 4 groups appears sufficient, the conclusion is that the 3D speed maps of the 28 training days can be synthesized into only 4 different consensual congestion patterns. For now, the groups and the consensual shapes have been defined based on the initial cluster shapes without using the link speed information. The remaining question is to assess whether the consensual shapes are also relevant to define homogeneous regions in speed for each group of days. Because the consensual shape is the same within a group, it is easy to calculate the mean speed for each of the 9 cluster IDs and each day. Figure 3.8(f) shows the distribution of the standard deviation of such a mean cluster speed among all days belonging to the same group. Such a calculation has been performed for all replications of the training set. It turns out that 37% of the standard deviation values are below 0.5 m/s and the vast majority (85%) is below 1 m/s. This means that the mean cluster speeds are very close for the same cluster ID among the days of the same group.

This is a major result because it implies that the consensual shape is also relevant to summarize the speed profile observed in the network over time for a same group of days. For a given group, we can associate to each consensual cluster ID the mean of the mean cluster speeds for each day and so, obtain a single 3D speed map that defines the congestion pattern of this group. In other words, all days of the same group can be synthesized into no more than 9 cluster shapes and 9 mean speed values. For our case study (the Amsterdam network), 4 consensual 3D speed maps look sufficient to capture the functioning of the entire work network over the 35 days and to get a full overview of the dynamic traffic conditions within the major road network of the city. This is strong evidence for a high degree of regularity and predictability of macroscopic traffic conditions in this network.

Application to real-time travel time prediction

We are now going to take advantage of the above major result to propose a fresh new look on a classical and popular problem in transportation systems, i.e. travel time prediction. This problem has been extensively investigated in the transportation literature using both (simulation) model-based and data-driven approaches as shown by recent review papers[132, 133]. Model-based approaches use network traffic flow models in conjunction with data assimilation techniques such as recursive Bayesian estimators to predict the traffic state and the resulting travel times in networks[134–136]. Data-driven approaches use general purpose

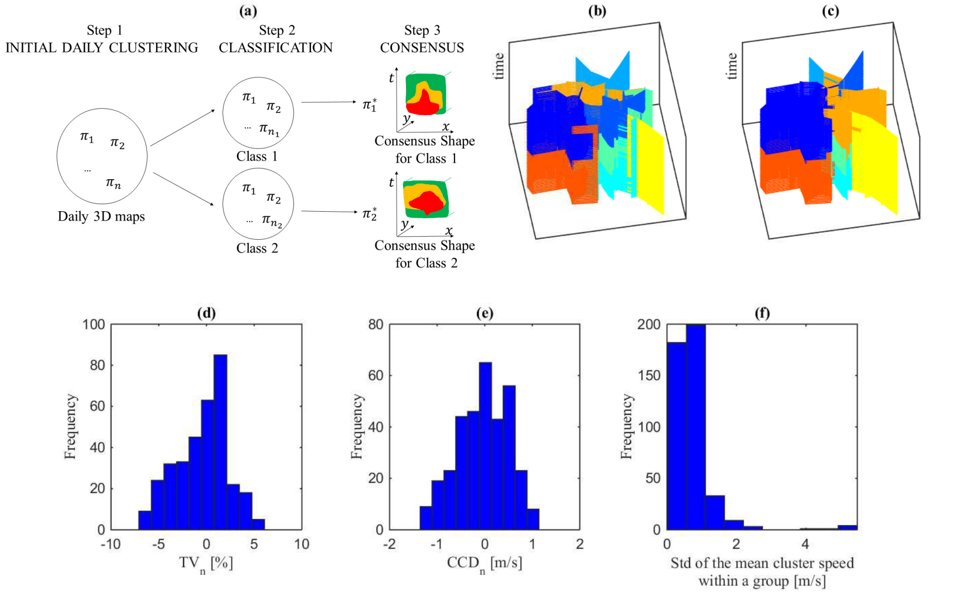


Figure 3.8: Classification of multiple days and congestion patterns identification for training sets. (a) The three steps to obtain consensual 3D speed maps (b) Original clustering for a particular day (c) Consensus clustering for the same day (d) Variation of TV_n between the original and the consensual cluster shapes for all days and all replications of the training set (e) Variation of CCD_n between the original and the consensual cluster shapes for all days and all replications of the training set (f) Distribution of the standard deviation of the mean cluster speed within a group of days (one value per cluster ID, group and replication). (d) and (e) show that in most case switching from the original to the consensus shapes for a day has minor to acceptable impacts on the TV_n and CCD_n values. This means that the consensus shapes can be considered as a good proxy for the clustering of each day. (f) shows that the consensus shape is also relevant to identify homogeneous regions in speed within a group as the standard deviation of the mean cluster speed remains below 0.5 m/s for the vast majority of cases.

parameterized mathematical models such as (generalized) linear regression[137, 138]; kriging [139]; support vector regression [140]; random forest[141]; Bayesian networks[142]; artificial neural networks, e.g. dynamic [143, 144] and (increasingly often) deep learning architectures [145, 146]; and many other techniques to capture (learn) from data the correlations between traffic variables (speed, travel time) over space and time. When reviewing the literature, there are many more approaches reported for estimation and prediction on freeway corridors, than for mixed or urban networks, which we hypothesize is due to two reasons. First, until recently, insufficient data sources were available for such large-scale urban prediction models. Additionally, and more tentatively, the urban prediction problem is a more complex problem to address than the freeway prediction problem because there are many more degrees of freedom that govern the underlying local traffic dynamics (e.g.

intersection control, crossing flows, high-frequency queuing also under free flowing conditions, much more route alternatives, etc), and thereby also the dynamics of speed and travel time. Recently, both model-based[136, 147] and more unified and systemic data-driven approaches[145, 148–150] have been proposed that, at least in principle, can be used to predict traffic variables in large-scale urban networks. However, when applied to large-scale networks, both model-based and data-driven approaches are indeed computationally complex, and methodologically cumbersome due to the high number inputs and parameters that continuously need to be calibrated and validated from data.

As an alternative, we propose a very simple and systemic approach that uses the consensual congestion patterns obtained in the previous section. First, let us define a number of probe trips that we will use for investigating the methods and the validation. Based on the network map, we define 10 trips that cover most of the network links, see figure 3.9(a). A virtual probe vehicle is launched every 10 min over the time period between 8am and 2pm and its travel time is calculated based on the time-dependent link speed information of the studied day. This defines for each day 370 probe trips characterized by the travel time that a vehicle would have experimented for this trip and this departure time. Note that travel time calculations are made on the directed version of the road network graph while the initial and consensual clustering were made without considering directions. First, we are going to investigate if the mean speed values related to the 3D congestion maps can be considered as a good proxy for the travel time calculation. For now, only the days included in the 12 different training sets are considered because their group label and thus their consensus clustering shapes are known. We define three methods to estimate the travel time depending all the options we have to define congestion maps:

- M1: initial cluster shape of the day + link speeds equal to the mean speed value of all links in each initial cluster and the same day;
- M2: consensus cluster shape of the group + link speeds equal to the mean speed value of all links in each consensus cluster and the same day;
- M3: consensus cluster shape of the group + link speeds equal to the mean speed value for all links in each consensus cluster over all days of the group.

Figure 3.9(b) shows the distribution (box plot) of the travel time estimation errors for all probe trips, all training days and the three methods. It appears that averaging the link speeds within each initial cluster (M1) obviously introduces errors in the travel time estimation: (i) the mean and median errors are respectively equal to -2.0% and -2.3%, and are thus close to 0 (ii) 50% of the probe trips (25th to 75th percentiles) have errors between -13.7% and 8.6% and (iii) 80% of the probe trips (10th to 90th percentiles) have errors between -22.1% and 17.6%. Interestingly, most of the errors come from the averaging process within the cluster: when switching to the consensus cluster shape (M2) or replacing mean cluster speeds of the day by the mean cluster speeds of the group of days (M3) leads to error distributions that are very close to what is observed for (M1). In particular, for M3, the mean and median error values are respectively -2.7% and -3.6%, 50% of the probe trips exhibit errors between -15.7% and 9.4% and 80% of the probe trips have errors between -23.9% and 19.2%. These results are fundamental because they first confirm from another perspective (here the travel time estimation) that consensus congestion maps with mean speed in each cluster determined over a similar group of days are very relevant to synthesize the traffic congestion pulse at the city level. We see no discrepancy when switching from M2 to M3 meaning that all days of a group have similar speed behavior within each consensus cluster. As no discrepancy is observed when switching from M1 to M2, the consensus shape appears to be a good proxy to partition all the days of the same group. Together, these two results demonstrate that the consensus cluster decompositions are relevant not only in terms of shape but also in terms of mean cluster speed values and provide a unique and systemic picture of what happen for all days belonging to a same group. Note that the same graph as figure 3.9(b) but with absolute travel time errors is presented in supplementary S6.

The previous analysis provides the rationale for a simple, systemic and real-time travel time prediction method for new days belonging to the validation sets. For a new day, M1 and M2 are no longer relevant because they require the data of this particular day. However, M3 still holds as long as the new day can be assigned to an existing group obtained through historical analysis, i.e. over the training set. The only missing component is a method to allocate in real-time the current observations of a new day to an existing group. Knowing the group, the predetermined consensus cluster shape and the related mean speed values for each cluster can be applied to predict the future travel times. Here, we propose a simple method with very low computational times to match a new day with an existing group. This method only requires the link speed information until the actual time t of the new day. First, we reduce the consensual cluster shape of each historical group (4 in our case) to the period of time between 7am and t. Then, we apply all restricted consensual cluster shapes both on the new day data and on the consensus map of the related group. Mean speed values for the same cluster i in the new day $x_{i,g}$ and the consensus $y_{i,g}$ are compared. The optimal group index g^* minimizes the Euclidean speed distance between the current day and the group:

$$g^* = \arg\min_{g} \left(\frac{1}{n} \sum_{i=1}^{n} (x_{i,g} - y_{i,g})^2 \right) \tag{3.14}$$

Note that the number of clusters n within the restricted time period (7am-t) can be lower than 9 in particular at the beginning of the day where all 3D patterns have not yet necessarily appeared. In practice, we can refresh the assignment of the new day to a group every hour starting at 8am, and assess the travel time predictions on the probe trips where a new virtual vehicle starts every 10 min. Figure 3.9(c) shows the results for a particular validation day and all trips starting at 9am. It appears that, even if the reference time period to assign the day to a group is short (here 7am-9am), the predicted travel times are close to the experimented one for all trips, i.e. all error values but one fall between -20% and 20%. Note that the travel times are simply calculated using the link speed values of the full day since we are not testing the application in real-time here. This means that we already know all the link speed information for the validation days on the contrary to a real-time implementation where the future is unknown. Figure 3.9(d) shows exactly the same results but now for all validation days (7 days and 12 replications meaning 84 in total) and all departure times between 8am and 2pm. Again, a large fraction of the travel time predictions (72.1% of the total probe trips) exhibit errors between -20% and 20% and almost all (91.9%) fall within an ±30% error margin. Despite its simplicity and its very low computational cost, the proposed method leads to accurate travel time predictions for most trips. This is confirmed by figure 3.9(e), which shows the cumulative distribution of all prediction errors. The mean and median values are equal to - 2.2% and -2.7%, 50% of the probe trips experiments errors between -15.5% and 10.0% and 80% of the probe trips have errors between -24.5% and 20.8%. The counterpart of figure 3.9(e) with absolute travel time errors is provided in supplementary S6.

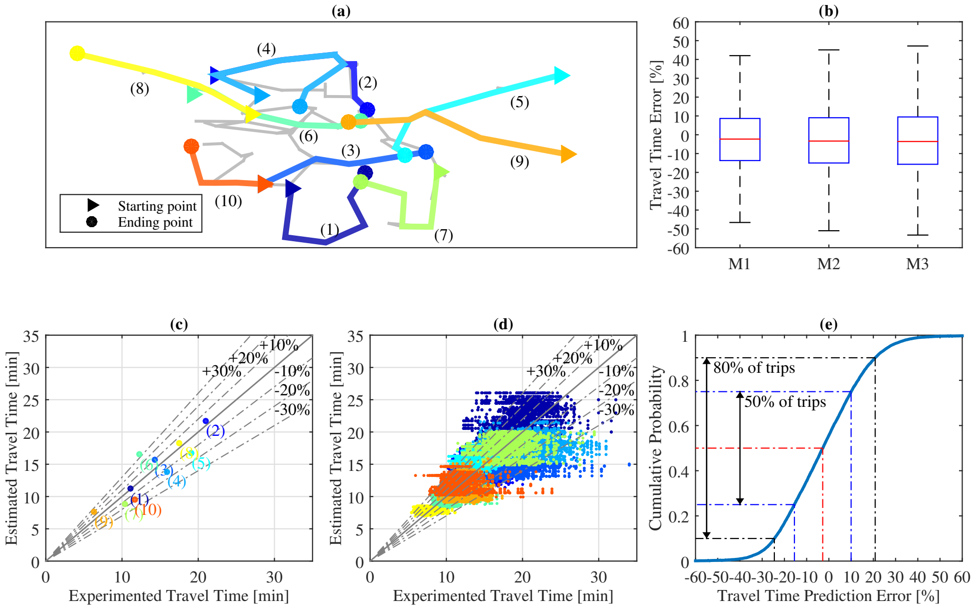


Figure 3.9: Travel time estimation based on congestion patterns. (a) Map of the probe trips (b) Travel time estimation errors for all probe trips and all training days considering the three estimation methods: M1, link speed is the mean speed in the original cluster; M2, link speed is the mean speed in the consensus cluster; M3, same as M2 but the mean speed is calculated over all days of the same group (c) Estimated vs. experimented travel times for the 10 probe trips, one validation day and a departure time equal to 9am (d) Estimated vs. experimented travel times for the 10 probe trips, all validation days and all departure times (e) Distribution of the travel time estimation errors for the 10 probe trips, all validation days and all departure times. (b) shows that travel time errors are in most case relatively low. Averaging speed within each cluster has the highest contribution to errors. Interestingly, using the consensus cluster shape ($M1 \rightarrow M2$) and the average of all days within a group ($M2 \rightarrow M3$) have very impacts on errors. (c-d-e) show that travel time predictions based on assigning a new day to an historical group and using the consensus cluster shape and the mean cluster speed of the group are very good for most probe trips.

3.2.3 Discussion

In this chapter, we questioned the regularity of day-to-day mobility patterns at the macroscopic level. The global analysis of Amsterdam link speed data over 35 days shows a high degree of regularity when comparing the daily congestion patterns. In our case, four consensual 3D speed maps related to four groups of days are sufficient to describe the daily traffic dynamics at the city scale. This is remarkable given the fact that these consensual 3D speed maps are very parsimonious: for our case study, they consists of 9 clusters (collections of link and time ID) only, each characterized by a single mean speed value. A key contribution here was to use consensus learning methods to turn the cluster shapes of different days belonging to the same group into a single common pattern. Note that if more days are available for the learning, it is possible to keep the same level of quality for the consensual shape by increasing the number of groups. The NMI index permits to monitor the level of dissimilarity within a group of days and determine if a group should be split or not. This chapter has thus demonstrated that consensual 3D speed maps are a new and very powerful tool to capture the congestion pulse in one shot at the whole city scale. It should be noticed that some factors that have not been observed during our sample of 35 days may influence the regularity of congestion patterns. From our experience, we can mention adverse weather conditions; exceptional (large cultural) events; or incidents as sources of major disruptions in the network. Over longer time periods, during which such situations are observed multiple times, the number of groups will increase to accommodate the resulting broader array of patterns, and most likely some regularity patterns with low frequency of appearance will emerge. Only the consequences of very rare or specific events are fully unpredictable.

A second major finding in this chapter is that these consensual 3D speed maps allow us to design a simple and systemic method to predict travel times in an entire city. In this method first prevailing link speed observations are matched to an existing group of days. Subsequently, the consensual 3D speed map related to this group is used to predict the travel time of any trip within the city. This method is *real-time and practice ready* as the matching step is computationally lightweight. It corresponds to the selection of the best consensual 3D speed maps among the existing group of days based on the comparison of the mean speed in each cluster. In our data set, we succeeded in making travel time predictions for more than 84% of the trips with an absolute error lower than 25%, which is sufficient for most potential practical applications like traffic information provision, route guidance, traffic control and management, or optimizing good deliveries and solving vehicle routing problems.

The methodology presented in this chapter to derive consensual 3D speed maps can be easily implemented in the real field. Link speed data at a granularity of say 1-10 minutes become more and more readily available thanks to advances in estimation methods using classical data (induction loops, cameras) and new data sources based on crowd-sourcing (mobile-phone records, GPS tracking). One clear direction for (methodological) improvement relates to decreasing computational costs, particularly when determining the initial 3D speed map for a new day on much larger networks in terms of number of links. Our aim was to make the case for 3D patterns as a new way to identify large-scale regularity in traffic networks and it turned out that with these 3D congestion patterns a new approach to a notoriously difficult problem (predicting travel times in urban networks) is possible. Even though optimizing the clustering and the post-treatment operations is very important for larger networks with (much) more links and data, it should be noticed that (continuously) learning and updating the consensual patterns with new daily patterns are off-line steps that can be performed over the night (determining the 3D congestion maps for a new day) and over the weekend (updating the consensual patterns). The critical component for real-time travel time estimation is the matching between the current observations and the historical data included in the 3D consensual congestion maps. With our method, this operation is so fast that it can already be applied in much larger networks. In this chapter, we do already hint at an important avenue to significantly cut computational costs for the original clustering operations. We constructed the 208 link graph of Amsterdam through *coarsening* the original 7512 link OSM network, using a constrained version of contraction hierarchy [151] as explained in [127]. Network coarsening [152] appears then as an efficient strategy to reduce the network size while preserving both network topology and the underlying data patterns. Also this strategy deserves further in-depth analysis and research.

Clearly, there are numerous other directions to further improve the methodologies behind the two contributions offered here. These relate for example to improve the underlying data processing methods, or to more advanced clustering techniques and matching procedures. Nonetheless, we believe the main results stand and touch upon a fundamental property of city traffic dynamics, and that is, that these dynamics may be more regular and predictable than expected. Consensual 3D speed maps enable us to extract the essence of large sets of detailed data to reveal the global picture about traffic dynamics in cities. We expect many applications of this concept not only for traffic monitoring and control but also for policy making and urban planning in general.

3.2.4 Methods

Initial dataset. In this study, link speed data are reconstructed from trip travel time observations. In Amsterdam, 127 cameras are recording license plates at the critical points of the major street networks (excluding freeways). This defines 314 single origin-destination (OD) pairs. For each OD pair the shortest path in distance is determined using the OpenStreetMaps GIS database [153]. The final network consists in all the links included in all the shortest paths, i.e. 7512 links in total. We apply an algorithm that merges together successive links in the same direction between two intersections. Internal links for intersections are also merged into a single node that only reproduces the available turning movements. At the end, the network has 208 links and 214 nodes [127]. The final step is to calculate the link speed information for 10 min time intervals from the individual travel times between OD pairs. We have a complete database of 35 days where we select the time period between 7am and 3pm (morning peak hour and lunch time). The mean number of individual travel time records per day is 171000. Each individual travel time information provides both the departure and the arrival times. All travel time data that exceeds a given threshold added to the current moving average for a given OD are considered as outliers and then disregarded (7% in total). The remaining information are then matched to links assuming a constant travel speed. We used a 10 min time window for link speed data, meaning that all observations coming from vehicles that drive through a link during the same 10 min period are averaged into a single link speed value. A complete description of the data preparation can be found in[127]. Note that the data processing in this chapter is not restricted to the data we used for the Amsterdam network but can be applied to any network with link speed information.

Ncut algorithm. Ncut is a clustering algorithm based on a similarity matrix *S(i,j)* that defines the level of similarity between two elements i and j of the dataset[122]. In this chapter, we use two different metrics to define the similarity: the Snake similarity[124] when determining the original clustering for each day and the NMI, eq. 3.12, when gathering days with similar patterns. More details about the Snake similarity are provided in supplementary S1. The different steps of the Ncut algorithm are:

1. Calculate the diagonal matrix D of the similarity matrix S
2. Calculate the normalized Laplacian matrix $L = D^{-1/2}(D-S)D^{-1/2}$
3. Calculate the eigenvalues of L and increasingly order the eigenvectors with respect to the eigenvalues
4. To obtain a partition in 2^m clusters, select the 2nd to the $(2+m-1)^{th}$ eigenvectors in the ordered list. The splitting point here is equal to 0 meaning that we separate for each eigenvector the values > 0 and ≤ 0. Each observation is then codified into a set of m binary values $>$ or ≤ 0 depending on the related values in the eigenvectors. Each observation with the same codification falls into the same cluster.
5. When the targeted number of clusters is not a power of 2, take the closest higher value for 2^m that then apply a merge algorithm. Clusters with the closest similarities are iteratively merged two by two[124].

k-means and DBSCAN. Before running the k-means or the DBSCAN we first normalized each observation i defined by the following vector (x_i, y_i, t_i, v_i), where x_i and y_i are the geographical coordinates of the middle of a link, t_i defines the time period and v_i the speed value. Normalization is performed based on the global minimal and maximal values for all coordinates. Speed values are then overweighted by a factor 3 because this variable should play a predominant role during the clustering process. For both algorithms, the distance between two observations is assessed based on the Euclidean one. The details of k-means algorithm can be found in [128]. The only parameter is the number of targeted clusters. The DBSCAN (Density-based spatial clustering of applications with noise) has been proposed by Ester et al. in 1996[129]. It is a density-based clustering algorithm that groups together points that are close, i.e. within a circle of radius ε (0.005 in our case). There is no targeted number of clusters but a minimal number of points to define a cluster (10 in our case). The algorithm stops when all points have been labeled. To obtain a given number of clusters, clusters are finally merged using the same algorithm as for the Ncut[124]. In practice both k-means and DBSCAN scripts have been retrieved from the MATLAB© File Exchange website[154, 155].

Chapter 4

Shape-based classification

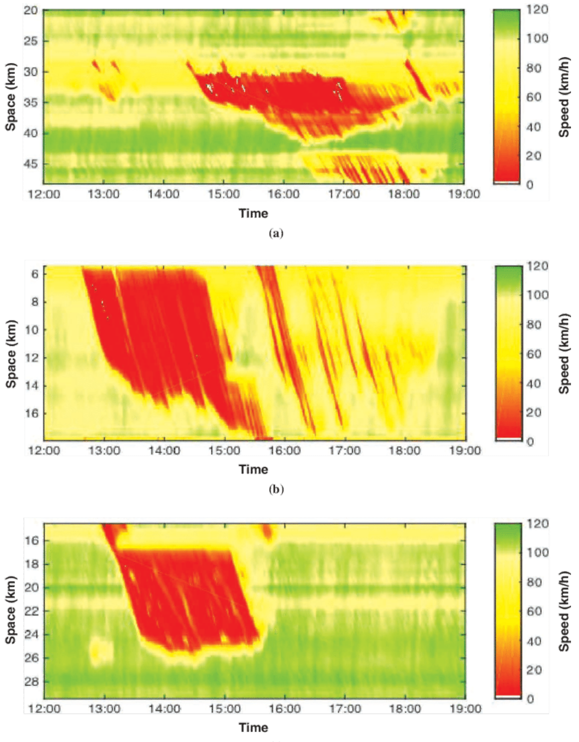
In the previous chapter, the consensus learning approach was used to extract the day-to-day regularity in data. However, this method relies on iteratively updating the speed of each link in the network to obtain an optimum pattern that fits multiple days, which is time-consuming. This chapter explores the potential of using a combination of image processing methods to identify and classify regions of congestion within spatiotemporal traffic contour maps to extract high-level features. The underlying idea is to use these regions as shapes that in many combinations can make up a wide variety of larger-scale traffic patterns.

This chapter is based on the following published paper:

Panchamy Krishnakumari, Tin Nguyen, Leonie Heydenrijk-Ottens, Hai L. Vu, and Hans van Lint. ”Traffic congestion pattern classification using Multiclass active shape models.” Transportation Research Record 2645 (2017): 94-103. https://doi.org/10.3141/2645-11

4.1 Introduction

In research, education and in practice, spatiotemporal contour maps of speed, density and flow provide an intuitive means to identify, study, explain and illustrate (longitudinal) traffic flow phenomena on the basis of either real traffic data or data from traffic simulation models. These phenomena include homogeneous congestion patterns at bottlenecks, reduced flows due to blockages, wide moving jams that propagate over large distances against the direction of traffic flow, or high density platoons of heavy vehicles that form moving bottlenecks, etc. With contour maps these phenomena become visible, which helps scientists to formulate hypotheses and derive theories and models to describe the underlying dynamics. Figure 4.1 for example shows speed contour plots on the A20 freeway between Rotterdam and Gouda (a); the A13 between Rotterdam and The Hague (b); and the A16 east of Rotterdam (c), respectively, which were used in a study of the traffic dynamics due to severe accidents.



(a)

(b)

(c)

Figure 4.1: Three severe incidents with similar characteristics in terms of e.g. incident characteristics, spatiotemporal extent of the queue, vehicle loss hours.

Constructing smooth contour maps from (sensor) data is relatively straightforward using the adaptive smoothing method introduced by Treiber and co-workers [156]; and further refined in e.g. ([157], [158]). Technically, contour maps represent matrices of traffic variables (densities, speeds, flows) on consecutive cells Δx along a route for consecutive time periods Δt, where neither Δx nor Δt need to be of constant size or duration respectively. When mapped onto an underlying grid $\{x_i, t_j\}$ with $i = 1, , N$ and $j = 1, , M$, traffic contour plots can also be understood as images with a color mapping from the traffic variable of interest to whatever color coding provides the required visual representation. This image representation opens up a huge array of possibilities for traffic scientists and engineers such as deriving distributions of wave speeds from raw traffic data, without making any prior assumptions [159]. These wave speeds and patterns can then be used for calibration and validation purposes of traffic flow models [160].

In this chapter we explore a different application perspective of image processing techniques within the traffic domain, i.e. the classification and identification of different traffic patterns. Classifying congestion patterns have two-fold applications for offline analysis and real time predictions. For offline analysis, it can be used to find days and routes in the historical database with similar congestion patterns. This can be helpful for the traffic managers to compare two incidents to gain insight for better traffic control. The classification of partial patterns along with metadata like incident location, severity, etc. can also be used for short-term predictions.

Classifying congestion patterns is certainly not a new idea. Kerner's ASDA/FOTO method [161] is a well-known patented, theory-laden approach and a multitude of machine learning alternatives are available [162–165]. Whereas the latter focus on class labels that indicate level of service (e.g. light, medium and heavy congestion); our aim, like Kerners', is to classify entire spatiotemporal congestion patterns. In contrast to Kerner, we do not employ an elaborate set of expert rules but flexible and data driven methods. In an earlier study [166], we use a supervised learning method to classify such patterns using multi-class Support Vector Machine (SVM). The contribution in that paper was to derive an equal-size feature vector for all small and larger traffic patterns identified in traffic contour maps. In the current chapter, we do not employ featured vectors, but (geometrical) shapes. To this end we use a methodology comprised of a number of image processing techniques to break down larger scale traffic patterns into smaller regions. The underlying idea is that these regions constitute base (archetype) shapes that in many different combinations can make up a wide variety of different larger scale traffic patterns. With robust identification methods for such archetype shapes, it becomes possible to dissect, identify and classify complex traffic patterns automatically. Despite its simplicity, we show later in the chapter that the classification results are quite encouraging with 70% accuracy by just using only two archetype shapes and simple logistic classifiers without resorting to the use of additional information (e.g. flow) as in the Kerners method. The applications for this technique are numerous and range from traffic database searching and indexing, to traffic state estimation and prediction.

The rest of the chapter is organized as follows. Section 4.2 gives an overview of the approaches involved in our multiclass classifier using Active Shape Models (ASM). Section 4.3 describes the experimental setup along with the validation method. The results of the method are presented and a synthesis of the findings is given in the section 4.4. We close with some preliminary conclusions and an outlook to the further improvement of the method.

4.2 Methodology

In this work, a fundamentally different approach than the state-of-the-art in traffic pattern classification is introduced and developed. Instead of using local features to identify char-

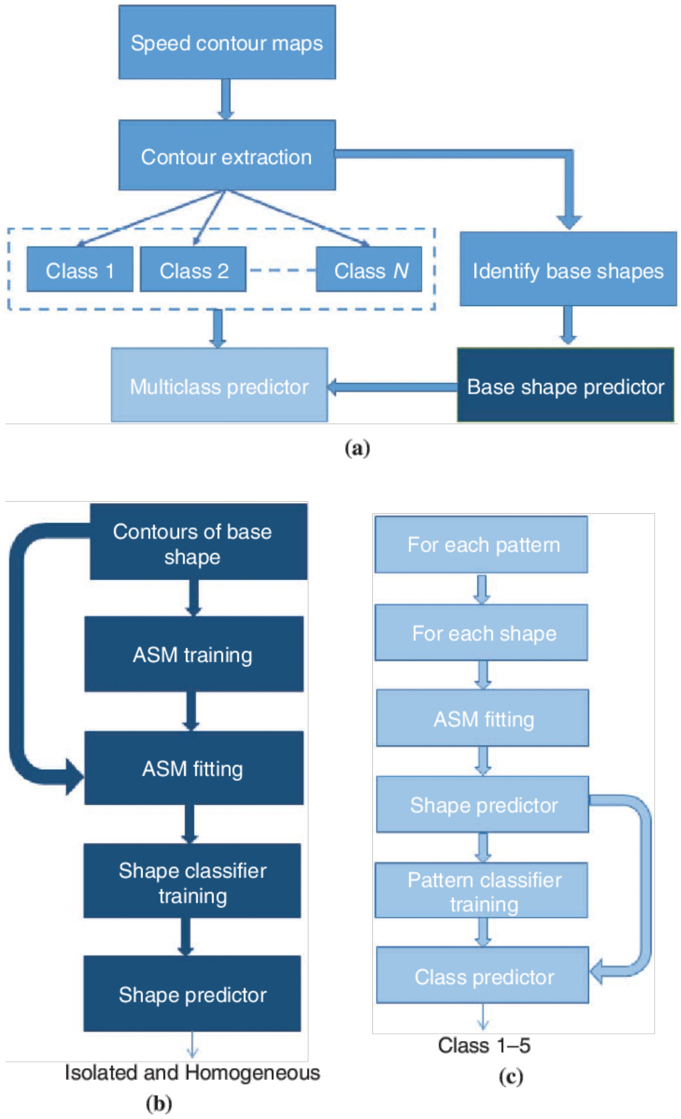


Figure 4.2: Complete overview of our approach. Note that ASM in this context stands for Active Shape Model.

acteristics of traffic patterns, we use (archetype) shapes to classify the patterns in order to capture the global structure of these patterns. This method essentially works at a higher abstraction level than feature-based methods. The shape based methods are usually used for shape recognition and fitting, mainly in the image recognition domain. This is the first work to the authors knowledge that introduces shape based classification in the traffic domain for multiclass classification purposes. The overall methodology is outlined in Figure 4.2(a).

The idea is to extract contours from speed contour maps and use these as the basis for building the shape model; the base shape classifier model; and the pattern classifier model. All these components are briefly explained below a full mathematical expos is beyond the scope of this chapter. The final part of the methodology explains how the fitting result from ASM has been used for multiclass prediction process. For more in-depth understanding on these topics, adequate references have been provided.

4.2.1 Contour Extraction

The basic ingredients for our method are contour maps of detector data generated with the adaptive smoothing method. For clarity, we refrain from denoting this method with ASM, because this acronym is reserved for Active Shape Model in this chapter. Since this method is extensively described elsewhere, we refer the reader to e.g. [156–158] for details. Raw data of one day is considered in one contour map, which therefore can contain multiple congestion patterns for a given day. The individual traffic congestion patterns from each space-time plot are extracted by a nave contour extraction.

The nave contour extraction aims at finding the outline of a pattern in an image as there are multiple patterns in one image, see Figure 4.3(a). The first step is to filter out irrelevant" information by assuming a speed threshold v_{thres} that differentiates between congested and freely flowing traffic. In this chapter we (arbitrarily) choose $v_{thres} = 65$. Note that this crude assumption can be relaxed we discuss this at the end of section 4. This results in a Boolean mask as shown in Figure 4.3(b). After thresholding, dilation is used to fill up the holes created so that we have a smooth mask. This smoothed mask is then used to detect the contours in the image by joining the continuous points along the boundary with similar pixel intensity [167]. The detected contours are used to define the boundary to extract each pattern in an image, Figure 4.3(c). Resulting from the nave contour extraction is a data set of different traffic congestion pattern images.

An additional contour refinement is then performed on the obtained patterns to extract the congestion shape from each pattern. The irrelevant information is filtered out from the nave extracted pattern image using the same assumption as before, low speed implies congestion as shown in Figure 4.3(e). Morphological closing strategy that consists of two successive binary transformations: dilation followed by erosion [168] is used to eliminate small and isolated gaps from the relevant regions without destroying the original shape as shown in Figure 4.3(f). For both transformations, a 3×3 cross structuring element was used. This binary smoothed mask is then used to detect the contours in the image (12) as shown in Figure 4.3(g).

4.2.2 Manual Classification

As a result of the refined contour extraction, we have a data set of different traffic congestion pattern images. This data set is manually classified into 5 classes depending on the size of images (space and time extent of traffic jam) and the type of congestion. Table 4.1 shows the five classes with some examples. This classification of course is arbitrary - other analysts may have come up with more or less classes and different criteria. We can also use unsupervised learning for creating these classes. However, this would require building feature vectors based on the application or using complete black-box methods like deep learning to find all the relevant features in the patterns.

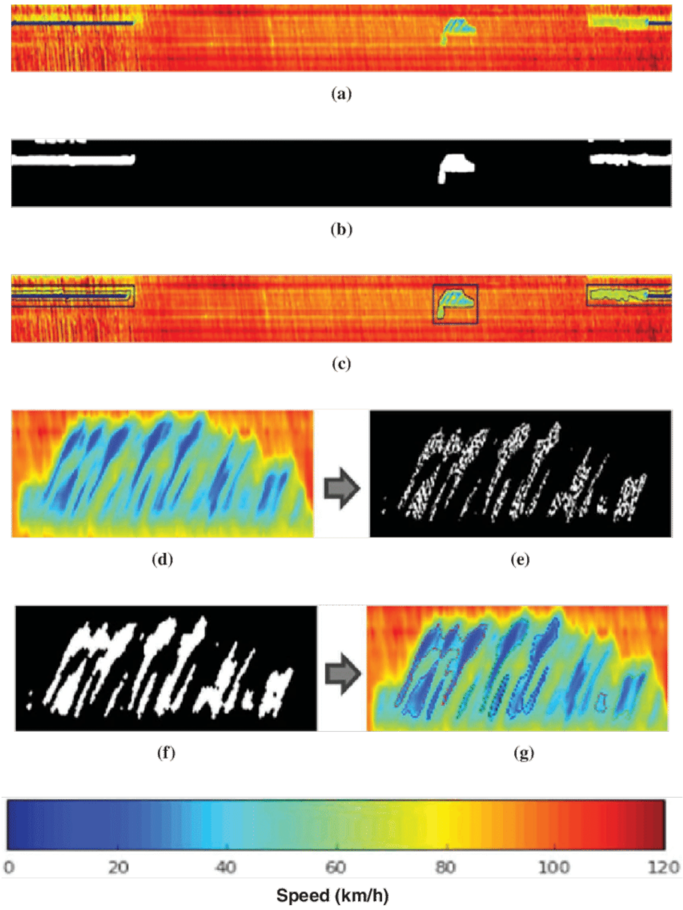


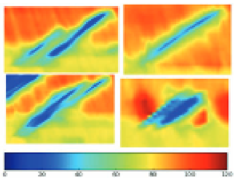
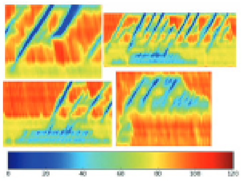
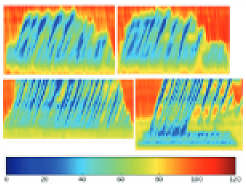
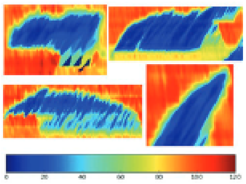
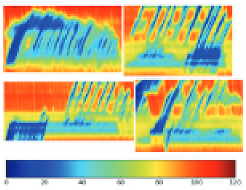
Figure 4.3: Contour extraction - nave and refined.

4.2.3 Base Shape Identification and Base Shape Predictor

From the manually classified data, one can observe that all the patterns in the class are approximately a combination of two base shapes, namely isolated WMJs (iWMJs) and homogeneous congestion (HC). For example, from Table 4.1, it can be seen that low frequency WMJ patterns are comprised of n number of WMJ shapes, and mixed class patterns are comprised of both HC and iWMJ patterns and so on. The key distinction in the context of this chapter between the two base shapes is that WMJs are stripe-like shapes, whereas HC patterns form triangular type shapes. This is also backed by theoretical researches which investigated wide range of congestion patterns [156, 169]. These works distinguishes two main types of congestion based on first order traffic flow theory synchronized and wide moving jams which includes similar characteristics as those in the base archetypes. Hence, we decided to start with these two distinct and well-defined base shapes. As more archetypes emerge from data, the proposed method can easily be scaled to include these base shapes.

The idea of Active Shape Model is that, given a new observed shape, we try to fit this

Table 4.1: Class Description For Manually Classified Pattern.

Name	Remarks	Examples
Isolated WMJs	Short (1–2 km) high-density traffic jams. Both head and tail of this queue propagate backward with virtually constant speeds (typically −18 km/h). WMJs typically result from large disturbances (e.g., abrupt braking) farther downstream.	
Heterogeneous Congestion 1, low-frequency WMJs	Large-scale light congestion patterns with a few WMJs emitting from the congested area.	
Heterogeneous Congestion 2, high-frequency WMJs	Large-scale light congestion patterns with many WMJs emitting from the congested area.	
Homogeneous congestion	High-density (low-speed) severe congestion regions, typically caused by incidents or other lane blockages. In terms of shape, the downstream front is stationary, whereas the upstream front moves with various shock wave speeds.	
Mixed large-scale pattern	Combination patterns not falling in either of the other categories.	

shape to one of the base shapes. To do this we have extended the originally single class Active Shape Model (ASM) algorithm to a Multiclass ASM (MASM) by including a linear classifier that predicts whether a given shape is HC or iWMJ. Figure 4.2(b) outlines the methodology to do this step by step. Below we briefly explain each step. Due to length limitations and for readability reasons, a full mathematical explanation of the ASM components including our multi-class extension is beyond the scope of this chapter. We first discuss the two main phases in ASM: constructing a statistical shape model (SSM) for each base shape (training) and fitting a new shape to this SSM (fitting).

Active shape model training

The ASM is a model based segmentation method introduced by Cootes and Taylor [170]. The method is based on the principle that a shape can be represented by a mean shape and its variations. The mean shape and the variances constitutes a so-called Statistical Shape Model (SSM) which contains all the parameters that are needed to define that shape. The SSM is used to find potential instances of the shape model in a new image/contour. Initially, a set of landmarks in the new contour is defined, after which the shape defined by these landmarks is deformed according to the SSM to provide the best fit possible within the SSM. The deformation is based on finding correspondences between the new shape and the shape defined by different SSM components and iteratively minimization a cost function for the fit. The allowed degree of deformation is constrained by the variations defined in the SSM. If the SSM includes large variances, large deformations are possible and vice versa.

An overview of the method is briefly explained below. A more detailed explanation can be found in Cootes et al. [170]. The steps for building the SSM model (i.e. the training phase) for a base shape are as follows:

Step 1 Align the shapes to the first shape in the dataset and generate a mean shape from the aligned shapes. Before this can be done all the shapes first need to have the same number of landmarks, which is rarely the case. To resolve this issue, we use a so-called iterative closest point (ICP) method to register the contours from all classes to a given model contour. In ICP, a point set is transformed in order to best match the chosen model, where the transformations are revised iteratively until the distance between the point set and the model is minimized [171]. The alignment itself is achieved using Procrustes Analysis as the standard ASM also uses this method for alignment readers are referred to [172] for details.

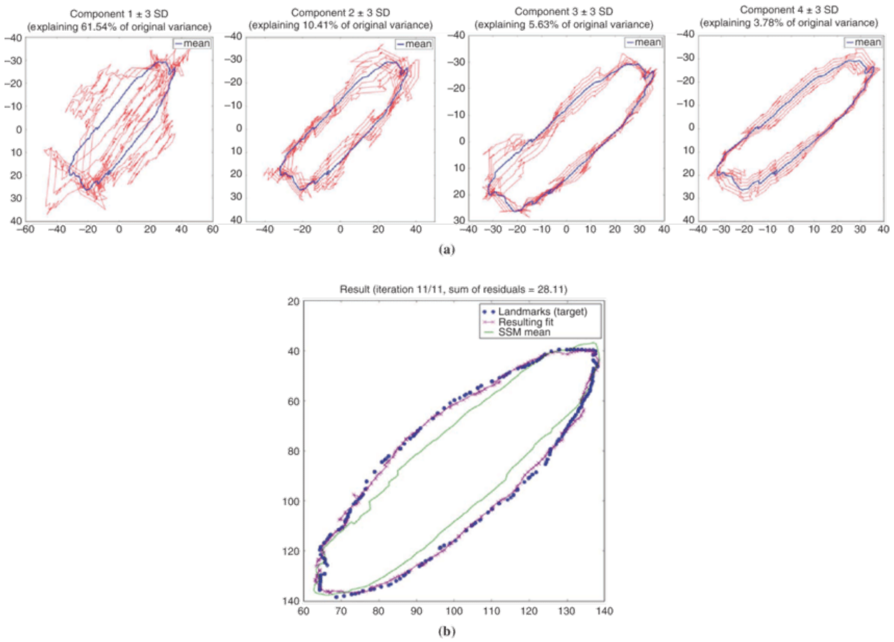
Step 2 Re-align the shapes to the mean shape and generate a new mean shape from the newly aligned shapes.

Step 3 Repeat step 2 (update the mean shape) until convergence.

Step 4 Finally apply Principal Component Analysis (PCA) to compute the Eigenvectors and Eigenvalues of the aligned shapes. The SSM components are the Eigenvectors of the centered shapes in the training data and the variances are the Eigenvalues of these shapes. If we apply PCA to the data, we can approximate any shape (within the training set) x (a n-dimensional vector of points) using:

$$x \approx \bar{x} + Pb \tag{4.1}$$

Here $\bar{x}$ denotes the SSM mean shape having point correspondences with x and the same dimension; P the SSM principal components; and b the parameters corresponding to the SSM components that deform the shape. This is the basis for fitting a new set of landmarks to the SSM. The first four components of the iWMJ base class are shown in Figure 4.4(a). This figure explains the different shape variations of the class according to the mean shape.



(a)

(b)

Figure 4.4: SSM and ASM fitting of the iWMJ (a) First four PCA components of the iWMJ shape model (b) ASM fitting for an isolated wide moving jam (iWMJ) shape after N^{th} iterations ($N = 11$).

Active shape model fitting

Given a new shape Y for testing, we first register the shape with the mean shape using ICP to compute Y. Using Y and the SSM model of the shape, the aim of ASM fitting is to find the model points x that best fit Y. The steps for ASM fitting are as follows:

Step 1 Initialize the shape parameter b as 0, implying model points = mean shape, i.e. $x = \bar{x}$

Step 2 Generate the model points positions using $x = \bar{x} + Pb$, where P is the SSM principal component.

Step 3 Find the pose parameters transform which best align the model points x to the new set of landmarks Y using Procrustes Analysis [172].

Step 4 Project Y into the model co-ordinate frame Y by using the inverse transform from eq.4.1.

Step 5 Update the shape parameters b to match Y' by finding least squared solution of $Ax' = B$, where A is P, x' is b and $B = Y' - \bar{x}$.

Step 6 Repeat step 2-5 until convergence.

An example of an ASM fitting result is shown in Figure 4.4(b). Note that the fitting error metric for ASM that is used as the convergence criteria is the Euclidean distance between the mean shape and the ASM fitted shape. The ASM method stops the iteration when there is no significant difference in the error metric result from the previous iteration and the current iteration. Here, we used an error difference threshold of 0.0001 which is statistically insignificant with respect to the error rate and the shape surface area.

Designing and training a base shape classifier

Now we have a working ASM model for both base shapes (iWMJ and HC), we need a shape classifier that can predict which of these base shapes provides the best representation for a newly found shape in a speed contour plot. To do so, we first fit the new shape with both base shapes. This will result in fitting error values, e_i (error on iWMJ base shape) and e_h (error on the HC base shape) respectively. Recall these errors are defined as the Euclidean distance between the SSM mean and the fitted shape. Additionally, to increase prediction accuracy, a third metric is computed based on additional non-shape properties of iWMJ and HC shapes respectively. As an example, in our case we choose a metric based on the spatiotemporal area covered by the shape a (in meter x seconds) and the gradient g (defined as the variation in speed in the given area a). We define the compound metric as the ratio g/a which can be understood as the amount of heterogeneity (variation in speed) per unit space x time, resulting in a three-dimensional feature vector (e_i,e_h,$\frac{g}{a}$). To build the classifier, we apply a well-known method in linear classification, logistic regression (18). It is originally a conditional probability model which measures the likelihood relation between a specific output and input by using a logistic function as the following:

$$p(t) = \frac{1}{1+e^{-t}} \tag{4.2}$$

Here, t is a linear combination of input feature vector x and β is the vector of coefficients which is considered as model characteristic. In preparing training data, isolated and homogeneous shapes are labeled as 0 and 1 respectively. These numbers are supposed to be outputs of logistic function. Coefficient vector β_{SC} is trained to minimize the cost of matching logistic function to training data which is measured by Euclidean distance.

$$cost = \sum_{x_i \in Training\ set} (p(\beta_{SC} \cdot x_i) - y_i)^2 \tag{4.3}$$

$$x_i = (e_i, e_h, \frac{g}{a}) \tag{4.4}$$

$$y_i = \begin{cases} 0, & \text{for isolated shapes} \\ 1, & \text{for homogeneous shapes} \end{cases} \tag{4.5}$$

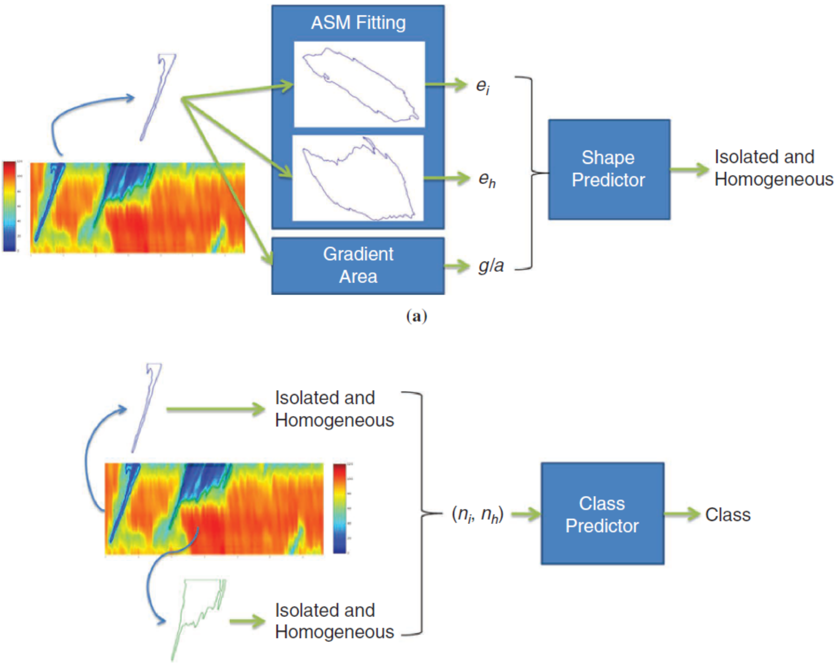
The resulting (trained) model is the (base) shape predictor which classifies a shape as either iWMJ or HC. The shape classifier will be used to build feature vectors for training pattern classifier for multi-class classification which will be described in section 4.2.4.

Using the base shape classifier

Given a new shape, the shape is fitted to the iWMJ and HC SSM model as described in ASM fitting to create a description vector $x_{SC} = (e_i, e_h, \frac{g}{a})$. This vector is used by the shape classifier model to make the decision based on the following equation:

$$\widehat{\text{shape}} = \begin{cases} \text{homogeneous shape} & ,\beta_{SC} \cdot x_{SC} > 0 \\ \text{isolated shape} & ,\text{otherwise} \end{cases} \tag{4.6}$$

This equation together with logistic function give a straightforward explanation for the classifier to make the decision. The new shape will be classified as a homogeneous shape if the probability given by logistic function is greater than 0.5. The overview of the method is shown in Figure 4.5(a).



(a)

(b)

Figure 4.5: Multiclass predictor - base shape prediction and class prediction (a) Base shape prediction given a shape (b) Class prediction given a new pattern.

4.2.4 Multiclass Pattern Classifier and Predictor

Recall that the aim of this work is to classify a given traffic (congestion) pattern into one of five predefined classes (see Table 4.1). The final step now is to design, train and test a classifier able to do this. A schematic overview of the method we propose is shown in Figure 4.2(c). Since each of these five patterns can be broken down into combinations of two (archetype) base shapes (iWMJ and HC), the first step in classifying a congestion pattern is to break down the pattern into these base shapes as shown in Figure 4.5(a). For each of the 5 classes we can now build a training set. where the output data is the class label (1 to 5) and the input data is equal to a simple 2D feature vector (n_i,n_h) in which:

n_i : number of occurrences of the iWMJ shape in the given pattern

n_h : number of occurrences of HC shape in the given pattern

The general idea of the multiclass pattern predictor is given in Figure 4.5(b). We use multiclass logistic regression to implement it. A simple, yet efficient method, one-vs-all scheme was used for training, which will construct one binary logistic regression model for each class, that assigns a probability p to the training sample belonging to this class versus the probability that it belongs to any of the other classes.

After training the classifier, we have 5 coefficients $\beta_{i=FC1..FC5}$ corresponding to 5 binary logistic regression models of 5 classes. For a given (new) pattern with feature vector $x_{FC} = (n_i, n_h)$, we now assign the class label to it that gives rise to the largest probability, that is

$$\widehat{\text{class}} = \arg\max_i \; p(\beta_{FCi} \cdot x_{FC}) \tag{4.7}$$

4.3 Experimental Setup

The data used in this study come from the National Data Warehouse for Traffic Information (NDW [173]). a Dutch organisation that archives and provides real-time access to traffic data from the Dutch agency of the Ministry of Infrastructure and the Environment (Rijkswaterstaat), the 12 Dutch provinces, two metropolitan regions and four of the largest cities in the Netherlands (Amsterdam, Rotterdam, Utrecht and The Hague). The data used for the experiments were collected from two heavily congested roads in Netherlands:

- The A13 Southbound from Den Haag-Zuid to Rotterdam center.
- The A15 Eastbound from Havens 5500-5700 to Rotterdam Ijsselmonde

Space-time plots of carriageway speed for these two roads were constructed using all available loop data for the entire month of March 2015. Each space-time plot represents a period of 24 hours from 00:01 to 23:59 at a resolution of 30s and 100m. Each of the plots contains multiple congestion patterns. After identifying and extracting each pattern separately using nave contour extraction, there were 140 traffic congestion patterns detected on the first road and 160 patterns on the second road. As we were interested only in large scale patterns with similar space-time ratios, 120 patterns were selected to create the classes. We manually classified the patterns into five different classes based on their spatio-temporal extent and the characteristics of traffic congestion as shown in Table 4.1. The number of patterns per class is small; this first trial is intended to demonstrate the ideas and learn lessons for larger-scale application. After the classes have been identified by the experts, the nave contours are refined to construct better distinctive contours from the patterns. As mentioned earlier, from these refined contours, two base shapes were identified. 35 contours were manually chosen to build the SSM models of both homogeneous and isolated base shapes. These same contours are used to train the shape classifier.

We used a simple one-verse-all (OVA) approach for the multiclass predictor that reduced the problem of classifying contours among 5 classes into 5 feature vectors, where each model discriminated a given class from the other 4 classes [174]. For this OVA approach, we had N=5 binary classifiers, where the kth classifier was trained with positive examples belonging to class k and negative examples belonging to the other 4 classes. The classifier that produced the maximum output was considered to be the best fit. Rifkin and Klautau [174] states that, provided that the binary classifiers are tuned well, this OVA approach is extremely powerful and produces results that are often at least as accurate as other more complicated approaches. The accuracy of the ASM OVA classifier was measured to judge the overall efficiency of the algorithm using the following formula:

$$\text{accuracy} = \frac{\text{Number of correctly classified images}}{\text{Total number of images}} \tag{4.8}$$

A confusion matrix, or contingency table [175], is constructed to help us investigate which class is behaving poorly and study that class further to understand the data better in order to make future recommendations to improve accuracy.

4.4 Result and Discussion

This section presents the results of the proposed method. The one-vs-all ensemble of ASM models achieved an average prediction accuracy of 70% which is relatively low compared to the state-of-the-art. However, this method represents 5 classes using only two archetype shapes whereas other methods need more degrees of freedom for defining each class. Thus, our method is able to constrain the classification complexity while providing satisfactory accuracy for a preliminary study with such a small dataset. A more elaborate confusion matrix is given in the Table 4.2. The table shows the percentage of correctly and erroneously classified patterns in each class given the ground truth. For example, for the iWMJs class, the ground truth patterns in that class were correctly classified with 81% accuracy, but there were some patterns in other classes which were wrongly classified as iWMJ. The High Frequency WMJs class contains the most patterns that were wrongly classified with 50% accuracy. We will discuss some of these wrongly classified patterns and the reason behind it.

Table 4.2: Confusion Matrix For OvA Evaluation.

	Predicted				
Known	Isolated	Low Frequency	High Frequency	Homogeneous	Mixed
Isolated	0.81	0.03	0.03	0.13	0.00
Low frequency	0.03	0.76	0.09	0.09	0.03
High frequency	0.00	0.28	0.50	0.16	0.06
Homogeneous	0.21	0.00	0.00	0.74	0.05
Mixed	0.05	0.00	0.22	0.17	0.56

From the Table 4.2, we can see that two classes have low accuracy compared to the other classes, namely High Frequency WMJ and Mixed. We hypothesize this is due to the small training data set and/or the limited representativeness of the samples in that class. We then investigate deeper into the data to qualitatively test this hypothesis.

Two of the wrongly classified patterns are shown in Figure 4.6(a); in this case both are wrongly classified as Mixed patterns instead of Low Frequency WMJ and Homogeneous Congestion respectively. It can be observed that these patterns are in fact Mixed congestion patterns. We can clearly see why the classifier would confuse these as they indeed have similar feature vector as the Mixed class. Clearly, successful classification depends on the subjective manual classification process, and the degree in which we succeed in labeling patterns that can be distinguished through OVA ASM. This is confirmed by an initial analysis of the wrongly classified patterns in which we agreed with the classifiers decision on 15% of all wrongly classified patterns rather than with the initial manual classification. In order to reduce the subjectivity of the manual classification, unsupervised clustering and reinforcement learning can be used as an alternative for building the classes.

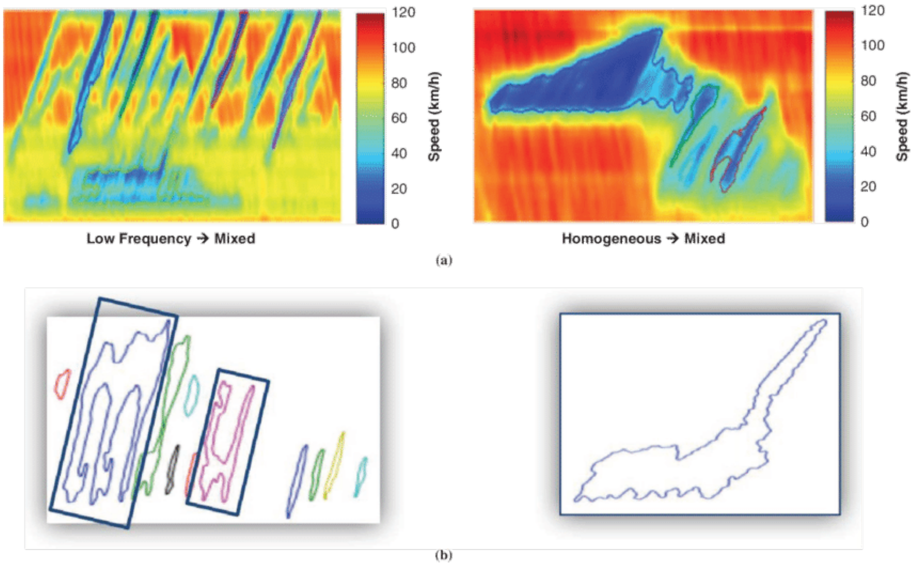


Figure 4.6: Synthesis of the wrongly classified patterns.

A second reason for the low accuracy are the constraints of the assumptions that have been made for extracting the contours such as thresholding at 65km/hr. These can be observed clearly from Figure 4.6(b) where the highlighted shapes consist of combinations of 2 or more isolated shapes and combinations of isolated and homogenous shapes respectively. A final point worth making is that we used exclusively speed to distinguish different patterns. Clearly, identifying and classifying distinct traffic patterns also requires information on the flows, particularly when distinguishing between congested patterns. Using just the speed and the nave single speed thresholding can only uniquely identify a limited number of patterns. To extend the repertoire of our classifiers we need to consider more archetypes to approximate all the unique shapes that are present within the patterns; we need to combine both speed and flow patterns; and we may have to use meta information (speed limits, geometry, etc.) to construct a more dynamic thresholding methodology to extract the shapes from the patterns.

Nonetheless, given all these limitations, we feel that the simplicity and transparency of the methodology (a few base shapes + logistic regression) offers great potential for further research and application development.

4.5 Conclusion and Future Work

In this chapter, we proposed a methodology, consisting of contour detection, the shape models and a multiclass OVA classifier, to automatically classify spatiotemporal traffic patterns using network sensor data. The different components proved sufficiently adequate to label complex traffic patterns with an acceptable accuracy, although the data set we used was too small to warrant definite conclusions. To this end, we have achieved 70% accuracy of the classification on the test data by just using only two archetype shapes and simple logistic classifiers. Furthermore, in the 30% of the cases the classifier could not decide on any of the five designated patterns was due to the fact that the contour shapes were affected by the assumptions and due to the subjective manual labeling to construct the training data.

There are many future directions of research that can further improve the accuracy. First, we can make use of the ensemble scores to directly put a confidence score to each classification. Further sophistication can be reached by ensemble bootstrapping or more modern Bayesian techniques. Second, we envisage an iterative manual-automated classification procedure in which we re-evaluate the manual classification after each training round using the classification scores of the OVA ASM. This may yield splitting or combining classes; hierarchically subdividing classes or otherwise. Third, we will combine meta data (type of date/time, circumstances, topological characteristics, etc.) to assist in classifying the patterns better. Another extension will be to combine speed plots with flow contour plots for better definition of different classes of congestion and for improved accuracy. Finally, we will continue to enrich our database with more congestion patterns and aim towards identifying more complex patterns.

Chapter 5

Image-based classification

In the previous chapter, we used domain-specific knowledge to define the high-level features. In order to enrich the feature space, we draw inspiration from the power of human vision in recognizing complex patterns and the success of computer-vision in mimicing those capabilities. Thus, in this chapter, we determine whether purely computer-vision-based features can be used to sufficently represent meaningful traffic states. In order to achieve this, we encode traffic states as images and use a pretrained deep convolutional neural network as a feature extractor.

This chapter is based on the following published paper:

Panchamy Krishnakumari, Alan Perotti, Viviana Pinto, Oded Cats, and Hans van Lint. "Understanding Network Traffic States using Transfer Learning." In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 1396-1401. IEEE, 2018. https://doi.org/10.1109/ITSC.2018.8569450

5.1 Introduction

For many applications within transportation, such as estimation and prediction of traffic conditions, and network-wide traffic control and management, methods to efficiently analyze large datasets associated with large-scale traffic networks are crucial. For prediction purposes, for example, by and large two categories of approaches can be identified - data-driven and model-based. Data-driven approaches use general purpose parameterized mathematical models to capture (learn) from data the correlations between traffic variables (speed, travel time, flow) over space and time. Examples include simple generalized linear regression [176, 177], support vector regression approaches [178, 179], and a wide range of different ANN models [143, 180, 181] to name a few (there are many overviews, e.g. [3–6]). In the last few years data-driven estimation and prediction approaches have been developed that operate on entire networks, particularly using deep learning techniques [43, 145, 146, 182]. Data-driven approaches require few prior assumptions; are typically robust to data failure; and can operate largely autonomously at good performance. The downside is that the past is not always a good predictor for the future, e.g. in case of incidents and accidents. Moreover, the lack of explanatory power makes them difficult to use for studying and exploring the actual traffic patterns, what-if reasoning or traffic management and control optimization.

At the other end of the spectrum we find (simulation) model-based methods. Examples include traffic flow models coupled with (extended) Kalman filters or more generally sequential Bayesian estimators[183–186]. The great advantage is that these methods provide an integrated solution for network wide state estimation and prediction, and that they use tractable behavioral and physical relationships, which make them highly suitable for studying and explaining traffic patterns, what-if reasoning, control optimization and application under non-recurrent conditions. The price for this explanatory power is that model-based methods are generally complex to design and maintain, and sensitive to data errors. Moreover, they require many inputs (e.g. traffic demand and control settings) and contain many parameters (driving and choice behavior) that need to be calibrated or even predicted from data. As such, model-based approaches, particularly in large networks, present many ill-posed problems and are typically highly underdetermined solutions given the available data. Even in cases we have abundant amounts of data, typically these do not encompass sufficient information (e.g. demand, behavioral relationships, traffic mix, route choices) for a large-scale model-based approach.

In this chapter we explore whether a data-driven technique *can* be used to shed light on spatiotemporal traffic patterns in large-scale networks. Many network spatial analyses methods are based on the assumption that a so-called macroscopic fundamental diagram is well defined for a homogeneous region [18]. Identifying such homogeneous regions allow us to model sub networks as reservoirs with predictable characteristics. Some of the works that are based on macroscopic fundamental diagrams create these homogeneous zones using network partitioning methods such as k-means [187] or snake similarity [188]. One recent and promising approach is to incorporate time into the spatial partitioning method based on macroscopic fundamental diagram, thus creating 3D zones [189] which can be used to create 3D network states [62]. Two recent papers take a different perspective to traffic data: they analyze *network* traffic as images [190, 191]. In the first paper [190], traffic data of a corridor is converted into spatiotemporal speed map; a convolutional neural network (CNN) is then trained on these maps (images) to make a traffic speed prediction. In the second paper [191], a so-called long short-term memory (LSTM) model is used to perform the predictions. The power of these ideas is that they allow application of machine learning techniques from the computer vision domain—e.g. deep convolutional neural networks—to traffic data.

Deep convolutional neural networks, when trained, act as hierarchical detectors of features, so that the learned features get progressively more complex (from segments to lines and contours) from the first layers further into the network, whereas the last convolutional layer detects high-level features [192]. In a trained convolutional network, the features detected by the convolutional layers are then correlated with the class labels by means of fully-connected layers. The main disadvantages of deep learning models are the computational resources needed, and the large amount of training data required to train a deep network from scratch. One of the emerging fields that try to overcome these limitations is transfer learning [193]. Transfer learning follows the intuition that many shapes and visual features are not domain-specific (e.g. a circular shape could correspond to an 'eye' in a face detector and with a 'headlight' in a vehicle detector), and therefore the features extracted from a network (trained only once on some generic-purpose dataset) can be used for different tasks.

To this end, we propose to use an opensource pretrained network to extract the relevant features from the traffic data represented as images. This is the first work to introduce transfer learning in transport, to the best of our knowledge. As of now, only limited number of features have been used to define network traffic states such as homogeneous speed regions, network connectivity, etc. From this work, the aim is to investigate whether features extracted from the pretrained model can be used to distinctly define network traffic states. These extracted features can then be used to enrich our knowledge into network traffic states. In this work, we also illustrate how these different network states can be used for look-ahead predictions, predicting the next traffic state given the current one.

The chapter is organized as follows: Section 5.2 discusses the proposed clustering technique for network traffic states using pretrained models and the one-step prediction. In section 5.3 we discuss Amsterdam data and parameters used for the methods. In section 5.4 we quantitatively and qualitatively discuss the results and conclude the chapter in section 5.5.

5.2 Methods

In order to use transfer learning, we have to use already existing pretrained models. However, most of the sophisticated models that are readily available are trained on images and not traffic speed data. So, the first step is to convert the network traffic variables into meaningful images that preserves both traffic data and spatial information. Once the data is transformed into images, we use pretrained deep learning networks to extract high level visual (spatial) features of the traffic network. These features are then used to automatically identify the different network traffic states. The traffic states are identified by clustering the feature vectors using hierarchical clustering and using the medoid of the clusters to represent the traffic state of that cluster. In order to illustrate the importance of identifying these traffic states and to access the quality of the clusters for different transport applications, we train a classifier to predict the next traffic state's clustering label, given the current traffic state. Thus, for network traffic state prediction using transfer learning, there are five steps - (A) data transformation, (B) feature vector extraction, (C) network traffic state clustering, (D) medoid construction and (E) one step prediction. These steps are explained in detail below.

5.2.1 Data transformation

Most of the pretrained deep learning models are trained on images. There are different ways to convert network traffic data into images. One such way is to construct the adjacency matrix weighted with speed and convert this matrix into an image. However, because of the scale-free property of most car traffic networks, the adjacency matrix is too sparse to extract any visible information. Furthermore, the adjacency matrix does not preserve the network nodes' relative positions (as rows and columns can be arbitrarily permuted). Another promising data representation method was introduced in [191]. The traffic data is encoded in a grid-like matrix where each grid represents a spatial region of $0.0001^\circ \times 0.0001^\circ$ (latitude $\times$ longitude), which is approximately 10m $\times$ 10m. In order to map the traffic data into this grid, the transportation network represented by latitude and longitude needs to be converted to a regular data grid with the aforementioned grid resolution and then the traffic data is encoded into the data grid.

To build the regular data grid, the maximum and minimum latitude and longitude of the transportation network are used to construct a boundary. A linearly spaced matrix with the given grid resolution is constructed. Now that we have the data grid, the next step is to encode the traffic data. The traffic data is usually mapped on a road stretch represented by a set of a consecutive coordinates. However, these coordinates are not uniformly distributed and might be more than the grid resolution, say 0.0001°, apart. Therefore, the coordinates of the road stretch are evenly spaced with 0.0001° between the points and then the grids that intersect with these points are filled with the traffic data value of the road stretch. An illustration of the steps involved in traffic data transformation is given in Fig 5.1.

5.2.2 Feature vector extraction

Transfer learning [193] can be implemented in several ways, but the common underlying approach is to use a pretrained model and either remove the last layers(s) or re-train them on the destination dataset (a process called *fine-tuning*). In this work, we are using the first scenario of the transfer learning where we use the pretrained model as the feature extractor by removing the fully connected layers, thus keeping the output of the convolutional layers. Most of the openly available pretrained networks are trained on ImageNet [194]: the ImageNet project is an ongoing effort and currently has 14197122 images from 21841 different categories; AlexNet, VGGNet, Inception, ResNet are some of the popular pretrained networks. We used Inception-ResNet-v2 network, an improved Inception network from Google, for the feature extraction which achieves 95.1% top-5 accuracy on ImageNet [195]. The assumption is that all the relevant features that make different traffic states distinguishable are captured by the deep convolutional layers and replacing the final layers with a simple clustering technique can provide meaningful groups of traffic states.

Due to weight sharing in the convolutional layers, it is easy to run a pretrained network on images of different size [193]. We fed each image, corresponding to a snapshot of the

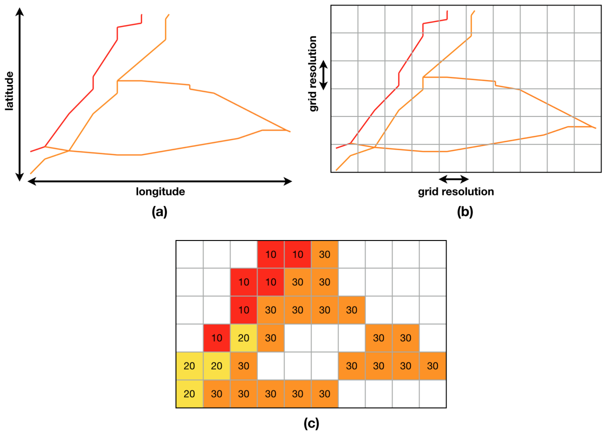


Figure 5.1: Data transformation - traffic data to image (a) Sample traffic network. The color represents the traffic variable value (b) Grid of specific resolution overlayed on the network (c) Traffic data embedded in the grid.

entire network at a given time, through the Inception model to extract the features. The Inception model has 128 filters of 3×3 kernel size and 32 filters of 5×5 size [195]; the dimensional space is progressively reduced by maxpooling, eventually resulting in a feature vector of dimension 1536.

5.2.3 Network traffic state clustering

To distinguish different traffic states, we cluster the feature vectors obtained from the pretrained deep nets. Since there is no general rule for defining the different network states, we have no prior information on how many traffic states there are. Two obvious anticipated categories are free flow and congested network states. Other than that, the rest of the network states heavily depend on the topology of the network and the demand and supply conditions. The main questions we hope to answer through the clustering is first, if the feature vector are distinguishable for obvious categories and second, if the feature vector can satisfactorily identify different and meaningful traffic states.

There are lot of options for feature vector classification. However, given that we do not have a priori information of the distribution of the types of network states, we opted for unsupervised hierarchical clustering. Hierarchical clustering builds a hierarchy of states based on the dissimilarity between them. By looking at the hierarchy construction, we can make a more informed decision about the number (and nature) of traffic states for a given network. The hierarchy can be constructed in two ways - agglomerative and divisive [196]. Agglomerative clustering is a bottom-up approach where all the samples start as a single cluster and then these clusters are merged together based on a distance dissimilarity measure to form new clusters, thus building up the hierarchy. In the top-down divisive clustering, all the observations or samples start in one cluster and are then split based on a distance dissimilarity measure for building the hierarchy.

In this work, we are using agglomerative clustering with the Euclidean distance as the dissimilarity distance measure. This method initiates each feature vector x_k as its own cluster. The connectivity between any two feature vectors, x_i, x_j with N dimensions, is calculated using Euclidean distance given by:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{N} (x_i^k - x_j^k)^2} \tag{5.1}$$

To measure the distance between two clusters, the average-link scheme is implemented which takes average Euclidean distance between all feature vector pairs from those clusters. Then, the two closest clusters will be merged to form a larger cluster. This process continues until only one cluster remains. The results of the hierarchical clustering is usually presented in a dendogram [197], a tree diagram illustrating the arrangement of the clusters. This provides a better overview of the structure of the hierarchy to judge the optimal number of clusters for this task.

5.2.4 Medoid construction

Now that we have the number of clusters, the aim is to construct a representative feature vector for each cluster. The most common one is centroids or mean of the clusters. But this leads to creating new data points which is unrealistic for traffic data. Instead, we are constructing a medoid of the cluster which corresponds to an actual data point in that cluster [198]. The medoid is chosen such that the average dissimilarity to all the objects within that cluster is minimized. Given a set of n feature vectors $x_1, x_2, ...x_n$ with N-dimensional real vectors and a dissimilarity Euclidean distance function d, the medoid x_{medoid} is defined in equation 5.2.

$$x_{medoid} = \underset{y \in \{x_1, x_2, \ldots, x_n\}}{\arg\min} \sum_{i=1}^{n} d(y, x_i) \tag{5.2}$$

5.2.5 One-step-ahead prediction

In order to illustrate how this kind of clustering can be used for real time applications, we demonstrate a rudimentary one-step prediction of traffic state in this chapter. Given the current traffic state, we predict the next traffic state using the feature vectors and cluster labels. The accuracy of the prediction is a measure of the usefulness of the feature vectors in defining traffic states and the quality of the clustering.

For the one step prediction, we are using an ensemble of classifiers: Multi-Layer Perceptron, K-Nearest Neighbors, Random Forest, Support-Vector Machine, and a Gaussian Process [199]. First, we randomly split the dataset into a training and a test set (encompassing 80% and 20% of the data, respectively). Each classifier is trained on the feature vectors of time n with the cluster label of time $n+1$ as the output. Note how this approach can be extended for k-lookahead prediction (simply by pairing the features vectors of time n with the labels of time $n+k$) and for taking into account the recent history as well (for instance, by including the feature vectors of time $n-1, n-2, ..., n-k$ as inputs). Each classifier has hyperparameters that were optimized by means of grid-search and random-search. Each trained classifier, when given an image from the validation set, outputs a probability vector with size equal to the number of clusters. For our prediction task, each element i of this vector corresponds to the estimated likelihood, for the traffic state determined by the input image and according to the trained classifier, that the state will evolve into a state belonging to the i-th cluster. The predictions of the five classifiers are then composed by means of the majority rule, thus creating an ensemble classifier.

A confusion matrix (also called a contingency matrix) [175] is used to analyze the accuracy of this global prediction against the ground truth. In this study, the ground truth is the hierarchical clustering label for the following time slice.

5.3 Experimental setup

The data used in this study is travel time data collected from license plate recognition systems at different critical points of the major street network of Amsterdam, The Netherlands. The shortest path between these points was mapped on Open Street Maps (OSM) to create the network shown in Fig 5.2(a). This mapped Amsterdam network comprises of 7512 links and 6528 nodes, excluding freeways. The travel time is converted into mean speed per link in the network for every 10 minutes between 00:00 AM and 23:50 PM for the 7512 links. Thus, there are 144 time slices for each day. The data is available for 42 days from 22 February 2015 to 4 April 2015. For more details about data preparation for creating the mapped network and converting travel time to speed, we refer to [189].

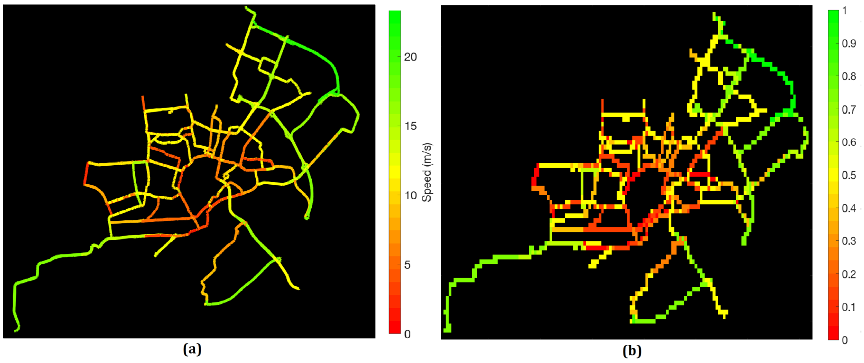


Figure 5.2: Data transformation of one time slice of Amsterdam network (a) Amsterdam network with latitude as x-axis, longitude as y-axis and the color represents the speed in m/s at that time slice (b) Corresponding data matrix with grid resolution 0.001°. *Speed is normalized between 0 and 1.*

For the data transformation, we used a grid density of 0.001° for both latitude and longitude, thus creating images of resolution 143 × 286 as shown in Fig 5.2(b). Some of the time slices in the 42 days have no observations due to faulty equipment and missing data. After removing these time slices, we have 5775 images for the 42 days. Inception-Resnet-v2 was trained on images that were scaled to 0 to 1. In order to have consistent data, we used a maximum speed of 25m/s (90km/hr) to normalize all the images, corresponding to the highest speed limit in the case study network.

5.4 Results and discussion

This section presents the results of the clustering and the prediction. The feature vector of 1536 dimensions are obtained from the Inception-Resnet-v2 model for the 5775 images. Thus, we reduced the complexity from 7512 to 1536 for a single time slice, approximately 80% reduction. Since some of these features might not be informative for the traffic problem, a further direction of work would be to further reduce the feature vector dimension either by investigating which of the layers in the deep network is returning a constant output for all images or using dimensionality reduction methods such as principal component analysis (PCA), linear discriminant analysis (LDA), etc. In this chapter, we use the 1536 dimensions to analyze the network states.

The dendogram of the hierarchical clustering of these feature vectors is presented in Fig 5.3. From the dendogram, the feature vectors are clearly distinguishable. The vertical height is an indication of the distance between the individual data points (feature vectors) or the clusters. We decided to set the number of clusters to 5 for understanding the different network states in each cluster. Each cluster is represented by a different color in the dendogram structure as shown in Fig 5.3. The cluster size can be modified per the application requirement.

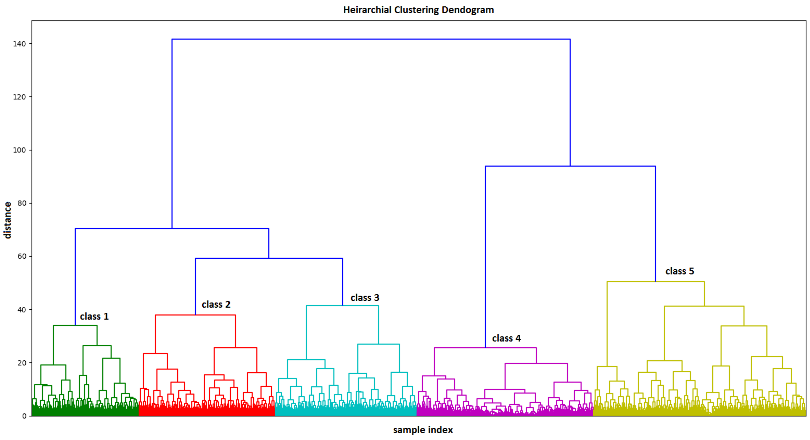
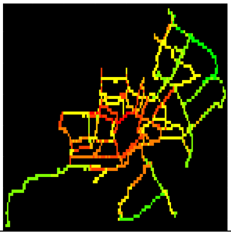
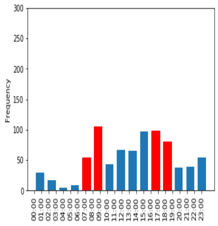
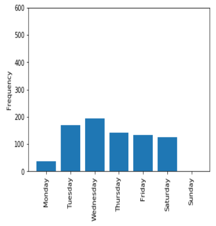
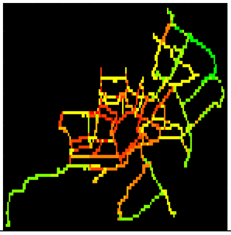
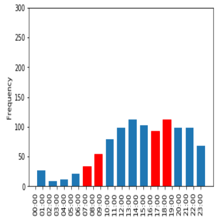
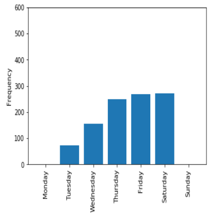
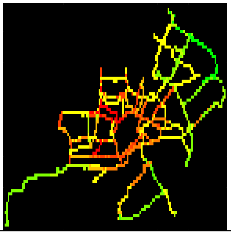
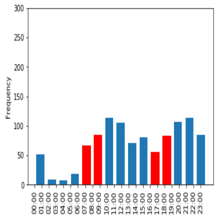
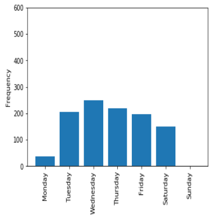
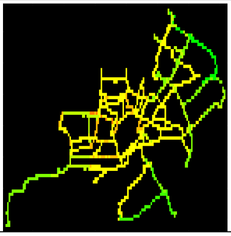
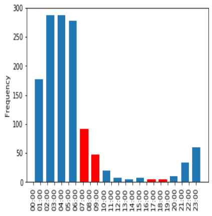
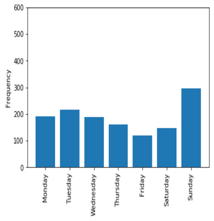
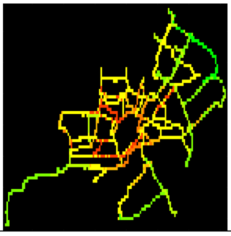
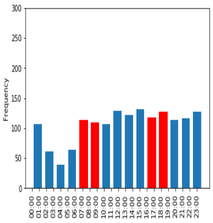
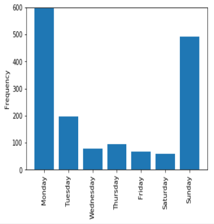


Figure 5.3: Dendrogram of the feature vectors

From the dendogram, we can observe that there are clearly two distinct branches comprising of (i) classes 1,2,3 and (ii) classes 4,5; and even further down the dendogram, there is clear separability. A more in-depth assessment on these classes is given in Table 7.1. From the medoid of these classes in Table 7.1, we can broadly identify these two branches as congested and free flow branch respectively. The congested branch comprising of classes 1,2,3 are more closer in feature space compared to the free flow classes as seen in the dendogram. This can be confirmed visually as well. It is hard to visually judge the difference between the medoid of these classes (Table 7.1) without additional meta information such as a difference images of the medoids or quantitative measures such as the average speed of the medoid of each class. A closer inspection does provide insight into the difference which is related to the difference in the spread of congestion into the network links. Fig 5.4

Table 5.1: Description of Amsterdam network traffic states.

Class	**Medoid**	**Distribution of time slices in each class**	**Distribution of days in each class**
1			
2			
3			
4			
5			

provides a quantitative variability between and within the classes - the distribution of the average speeds in each class. Even though the congested classes are similar, there is clearly variability in speeds and network structure that was captured by the deep networks. Thus, we broadly defined the class 1,2,3 as severe congestion network state, class 4 as free flow and class 5 as free flow with mild congestion. Based on the application, merging or further division of classes can be done. In this work, we proceed with 5 classes as the aim is not to find optimum number of classes but rather to see if predictable traffic pattern emerges.

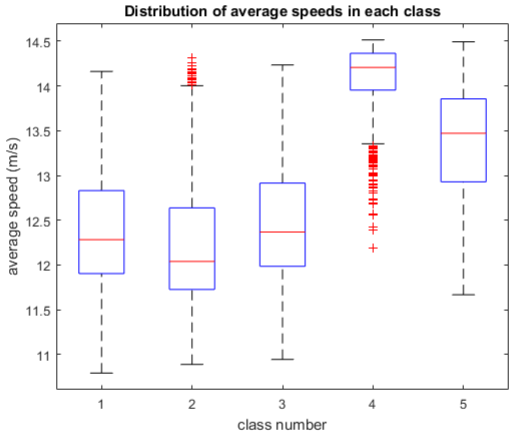


Figure 5.4: Variability of speeds in each timeslice within classes

A more in-depth look at the clusters also provides insight into the temporal differences among the clusters. The third column of the Table 7.1 shows the distribution of the time slices in each cluster. The first and second set of red bar plots corresponds to the morning (6.30 am to 9 am) and afternoon (4.30pm to 6pm) peak period in the Netherlands respectively. It can be seen that the afternoon peak period is more severe than the morning peak period for these 42 days as the occurrence of free flow class 4 for the afternoon peak period is significantly low. The most frequent congestion pattern during both the peak periods is the class 5, implying mild congestion is the regular network state in Amsterdam. Further study can reveal which network state corresponds to special or non-recurrent incidents, but this requires reliable incident data.

This study also reveals the day-to-day regularity in the congestion patterns by examining the distribution of days within each cluster, shown in the column 4 of Table 7.1. It is clear that severe congestion is absent on Sundays. Only class 4 and 5 network states occurred on Sundays within the 42 days. Another find was that Monday have less severe congestion for a working day with class 5 as its most recurrent network state. Monday is comparatively more similar to Sundays than the weekdays. Among the weekdays, Friday is the most severe in terms of ratio of frequency of congestion to free flow condition occurrences.

We present one of the application of network state clustering in this study - one-step prediction. This shows how time dimension can easily be integrated. Table 7.2 presents the 80 - 20 cross-validation results of the one-step prediction. It gives an average accuracy of approximately 58% using the naive classification approach. Even though the accuracy is not high, the classifier is clearly learning some patterns emerging in the temporal dimension using these feature vectors. More complex time series based methods, such as LSTM, can be used to encode the evolution of the feature vectors in the temporal dimension for better accuracy.

Table 5.2: 80-20 one-step prediction results

	PREDICTED				
KNOWN	Class 1	Class 2	Class 3	Class 4	Class 5
Class 1	0.28	0.23	0.32	0.07	0.10
Class 2	0.07	0.63	0.23	0.07	0.00
Class 3	0.13	0.29	0.47	0.05	0.06
Class 4	0.00	0.02	0.06	0.84	0.08
Class 5	0.02	0.03	0.08	0.17	0.70

5.5 Conclusion and further research

In this chapter, we introduced transfer learning to solve the problem of spatial clustering in the traffic domain. The assumption is that even though the pretrained networks are trained on natural images, the approach can be generalized to create feature vectors that can used to identify meaningful network states. This is proven in this study. The data dimension is reduced from 7512 to 1536, a reduction of 80% - simply by presenting data-as-images to pretrained neural networks. This process requires no training, and therefore is fast, scalable, and does not suffer from overfitting. This kind of analysis opens up many possibilities for new ways of looking at network traffic from the perspective of computer vision.

There are various promising future directions for this study. One of the main directions is to unravel the deep network to understand which filters are aiding in identifying these patterns. This can help in further reducing the dimensionality of the feature vectors and also in understanding some of the known or unknown traffic phenomena that the deep network identified in its 160 filters. Another topic that is interesting for future research is to look into 3D deep networks with 3D kernels or filters so that the temporal dimensions are incorporated into the deep network rather than approaching it as two step process - spatial, then temporal or vice versa. The spatiotemporal feature vectors could provide more insight into spatiotemporal evolution of congestion in a network. Since we have successfully applied transfer learning for understanding spatial patterns in network traffic patterns, other potential applications can be public transport network analysis, water network or any other domain with spatiotemporal data.

Part III

Applications

Chapter 6

Data-driven OD estimation

In the previous chapters, the traffic states clustering has been validated using traffic predictions. However, there are various other applications for such patterns. The fundamental challenge of the origin-destination (OD) matrix estimation problem is that it is severely under-determined. In this chapter, we propose a new data-driven OD estimation method for cases where a supply pattern in the form of speeds and flows is available. The minimal ingredients that are needed are (a) a method to estimate/predict production and attraction time series; (b) a method to compute the N shortest paths from each OD zone to the next; and (c) two possibly OD-specific assumptions on the magnitude of N, and on the proportionality of path flows between these origins and destinations, respectively. Here, the 3D supply patterns are used to predict the production and attraction time series, and the 3D spatiotemporal maps are used to compute the proportionality based on travel time.

This chapter is based on the following published paper:

Panchamy Krishnakumari, Hans van Lint, Tamara Djukic, and Oded Cats. ”A data driven method for OD matrix estimation.” Transportation Research Part C: Emerging Technologies (2019). https://doi.org/10.1016/j.trc.2019.05.014

6.1 Introduction

Understanding the dynamics of traffic demand over space (from origins to destinations) and time is quintessential for many applications over the entire transportation domain, from operations, control and management; to planning and policy assessment. Since we do not directly observe where everyone is coming from and going to at all times (yet[1]), we have to estimate time-dependent origin-destination (OD) flows $\mathbf{x}_k$ from whatever data and information *are* available. As it stands, the OD estimation problem is (still) one of the toughest problems in transportation to date. The most important difficulty in estimating OD matrices is that, particularly for large congested networks, the problem is severely under-determined, a fact that was already recognized in the early days of the OD estimation literature (e.g. [200–202]) and is emphasized in virtually all contemporary OD estimation research as well. To put this problem in a practical perspective, the OD matrix of the planning model for the Western part of the Randstad (The Netherlands) contains over two million OD-pairs, whereas literally orders of magnitudes fewer independent equations can be formulated to constrain these OD flows with actual observations, such as OD flow samples, links counts and speeds, that can be related to the OD matrix. Evidently, when we consider dynamics (e.g. with time steps $t+k\Delta t, k = 1,2,\ldots,$ and typically $\Delta t = 15,30$ or 60 minutes) the situation gets worse. Consequently, solving the OD estimation problem in large networks requires making a large number of assumptions.

There are essentially two paths to estimating OD matrices. One is a “forward path” in which demand is estimated (predicted) using models and data that link activity patterns, land use and zone production and attraction potential to the resulting OD flows. This encompasses a broad research area with modeling approaches ranging from estimating production and attraction potential of zones in an aggregated way to highly detailed disaggregate activity based models (e.g. [203–207]). The second path is to “reversely engineer” OD matrices by assimilating a wide variety of data sources into models that describe the assignment of OD matrices onto actual service and infrastructure networks and the resulting route choice and network traffic conditions. This is the common connotation of the term OD estimation and there is a long record of contributions in this area as well (e.g. [200–202, 208–216] to name a few in chronological order). Either way, in this data assimilation process, three types of assumptions play a role. First, there are assumptions about the quality and semantics of the available data (the observations, the evidence). Second, a large number of assumptions are incorporated in quantitative models describing the entire chain from activities to traffic flows, which govern how the data presumably relates in mathematical terms to OD flows. These are typically behavioral assumptions and assumptions related to how we represent and abstract transport and traffic flows (micro, meso, macro). Third, there are assumptions involved in the assimilation process itself, and in the supporting algorithms. This assimilation process may have the form of an off-line (bi-level) minimisation or maximisation problem (e.g. [213, 217, 218]), or of a sequential (recursive) estimation process (e.g. [211, 219–222]). Below we briefly overview the three categories of assumptions outlined above and how they—in general terms—affect OD estimation quality. We then use this review to motivate the approach we choose in this chapter, which we outline in subsequent sections. Note that the scope in this chapter is car traffic, and, in terms of the modeling literature we discuss, the “reverse engineering” path to OD estimation.

[1] unfortunately from a scientific perspective, but fortunately from a societal perspective

6.1.1 OD estimation assumptions: observations

Let us first distinguish between observations that can be directly (i.e. in closed form) related to specific OD flows, and those that can not. The first category of such direct observations are sampled OD matrices obtained from surveys (e.g. [201, 223]) or travel diaries (e.g. [205]). Increasingly, slightly larger samples can be obtained through vehicle (re)identification systems (e.g. [224–227]) or data from mobile phone or GPS based movement traces (e.g. [228–232]). Such OD samples $\mathbf{x}_k^{obs}$ are then typically used to construct a prior OD matrix, e.g. $\tilde{\mathbf{x}}_k = \beta \mathbf{x}_k^{obs}$, with β a possibly OD flow specific scaling factor. The standard way of exploiting the information in what are called prior OD matrices is to assume that $\tilde{\mathbf{x}}_k$ is informative of the spatial (in the dynamic case: spatiotemporal) structure of the (unknown) OD matrix $\mathbf{x}_k$. The widely used (static) formulation of the OD estimation problem (e.g. [233])

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} f_1(\mathbf{x}, \tilde{\mathbf{x}}) + f_2(\mathbf{y}(\mathbf{x}), \tilde{\mathbf{y}}), \tag{6.1}$$

exploits this assumption by adding a distance function $f_1(\mathbf{x}, \tilde{\mathbf{x}})$ to the overall objective function, that gently forces the OD estimation solution to minimize the distance between the final estimate $\mathbf{x}$ and the prior matrix $\tilde{\mathbf{x}}$—the second term f_2 penalizes errors on reconstructing the observations $\tilde{\mathbf{y}}$ using the OD estimate, which we discuss below. Clearly, a proper choice of the distance function f_1 is crucial. Standard distance metrics (L2 norm or RMSE) may not steer the estimation in a direction that favors similarity in spatiotemporal structure, for which other metrics such as the structure similarity index (SSI) may be more appropriate [234]. Regardless of how f_1 is formulated, the validity of the underlying assumption (that $\tilde{\mathbf{x}}$ has a similar spatiotemporal structure as $\mathbf{x}$) depends on the representativity of the sample for the population, i.e. whether the traveler traits (personal characteristics, income, car ownership, preferences, motives, etc.) and the corresponding destination choices in the sample represent a balanced cross section of the population [235]. With a large enough sample, survey and diary data collection efforts, can be—to a degree—controlled for representativity. This is more difficult in the case of movement traces, which typically come (to the researcher) without these contextual and privacy sensitive data [228]. There is a clear and large potential of using both active and passive mobile data for inferring activities, origins, destinations, paths and modes from such movement traces (e.g. [236–239]). [240] convincingly show that mobile traces may improve determining activity patterns, however, it is unclear whether, and under which circumstances, the (much) larger sample sizes obtained with these data *indeed* eradicate a possible representativity bias in the resulting (prior) OD matrix, which would invalidate the assumption that the spatiotemporal structure of a prior OD matrix is fully informative for the population OD matrix.

Under two very strict assumptions, also traffic counts (at cross sections in the network) can be directly related to specific OD flows.

$$y_{d\ell} = \sum_{i,j,k} a_{ijk}^{d\ell} x_{ijk}, \tag{6.2}$$

where $a_{ijk}^{d\ell}$ depicts an assignment matrix that maps the portion of each OD flow x_{ijk} between i and j departing at period k to the count location d in period ℓ. The fundamental problem is that counts in or downstream of congestion are not informative of demand, but of (discharge) capacity [241]. More precisely, equation (6.2) is valid only and only if (a) $y_{d\ell}$ is observed in uncongested conditions; and (b) no OD path flow x_{ijk}^n experienced a saturated bottleneck upstream before passing cross section d. These assumptions are typically violated in even mildly congested networks. Under such circumstances the counts have no direct linear relation to specific OD flows. Nonetheless, they are still part of an emerging traffic pattern that may be typical for a particular spatiotemporal OD pattern. As a result, there may exist a complex and nonlinear relation between those counts and the overall spatiotemporal OD pattern or perhaps a specific subset of OD flows. The same can be said for many other observations such as densities, speeds, travel times, realized paths, etc. Also these are (highly) non-linearly related to the prevailing dynamic OD pattern or a specific subset of OD flows, however, the form of this relationship is way more complex than a simple linear sum as in equation (6.2). A possible alternative interpretation is that $a_{ijk}^{d\ell}$ in (6.2) depicts a probability [239] that is governed by a nonlinear process that maps the OD flows to the network. This naturally brings us to modeling assumptions.

6.1.2 OD estimation assumptions: modeling

The unobserved OD pattern $\mathbf{x}_k$, in combination with the available network (coded in a graph G); prevailing traffic control $\mathbf{u}_k$; route-choice and driving behavior of travelers (ϕ_k, and θ_k); and the external conditions ω_k (e.g. weather, information) affecting these, ultimately result in an (at least to a degree) observable traffic supply pattern $\mathbf{y}_k$, in terms of flows, speeds, travel times, etc. The natural way to model this complex and non-linear mapping $\mathbf{x}_k \to \mathbf{y}_k : \mathcal{R}^n \to \mathcal{R}^m$ from OD pattern to supply pattern is to use a traffic simulation model of the form

$$\mathbf{y}_k^{sim} = \mathcal{A}(\mathbf{x}_k, \mathbf{u}_k, \hat{\phi}_k, \hat{\theta}_k, \omega_k) \tag{6.3}$$

in which $\mathcal{A}$ may represent any micro-, meso, or macroscopic traffic simulation model, and $\hat{\phi}_k, \hat{\theta}_k$ represent assumptions (mechanisms and estimated parameters) related to route choice and driving behaviour. Clearly, the linear mapping between counts and OD flows in equation (6.2) is a special case of (6.3), which is valid only, and only if, route choice is fixed and known, and the network is uncongested (i.e. has no capacity constraints), which means that under most conditions more complex simulation models are required. In many cases where such a simulation model is used, an additional assumption is added, which pertains to the existence of a (stochastic) equilibrium state. Whether and under which circumstances (for which purposes) the notion of equilibrium is a valid assumption is highly contested, due to the many open research questions related to the mechanisms with which travelers make choices and how rational these really are [242]. To the best of our knowledge, no conclusive empirical evidence has been reported that supports the existence of particular types of equilibria in actual large traffic networks.

As mentioned above, the standard approach to solve the OD estimation problem is to use the same DTA machinery to reversely engineer dynamic OD patterns from traffic supply data in (6.2), and by extension, *to use the same large amount of assumptions*, e.g. the notion of equilibrium (or not); the assumptions specified in $\hat{\phi}_k, \hat{\theta}_k$; and the assumption that all circumstances ω_k, and control $\mathbf{u}_k$ affecting route choice and driving behavior are known. In this reverse estimation problem, then iteratively an OD pattern is sought that is consistent with (a) the observed data; and (b) all the (behavioral) assumptions discussed here. Finally, there are also (necessary) assumptions pertaining to the data assimilation methodology, that may affect OD estimation quality.

6.1.3 OD estimation assumptions: solution algorithms

In illustration, we discuss sequential methods. Okutani was one of the first to formulate the time-dependent OD estimation problem [219] in state space form such that it can be recursively solved by a Kalman filter (KF), and many authors have followed along a similar path (e.g. [213, 220–222, 227, 243–247]). The central idea of these approaches is a formulation in state-space form, that is,

$$\mathbf{x}_{k+1} = F\mathbf{x}_k + \mathbf{w}_k, \tag{6.4}$$

$$\mathbf{y}_k = A\mathbf{x}_k + \mathbf{v}_k, \tag{6.5}$$

in which the dynamic equation (6.4) represents a linear (autoregressive) process driven by matrix F, and observation equation (6.5) refers to the assignment mapping (as in (6.2)), in which A depicts the assignment matrix. The appeal of the KF approach is that it offers a sequential estimation approach that is optimal in terms of error variance. However, the catch lies in the noise terms $\mathbf{w}_k$ and $\mathbf{v}_k$, which, to guarantee optimality, are assumed to be drawn from independent zero-mean Gaussian noise processes with known co-variance matrices. Put differently, the KF is optimal if both process and observation models are linear and unbiased and the errors they make come from known Gaussian white noise processes. Clearly, these assumptions are typically violated for numerous reasons (biased models, non-negativity flows, autocorrelation in the errors). As a result, many of the “KF approaches” to OD estimation propose ideas to either reformulate the problem, so that a KF approach is better justified—e.g. [210, 243] propose an OD state vector that consists of deviations between estimates and prior OD flows; [221] also propose a reduced state vector (obtained through PCA) and propose a ”coloured KF” to address autocorrelated errors. Alternatively, more advanced data assimilation methods are proposed to relax some of the highly restrictive assumptions, e.g. [222] propose a variation on the ensemble Kalman filter to alleviate the need for an explicit assignment matrix. In conclusion, whatever the (online or offline) solution methodology chosen, implicit and explicit data assimilation assumptions are necessary to solve the OD estimation problem, and the literature offers many great ideas to this end.

6.1.4 Motivation and rationale of a new approach

From the brief overview above, it is clear that to solve a severely underdetermined problem like OD estimation, one has to make many assumptions related to using the data, the models that relate these data to the unobserved OD flows, and the assimilation / solution methodologies to derive the OD flows from the combination of those data and models. However, it is not self-evident how many of those assumptions we really need. What is evident, is that the available data (the evidence) should dictate the amount of assumptions needed, and that we should choose methods that are parsimonious, since we generally lack the information to even estimate the magnitude and direction of the possible biases caused by (unnecessary) assumptions.

Now consider the case in which we have speeds either on each link, or in the form of so-called 3D supply patterns—speed averages over regions of the network over space and time (we return to this in the next section). In this case path travel times and consequently also all (or a selection of say N) shortest paths can be derived between any given OD pair.

This implies we do not need an iterative network loading process to compute travel times over the available paths between origins and destinations, since we already have those travel times. It doesn't matter for the OD estimation task whether this pattern of path travel times represents an equilibrium state, and what the precise nature of this equilibrium is. What matters is that these were the actual realized travel times, given the OD matrix we seek to find. To do so, we propose an idea that follows the same rationale as [213], in which "*an OD estimator is proposed based on the assumption of constant distribution shares across larger time horizons with respect to the within-day variation of the production profiles, leading to an estimator which improves dramatically the unknowns/equations ratio*". We (can) go a step further, since we have available *all* realized travel times over *all* (shortest) paths. We will show in this chapter that the assumptions we need to specify are (a) how many of the shortest paths were actually used for each OD pair; and (b) the proportions of each OD flows over these (N) shortest paths. The latter *is*, of course, a behavioral assumption, but at the macroscopic scale (a path flow proportion), and not in the form of a detailed route choice model with (elaborate) trade offs.

However, these two assumptions about the distribution of traffic over the network are not sufficient to estimate the underlying OD matrix, volumes are needed as well. To this end, we can use flow counts on those locations that meet the two strict requirements mentioned under equation (6.2). In the remainder of this chapter we will show that given production P_{ik} (sum of all outgoing OD flows of zone i during period k) and attraction A_{jk} (sum of all incoming OD flows of zone j during period k) are observable, the two distributional assumptions and counts are sufficient to reliably estimate the full OD matrix in smaller networks. For larger networks, however, we require additional constraints. We will show these can be derived *directly* from the data by using principal component analysis (PCA). Through PCA, we are able to find upper bounds for the so-called "dominant" OD flows, i.e. those OD flows that exhibit the largest dynamics. With these constraints then finally, also the underlying complete OD matrix can be estimated.

6.1.5 Chapter outline

To this end, section 6.2.1 describes the overall estimation framework; section 6.2.2 the basic estimation logic from production and attraction totals to OD flows; section 6.2.3 the PCA method to constrain the solution space for large networks; and section 6.2.4 the methodology to estimate the attraction and production totals from so-called 3D supply patterns. In section 6.3 we outline how we assess the method. We do this on two networks: a small network with which we vary extensively with the assumptions in our method; and a larger network to demonstrate the feasibility. These results are presented in section 6.4. We offer conclusions and a discussion on further research avenues in section 6.5.

6.2 Methodology

For convenient reference, the notation used for recurrent variables in the methodology is first presented as follows:

$\mathbf{x_{ijk}^{n}}$	path flows between $v_i, v_j \in Z$ for travelers departing in period k with $n = 1, \ldots, N_{ijk}$ paths

$\mathbf{x_{ijk}}$	OD flows between $v_i, v_j \in Z$ for travelers departing in period k
$\mathbf{P_{ik}}$	production (sum of all outgoing OD flows) of zone i during period k
$\mathbf{A_{jk}}$	attraction (sum of all incoming OD flows) of zone j during period k
$\mathbf{TT^{n}_{ijk}}$	travel time for vehicles traversing path n between node i, j departing in period k
$\beta^{\mathbf{n}}_{\mathbf{ijk}}$	route proportion
$\mathbf{y^{m}_{k}}$	link count for link e_m in period k
$\mathbf{FV}$	speed-flow based (SF) feature vector
$\mathbf{F\hat{V}}$	speed-flow-topology (SFT) based feature vector

6.2.1 Framework: OD estimation with minimal assumptions

Figure 6.1 outlines the main components of our framework. We use two data sources: 3D supply patterns—either in the form of the underlying "raw" link flows and speeds, or in the form of a condensed 3D consensual pattern [248]. The method allows for inclusion of additional data sources that provide more evidence for either the production and attraction patterns, or for the resulting OD matrices and/or path flows. The overall logic now is as follows. Assume that for a given set of zones we can estimate / predict production and attraction time series using the 3D speed and flow patterns, possibly augmented with other data sources and/or models. Figure 6.1(a) indicates that in this chapter we will use machine learning techniques for this purpose, but any other method would suffice as well. With the 3D *speed* patterns we also compute N^* weighted (by travel time) shortest paths (Figure 6.1(b) and (c)), where N^* is an assumption that reflects our belief into how many route alternatives—on average—were used for each OD pair. This assumption is likely topology and circumstance dependent. The second assumption we make (Figure 6.1(e)) is that the proportion of each OD flow on each of the N^* paths is inversely proportional to the (realized) travel time along that path, where we do take path overlap into account. We finally constrain the path flow solution space by using all the *admissable* link counts (Figure 6.1(d))—this will be elaborated further below. We explain our methodology in three parts: (I) the basic rationale of the method; (II) a dimension reduction technique (PCA) required to make the approach scalable for large networks (Figure 6.1(f)); (III) an example approach to infer production and attraction time series from 3D supply patterns using machine learning.

6.2.2 Part I: from production and attraction patterns to OD matrices

Consider a directed graph $G(V,E)$ with nodes (vertices) $v_i \in V, i = 1,\ldots,N_v$ and links (edges) $e_m \in E, m = 1,\ldots,N_m$. The set $Z \subset V$ describes the N_Z origin and destination zones in this network, and a dynamic OD matrix x_{ijk} describes the OD flows between $v_i, v_j \in Z$ for travelers departing in period k. These OD flows are distributed over a set P_{ijk} of N_{ijk} paths $p^n_{ijk} = \{\ldots, e_m, \ldots\}$ resulting in path flows x^n_{ijk}, with $n = 1,\ldots,N_{ijk}$. Our OD estimation

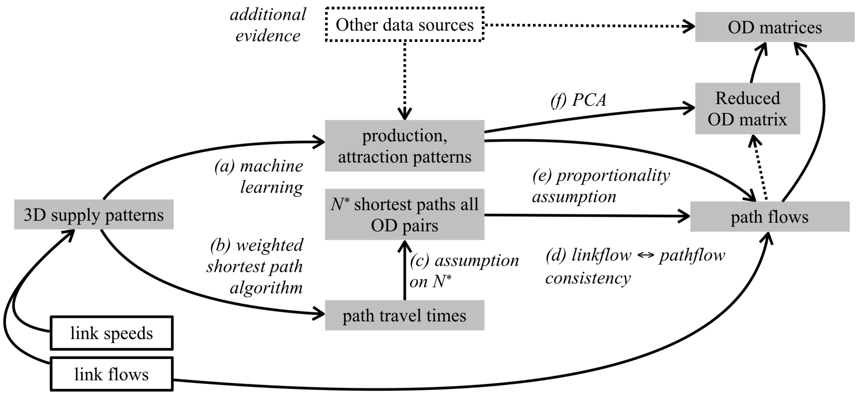


Figure 6.1: Framework of the method

method is based on utilizing three data sources; these are: production and attraction per zone; realized travel times on all routes; and (a limited number of) link counts. For now assume that we can either directly measure or estimate the production (sum of all outgoing OD flows) of zones i and the attraction (sum of all incoming OD flows) of zones j during period k (in section 6.2.4 we outline a machine learning method to do so). These data then add two constraints to the path flows

$$P_{ik} = \sum_j \sum_n x^n_{ijk} \tag{6.6}$$

$$A_{jk} = \sum_i \sum_n x^n_{ijk} \tag{6.7}$$

Secondly, consider we have available realized or estimated link speeds with which we can determine for every path n the travel time TT^n_{ijk} for vehicles traversing path n between node i, j departing in period k. Note that we define the travel time for vehicles traversing path n between node i, j arriving in period k by $\tilde{TT}^n_{ijk}$. We employ the conventional assumption that the distribution of OD flows over these paths follows the Logit model using the realized travel times as the key explanatory variable, that is,

$$\beta^n_{ijk} = \frac{e^{TT^n_{ijk}(1-PS^n)}}{\sum_n e^{TT^n_{ijk}(1-PS^n)}} \tag{6.8}$$

where β^n_{ijk} is the route proportion; and PS^n the path size factor defined as follows (Ben-Akiva and Bierlaire, 1999):

$$PS^n = \sum_a \left(\frac{l_a}{L^n}\right) \frac{1}{\sum_n \delta^n_a} \tag{6.9}$$

Where l_a is the length of link a, L^n is the total length of path n and δ^n_a is the link-path incidence variable which equals one if link a is on path n and zero otherwise. The path size factor inclusion in Eq.(6.8) ensures that paths are penalized based on the extent to which they overlap with other paths included in the set of considered paths. This formulation of the flow distribution assumption is inspired by the path size logit (PSL) model for determining individual route choice probabilities. Accounting for the effect of path overlap is especially important in the case of larger (and particularly in grid or triangular) networks, where the number of available paths between zones i, j increases exponentially. The key idea of our approach is that we cut off the number of considered paths, and compute the path proportions in (6.8) for a set P_{ij}^* of $N_{ijk}^* \leq N_{ijk}$ shortest paths. This implies we now have an approximation of the OD matrix that reads

$$\hat{x}_{ijk} = \sum_{n=1}^{N_{ijk}^*} \beta_{ijk}^n x_{ijk}^n \tag{6.10}$$

The quality of this approximation depends on the fraction of the OD flow that was indeed distributed over the cut-off set of paths P_{ij}^*; we return to this point below. By definition, we can also write

$$x_{ijk}^n = \beta_{ijk}^n x_{ijk}, \forall, i, j, k, \text{and } n \in \{1, \dots, N_{ijk}^*\} \tag{6.11}$$

Finally, consider we have - for a limited portion of the links - counts y_k^m that are informative of traffic demand, or more precisely of those path flows that go through link e_m in period k. This is the case if and only if (a) e_m is not congested in period k, and (b) none of the links upstream of e_m in the set of paths $p_{ijl}^n \in P_k^m, l \leq k - \overline{TT}_{ijk}^{n|m}, \forall i, j, n$ (i.e. all paths that traverse e_m during period k) were congested. $\overline{TT}_{ijk}^{n|m}$ depicts the partial arrival travel time (i.e. up to link e_m) along path p_{ijl}^n when traversing link e_m in period k. These counts now add a fourth constraint on the path flows, that is,

$$y_k^m = \sum_i \sum_j \sum_{p_{ijl}^n \in P_k^m} x_{ijl}^n \tag{6.12}$$

Equations (6.6), (6.7), (6.11), and (6.12) constitute a system of equations for the path flows and via (6.10), they can be defined based on OD flows as formulated in (6.13), (6.14), and (6.15). A full derivation of the system of equations is provided in A. This system is either underdetermined or overdetermined (or in rare cases full rank), depending on the available link counts, and the choice for the number of shortest paths N_{ijk}^*.

$$x_{i1k} + \cdots + x_{ijk} + \ldots = P_{ik} \tag{6.13}$$

$$\vdots$$

$$x_{1jk} + \cdots + x_{ijk} + \ldots = A_{jk} \tag{6.14}$$

$$\vdots$$

$$\beta_{11k}^n x_{11k} + \cdots + \beta_{ijk}^n x_{ijk} + \ldots = y_k^m \tag{6.15}$$

$$\vdots$$

This system of equations can be solved by reformulating it as a matrix equality

$$C\mathbf{x} = B, \;\; \mathbf{x} \geq 0; \tag{6.16}$$

where $\mathbf{x}$ is shorthand for the OD matrix x_{ijk}; C contains ones (LHS of equations (6.6), (6.7)) and the proportionality assumptions in equation (6.12); and B the RHS of equations (6.6), (6.7); and the LHS of equation (6.12), respectively. Matrix equality (6.16) can be solved using constrained linear least squares solution [249] with lower bound set to 0 to ensure a non-negative solution. In case a non-negative solution does not exist, an ordinary least square solution is computed and the negative values of the OD matrix are ignored when computing the estimation error.

6.2.3 Part II: OD estimation in large networks: reducing the solution space through PCA

Since the number of unknown OD flows grows quadratically with the number of (production and attraction) zones, and the number of rows in matrix equation (6.16) only grows linearly in the number of zones (6.6) & (6.7) and linearly in the number of link flow constraints (6.12); the linear system in (6.16) becomes (severely) under determined for large networks with too few link flow constraints. To solve the estimation problem for such large networks, we make use of the results in [211, 250] which suggest that, loosely speaking, the largest part of variance in demand flows can be attributed to a few dominant (temporal) patterns, that essentially reflect (daily, weekly) seasonal patterns; large deviations from those patterns, and random fluctuations around these patterns [250]. Our assumption is that a similar phenomenon may hold for the production and attraction flows P_{ik} and A_{jk}, and that—as in [250]—principal component analysis (PCA) can be used to (a) reduce the dimensionality of these time series and (b) as a result help us figure out which production and attraction zones $\tilde{P}_{ik}$ and $\tilde{A}_{jk}$ are sufficient to reconstruct a corresponding reduced set of (dominant) OD flows $\tilde{x}_{ijk}$ from which in turn (c) the complete OD matrix can be inferred. Put simply, because we expect a limited and predictable set of dynamical demand patterns, we may be able to infer these from just a limited set of production and attraction patterns.

We briefly outline the main rationale here; for further details on the PCA method, refer to e.g. [251]. Consider the time series of attractions $[\ldots, A_k^T, \ldots]$, with $A_k = [A_{1k}, \ldots, A_{N_Z k}]$, where N_Z is the number of OD zones and A_k^T is the column vector of A_k. Let μ_A, and Σ_A depict the mean and covariance matrix of A_k, and Γ the matrix of eigenvectors, so that $\Gamma^{-1}\Sigma_A\Gamma = \Lambda$ diagonalizes the covariance matrix; i.e. Λ is a diagonal matrix with the eigenvalues of Σ_A. With these eigenvectors and values, we can now express a set of new orthogonal uncorrelated variables called "principal components". Ranked in order of eigenvalues λ_i (diagonal element i in Λ); these principal components $p_i = A_k\gamma_i$, $i = 1, \ldots, N_Z$, with γ_i the eigenvector (column i in Γ), then depict those components in the data that explain most variance in the data in decreasing order. What this means is that if for example the first 3 principal components explain say 95% of the variance in the A_k time series, we can use just these 3 components to reconstruct the entire N_Z dimensional time series A_k with a loss of only 5% in terms of variance, which would entail a reduction in dimensionality of $3/N_Z$! Note that the columns in Γ represent the eigenvectors of Σ_A, the rows in Γ can be interpreted as the weights with which each original dimension in the data (i.e. the production or attraction of a particular zone) contributes to each principal component. We apply the same (PCA) process to the time series of production $[\ldots, P_k^T, \ldots]$. As a result, we can find the zones that correspond to the largest dynamics in both production and attraction and use the union of these selected zones for constructing a (strongly) reduced OD solution space.

Consider, for example, we find 3 “dominant” production zones and 2 dominant attraction zones. We can then estimate the 6 OD flows between this set of selected zones (we explain how below). These OD flows are *upper bounds*, because in computing them, we disregard path flows (6.12) between zones that are not in the “selected” set. In turn, these upper bounds can be used to constrain the system of equations in (6.16), which may now be solvable. We propose the following procedure:

1. Compute the principal components of A_k.
2. Select the n_p principal components that explain $X\%$ of the variance, with X an arbitrary threshold (e.g. 95).
3. Use a cutoff threshold $0 \leq \theta \leq 1$ for the correlation coefficient in Γ to determine the selection of original dimensions (zones) in A_k we consider are most relevant (i.e. are associated with most variance in the data).
4. Construct a reduced set of OD flows $\tilde{x}^a_{ijk}$ for $\tilde{A}_k$ using the selected zones.
5. Repeat steps 1-4 for $\tilde{P}_k$ to construct a second reduced set of OD flows $\tilde{x}^p_{ijk}$ for $\tilde{P}_k$
6. The union of $\tilde{x}^p_{ijk}$ and $\tilde{x}^a_{ijk}$ constitutes the final reduced set of OD flows $\tilde{\boldsymbol{x}}$ (shorthand for $\tilde{x}_{ijk}$)
7. Filling in $\tilde{\boldsymbol{x}}$ in (6.16) now leads to an inequality, since B will also contain OD flows *not* in $\tilde{\boldsymbol{x}}$. As a result we must now solve

$$C\tilde{\boldsymbol{x}} \leq B, \;\; \tilde{\boldsymbol{x}} \geq 0 \tag{6.17}$$

8. With (6.17) solved (i.e. $\tilde{\boldsymbol{x}}$ are now known), we may be able to solve the ”original” matrix equation in (6.16) using $\tilde{\boldsymbol{x}}$ as upper bounds for the corresponding set of OD flows in the full OD matrix $\tilde{\boldsymbol{x}}^+$, i.e.

$$C\mathbf{x} \leq B, \;\; \begin{cases} \mathbf{x} \geq 0, \\ \tilde{\boldsymbol{x}}^+ \leq \tilde{\boldsymbol{x}} \;\; where \;\; \tilde{\boldsymbol{x}}^+ \subset \mathbf{x}, \;\; and \;\; \tilde{\boldsymbol{x}} \subset \mathbf{x} \end{cases} \tag{6.18}$$

6.2.4 Part III: estimating production and attraction patterns

Let us emphasize on beforehand that deriving the production and attraction time series is a (replaceable) component in the OD estimation method. We may estimate such production and attraction patterns from demographic data or any other source using any appropriate method. In this section, we propose a machine learning approach to derive the relationship between demand and the (3D) supply patterns (Figure 6.1(a)). The overall idea for this approach to predict production and attraction patterns has its seeds in previous work on estimating so-called 3D (network × time) supply patterns [248, 252] using speed data available in urban networks. In [252] we show (a) that the network of Amsterdam over 35 days can be synthesized into only 4 such 3D patterns with each 9 clusters (a tremendous reduction in data that nonetheless reveals underlying regularity patterns); and (b) that these patterns successfully predict the travel time of 84% of all trips with an error margin below 25%. The hypothesis here is that such 3D supply patterns (that span (sub)networks and multiple time periods) are also informative of demand patterns.

We now seek a way to represent the traffic dynamics of the network with compact yet insightful feature vectors. We use two different approaches to build these feature vectors. The first is a Speed-flow based (SF) feature vector, in which we simply use the raw speed and flow time series in the network, and concatenate these as a (high dimensional) multivariate time series. The second is a Speed-flow-topology based (SFT) feature vector. In this case we use a condensed 3D pattern we build using the methodology in [248] and add topological information to it. This is a much lower dimensional feature vector. In the SF based approach, the supply information is formulated as a time series of speeds and flows, that is

$$FV = \{\tilde{u}_{pqk}, y_{pqk}\};\ \forall\ p,q \in V \text{ and } k \in \{1,...,n\}, \tag{6.19}$$

where, for a link between node p,q for period k, $\tilde{u}_{pqk} = u_{pqk} - u_{pq}^{max}$ is the speed relative to the speed limit; y_{pqk} the link flow; and n the number of time periods considered for incorporating the time dynamics of the supply information. The relative link speed is used since the network contains links with different functional road classes with different speed limits. Note that the SF based feature vector formulation does not contain any topology information or any relationship between the static OD zones Z in the demand space and the speeds or flows in supply space.

In order to incorporate spatiotemporal dynamics in the supply data, we construct the alternative SFT based feature vector using the 3D dynamic clustering proposed by [248]. The 3D clustering provides a compressed form to represent spatiotemporal dynamics in a supply pattern. The 3D clustering can be summarized in the following steps:

1. Spatiotemporal link speeds are clustered using datapoint clustering technique - we use Growing Neural Gas [253] to do this.
2. The unconnected clusters undergo post treatment to generate connected clusters that minimize the speed variance within the clusters and maximize speed variance between the clusters.
3. Each connected 3D cluster is represented by a single speed (mean speed of the 3D cluster) and flow (mean flow of the 3D cluster) values.

The result is a very compact representation of the same spatiotemporal dynamics contained in the raw speed and flow time series, but now in the form of a few 3D zone variables (average speeds and average flows). The OD zones are conventionally designed by experts with prior knowledge about the infrastructure network and using additional information such as surveys, socio-economic data, etc [254]. The OD zone is connected to the infrastructure network via unweighted directional virtual links known as connectors which are also defined by experts. A key missing ingredient is the relationship between the OD zones Z and supply space. We incorporate this relationship between these two different spaces using the 3D zones $\hat{Z}$ in supply space with

$$\hat{FV} = \{\delta_{ij}, \hat{u}_i, \hat{y}_i, \sigma^2_{u_i}, \sigma^2_{y_i}\};\ \forall\ i \in \{1,...,m\} \text{ and } j \in \{1,...,n\}, \tag{6.20}$$

where m is the number of 3D zones in the supply space and n is the number of OD zones in the demand space. $\hat{u}_i$ is the average link speed of the 3D zone i for all time periods k,

$\hat{y}_i$ is the average flow, $\sigma^2_{u_i}$ is the speed variance within the 3D zone i and $\sigma^2_{y_i}$ is the flow variance. δ_{ij} is the zonal incidence variable that represent the relationship between the 3D supply zone and OD zone given by

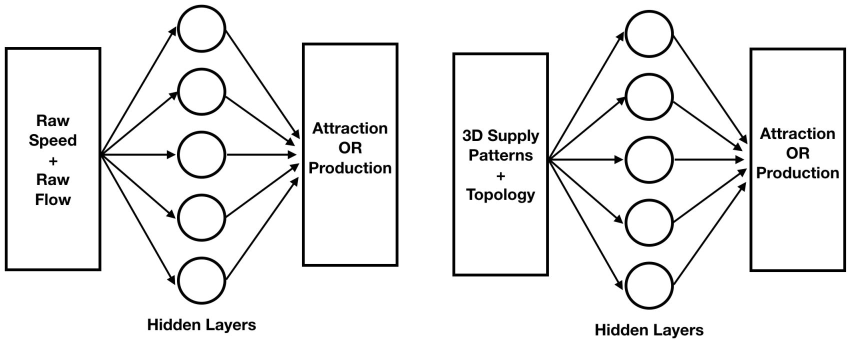
$$\delta_{ij} = \sum_{k=1}^{t} \delta_{ijk} \; \forall \; i \in \hat{Z}, \; j \in Z, \tag{6.21}$$

where δ_{ijk} is 1 if the 3D zone i intersects with OD zone j at time period k and 0, otherwise. t is the total number of time periods in the 3D zones. A 3D zone i intersects with OD zone j if a connector (that connect the OD zone to the supply space) directly connects that OD zone to any node or link in that particular 3D zone. This is under the assumption that the OD zones and the connectors between the supply space and OD zones are known. Thus the δ_{ij} represents the topological relationship between the supply and demand space, $\hat{u}_i$ corresponds to the zonal speed, $\hat{y}_i$ the zonal flow, $\sigma^2_{u_i}$ the zonal speed variance and $\sigma^2_{y_i}$ the zonal flow variance. These attributes together define the SFT based feature vector $\hat{FV}$. Additional attributes can easily be incorporated to extend the definition of the $\hat{FV}$.

There is no known relationship between supply patterns and productions and attractions and hence supervised learning is needed for estimating that relationship. Assuming that the feature vectors FV (6.19) and $\hat{FV}$ (6.20) that represent the supply space have non-linear relationships with the demand space (productions and attractions), we choose a non-linear machine learning approach to model this relationship - an artificial neural network (ANN). ANN is trained with the feature vectors (FV and $\hat{FV}$) as the input and some ground truth for production and attraction as the output. We choose a standard feed-forward model with three layers: an input, hidden and output layer, with linear output functions and the well known hyperbolic tangent sigmoid function as activation functions for the neurons in the hidden layer [255]. Two separate neural network models are trained (with feature vectors FV and $\hat{FV}$ as input vectors respectively) to predict the two demand variables (production and attraction respectively), resulting in a total of four neural networks.

In order to minimize model complexity (i.e. keeping the number of weights as small as possible) while retrieving the relevant relationship within the data, we use a relatively small neural net with just 5 hidden neurons. We did not perform an extensive method to converge to the “optimal” number of neurons in our models, but rather used a heuristic trial and error method. It turned out that SFT models did not converge with 1-4 neurons, but both SFT and SF did with 5, yielding the simplest still effective neural net design. Our assumption is that since traffic knowledge has been incorporated into the feature vector, the correlations with the demand and supply space will be more evident. Since the feature vector contains different types of measurements with different units, we normalize the feature vector in order to avoid biasing the neural net against a particular feature dimension or data source. The neural network models for FV and $\hat{FV}$ are given in Figure 6.2.

In this work, we use Levenberg-Marquardt optimization [256] for updating the weights and bias values of the neural network [257]. To avoid overfitting we choose a (time-consuming but robust) leave-one-out strategy for the training and testing. This strategy works like this: given a dataset of say 100 elements (inputs and targets), 99 are used for training and validation to construct the neural network whereas predictive performance is done on the remaining 1 data element. This is repeated iteratively for all the elements in the data set to achieve a robust prediction accuracy.



(a) *(b)*

Figure 6.2: Neural network model with 5 neurons in the hidden layers for (a) SF (with FV) based predictions (b) SFT (with $\hat{FV}$) based predictions

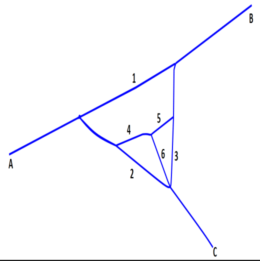
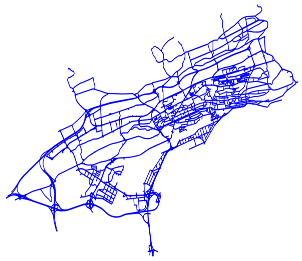
6.3 Experimental setup

6.3.1 Data and networks

As outlined in Figure 6.1, our method consists of two main components. The first is to predict time series of production and attraction for each demand zone; and the second is to use these predictions to estimate the OD matrix. We demonstrate and validate both components with two networks. First, a relatively simple toy network depicted in Table 6.1 (middle column), in which nodes A, B and C represent both origins and destinations. This results in 6 OD pairs, each of which has between 2 and 6 (reasonable) routes over which the OD flows are distributed. Note that the graph is directed; however, we use the same link number (1 to 6) to refer to both directions. We simulate this simple network in Aimsun Next [258] with a ground-truth OD matrix such that (mild) congestion occurs at the merges (i.e. upstream of some nodes). The entire framework is also applied to a much larger and (with actual data) validated network of Santander, Spain, to demonstrate its scalability in terms of estimation quality and in its ability to reconstruct the correct spatiotemporal structure of the OD matrix. The properties of the network and data availability of the toy network and Santander network are described in Table 6.1. Note that a zone as defined here may contain multiple origin and destination nodes (sources and sinks) that all connect to different links in the infrastructure network. In both networks the simulated data sufficiently cover supply space, that is, we have a sufficient spread over congested and uncongested conditions, varied over different parts of both networks. This assures the generalization power of the method for different traffic states.

A key assumption in this study is the size N^*_{ijk} of the path set P^*_{ij} between the OD zones for computing the proportionality (equation 6.8) and the time dependent travel time for the OD estimations. Although this is not necessary for the method, we choose one cut-off value $N^*_{ijk} = N^*$ for all OD pairs and departure time periods within a single scenario and vary with N^* over different scenarios (see below). We emphasize, however, that in principle N^*_{ijk} may be origin, destination, or OD specific and/or even time dependent. We return to this point

Table 6.1: Data description for the case study - toy network and large scale Santander network.

Properties	Toy Network	Santander Network
Network Description		
Network Size (Number of links, Number of nodes)	(40, 25)	(4205, 1630)
Number of OD Zones	3	115
Supply Aggregation	5 mins	5 mins
Demand Aggregation	10 mins	5 mins
Supply Time Periods	288 timeslices for 24 hours	48 timeslices for 4 hours of peak period
Demand Time Periods	96 timeslices for 24 hours	48 timeslices for 4 hours of peak period
Data Availability	7 days	7 days

in the conclusion and discussion section. For the toy network, it is viable to compute *all* the possible paths between each of the OD zone pairs. The paths between two zones i, j are computed by finding the paths between centroid of a given zone (source) to centroid of the other zone (sinks). This computation can easily explode when the network is larger. For example, using a simple Dijkstra algorithm to find the top 10 shortest paths between the 115 OD zone pairs in Santander requires a computation time of approximately 2 weeks on a decent 64-bit windows machine. There are different approaches for speeding up this computation, such as using link elimination to find alternative routes. In this study, for the toy network, we use all the possible paths between the OD zones; for the Santander network, we use the top 10 shortest paths between a given zone to the target zone.

6.3.2 Scenarios and performance measures

We conduct an extensive validation study on the toy network for the two components of our method described in section 6.2.2 and 6.2.4 using different scenarios. Firstly, we assess the quality of estimated OD matrix as a function of the accuracy with which we predict production and attraction. Prediction accuracy for all the scenarios in this study is measured using mean absolute percentage error (MAPE) given by

$$M = \frac{1}{n}\sum_{t=1}^{n} |\frac{x_t - \hat{x}_t}{x_t}| * 100\%, \tag{6.22}$$

where x_t is the ground truth and $\hat{x}_t$ is the predicted value. We consider 11 equally distributed prediction error bins from 0 to 100% to analyze this.

Second, we vary with the time aggregation of the production and attraction time series, and as a consequence of the dynamic OD matrix. For the feature vector that contains topology information, we use 3D zones through clustering. One of the key parameters for building these 3D zones is the number of clusters relative to the size of the network. A third scenario we consider is hence the number of clusters or 3D zones we consider. Fourth, we vary with the number of paths N^* between the OD zones. The optimal choice for N^* will likely heavily depend on the network topology but a thorough sensitivity analysis can provide insight into the number of paths required for fully describing the OD space of that network. We consider six "N^* scenarios": $N^* = 1,2,3,4,5,6$ for the toy network and 10 for Santander network.

Finally, link counts are necessary to solve the system of equations for the OD estimates. The link counts are available for all the links in the network for every 5 minutes. However, it is more important to have the counts at the right locations than having more counts at irrelevant locations. For the toy network, we study three link-count availability scenarios:

1. counts on the outer ring (links 1, 2, 3 in Table 6.1),
2. counts on the inner ring (links 4, 5, 6 in Table 6.1),
3. counts on both inner and outer ring (links 1 - 6 in Table 6.1).

This will shed some light on the sensitivity of the OD estimation in the absence of such data. Thus, with the 3 link count scenario and 6 N^* scenarios, we have a total of $3 \times 6 \times 11 = 198$ scenarios for the OD estimation in the toy network. For the Santander network, there are 334 detectors available and we use the link counts at these locations for the OD estimation error analysis. Thus, there are $10 \times 11 = 110$ scenarios for the Santander network with 10 shortest paths. A summary of all the parameters and scenarios used in the study is given in Table 6.2.

In the next section, we discuss the results of the two methods, that is, production and attraction prediction and OD estimation. To this end we give an extensive sensitivity analysis of different parameters in both the methods for the toy network along the dimensions in Table 6.2. A thorough sensitivity analysis is not performed on Santander. The aim here is to show that the method is scalable and that the assumptions are valid for different networks as well.

Table 6.2: Summary of the parameters used in the scenarios

Parameters	Toy Network	Santander Network
FV	X	
$\hat{FV}$	X	X
Aggregation	X	
Number of clusters	X	
Number of shortest paths	X	X
Link count availability	X	

6.4 Results and discussion

6.4.1 Production and attraction prediction

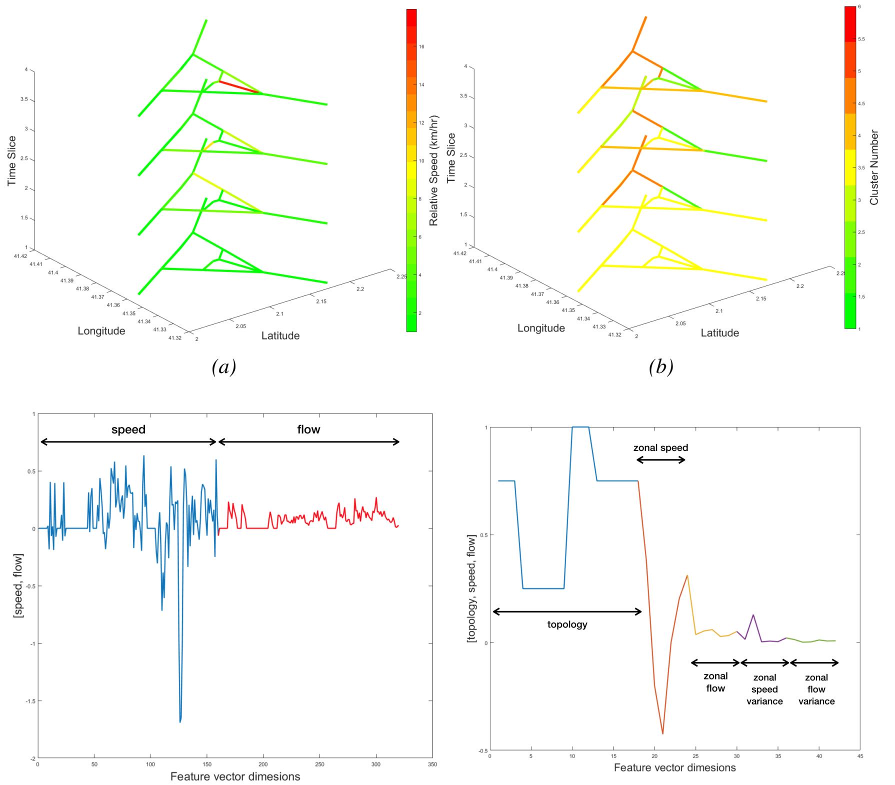
Before discussing prediction quality, we first discuss the differences between applying the two feature vectors we introduced in the previous section. Figure 6.3 illustrates the difference between the two feature vectors FV and $\hat{FV}$ (section 6.2.4) for a relatively short period of 4 time periods, see Figure 6.3(a). The color represents the relative speed (negative values imply a lower speed). Note that the relative speed of the toy network for the whole data set range from $-60km/hr < u_{pqk} < 30km/hr$. The associated 3D clustering results for the 4 example time periods into 6 3D zones are shown in Figure 6.3(b). The zones are clearly connected in space and time and each zone can be represented by a single relative speed (and an average flow computed for the same links). Figure 6.3(c) and (d) show representations of the feature vectors FV and $\hat{FV}$ respectively. The feature vectors are clearly different from each other for the same traffic state. Obviously, $\hat{FV}$ is much more compact than FV—it contains 6 speeds, flows and variances of both, and an incidence vector depicting how the six zones connect to the demand zones, in total 42 dimensions. The dimension of FV is much larger (over 320), and is a function of the number of time periods considered. Whereas the dimension of FV will explode with increasing number of time periods; the dimension of $\hat{FV}$ will remain the same.

Figure 6.4 shows a single instance of the production and attraction prediction accuracy using the FV and $\hat{FV}$ feature vectors respectively. Both the approaches have a median prediction error of $17-18\%$. Closer inspection shows that particularly in predicting production the $\hat{FV}$ results follow the ground-truth more closely (Figure 6.4)—a result we found consistently over the various scenarios. We can safely conclude that it is advantageous to construct a feature vector using the 3D supply patterns developed in [248]. By encoding mean and variance of 3D cluster speed and flow in combination with the topological connection of the 3D clusters to the demand zones, this reduced feature vector $\hat{FV}$ offers a parsimonious abstraction of the entire spatiotemporal supply dynamics of a network. It turns out that the neural net is quite able to correlate $\hat{FV}$ with production and attraction dynamics, and it does so equally well or better than with the much larger feature vector FV, in which simply a time series of all link speeds and flows are used. In the ensuing, we show result with the reduced feature vector only.

Figure 6.5 shows boxplots of the leave-one-out production and attraction prediction errors using the $\hat{FV}$ feature vectors with 6 3D zones and a time aggregation of 20 mins. The figure shows that 75% of the predictions for both attraction and production have errors less than 35% with a median error of 17% for each production and attraction dimension. This is solid support for the hypothesis that there are strong correlations between supply and demand patterns that can be exploited for OD estimation. We will discuss in the next section whether these errors are small enough to estimate the resulting OD matrix.

Figure 6.6(a) shows the relation between level of aggregation and prediction quality. Both attraction and production predictions show similar trends of increasing errors with larger time aggregation. For both we find a median MAPE error of $\approx 16\%$ for aggregation levels of 20 and 60 minutes. This is good news, a 20 minute dynamic production and attraction pattern provides a better basis for estimating dynamic OD matrices in many contexts than a very coarse aggregation that encompasses an entire peak period. The knife cuts both ways here: lower aggregation also provides more training data for the neural network. We return to this point also in the final section of this chapter.

Figure 6.6(b) shows the relation between the number of 3D clusters that constitute the $\hat{FV}$ feature vector and the production/attraction prediction quality. Our assumption was that as the number of clusters increases, we can capture more dynamics in the data. However, as can be seen from Figure 6.6(b), having more clusters in a small network actually impedes



(a) *(b)*

(c) *(d)*

Figure 6.3: Feature vector representation for SF based predictions (a, c) and SFT based predictions (b, d). (a) Speed for 4 time slices. Red represents high speed difference. (b) 3D zones for 4 time slices (c) Corresponding feature vector FV for SF contains normalized {speed, flow} as defined in (6.19) (d) Corresponding feature vector $\hat{FV}$ for SFT contains normalized vector for representing topology relationship between OD zones and 3D supply zones and supply information for the 3D zones as {topology, zonal speed, zonal flow, zonal speed variance, zonal flow variance} as defined in (6.20).

learning the patterns. In Figure 6.6(b), the graph is relatively stable until the number of clusters exceeds about 25% of the network size which is equal to 10 links for the toy network. In hindsight, this inverse proportional relationship between cluster size and prediction error can be easily understood. In the extreme case when the number of clusters is for example 50% relative to the size of the whole network, each cluster contains on average only 1 or 2 links. Such small clusters do not provide much meaningful information about the spatiotemporal traffic dynamics. Hence, the number of clusters have to be chosen relative to the size of the network. We will now look at the OD estimation results on the basis of these predicted production and attraction patterns. For this analysis we consider the optimal time aggregation of 20 mins, and a feature vector with 6 3D clusters.

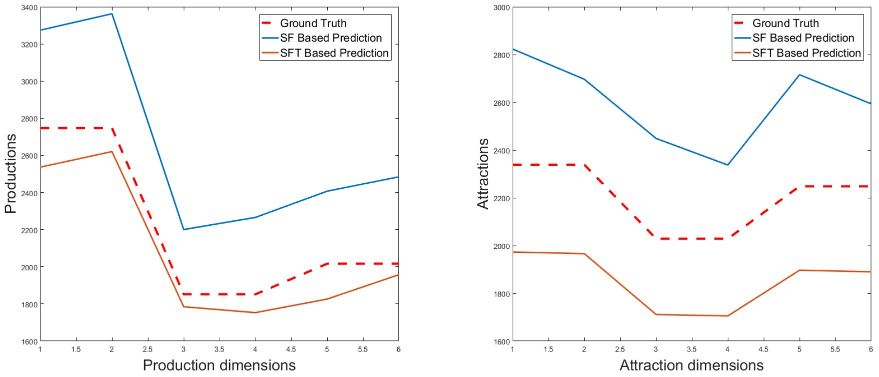


Figure 6.4: Single instance of production and attraction prediction for 20 mins time horizon respectively for toy network

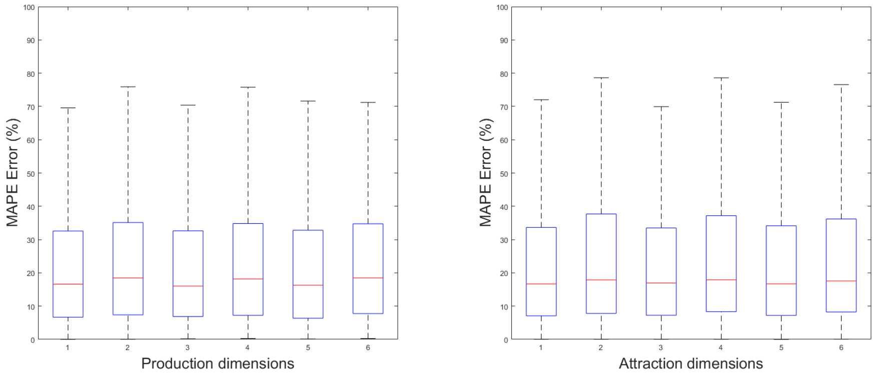
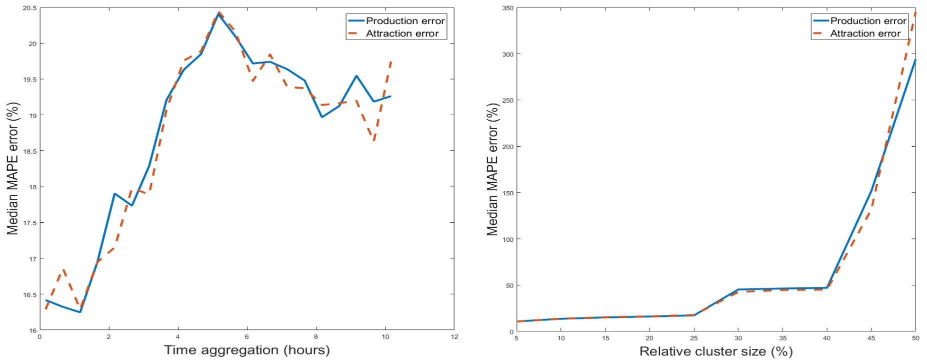


Figure 6.5: Leave-one-out production and attraction errors respectively for toy network.

6.4.2 OD estimation accuracy

Figure 6.7 presents the OD estimation errors as a function of (vertically) the errors in production (and attraction—these figures look very similar and are not shown here), and (horizontally) the number of shortest paths (N^*) that is considered. The errors are provided as a discrete contour plot, with the color indicating the error size. Each picture in Figure 6.7 represents 1 of the 3 detector availability scenarios for the toy network. The figure shows that the OD estimation error gradually increases with increasing production errors. A $0-10\%$ mean prediction error in the production results in an OD estimation error of $\approx 20\%$ in the first detector scenario (counts on the outer ring). The error then increases at an almost constant 10% rate with each 10% increase in production error.

The relationship with the number of shortest paths N^* is very different, There appears to be an optimal N^* for a given network to obtain the smallest OD matrix error—in the toy network case we find $N^* = 5$. The optimal N^* will likely depend on the network topology and the prevailing travel patterns in the network. OD estimation accuracy deteriorates when N^* is smaller than the actual average number of routes between OD pairs for obvious reasons:



(a) *(b)*

Figure 6.6: Sensitivity analysis of production and attraction prediction accuracy with respect to time aggregation and number of 3D zones respectively.

too much flow is distributed over too few routes. The estimation accuracy also deteriorates when N^* is larger than the actual average number of routes between OD pairs, because the proportionality constraint will now redistribute the flow to these additional irrelevant paths. Figure 6.7 provides clear evidence for this. For all 3 cases, the optimal N^* indeed equals 5. Note that we could have diversified N^*_{ij} for different OD pairs i, j to account for the fact that different OD pairs may have different "optimal" number of paths.

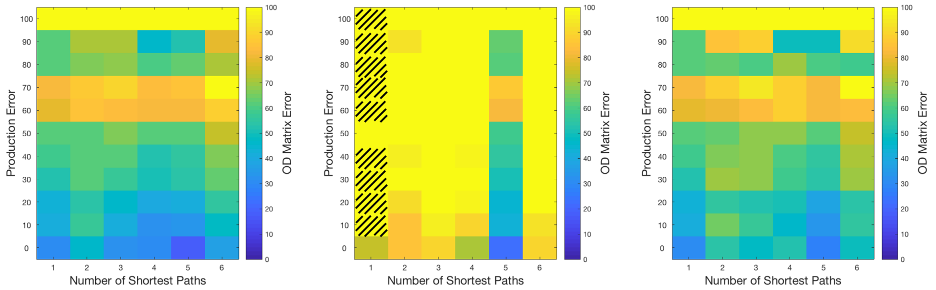


Figure 6.7: Mean OD matrix estimation error for the 198 scenarios of the toy network (a) Case 1 - counts on the outer ring, (b) Case 2 - counts on the inner ring, and (c) Case 3 - counts on the inner and outer ring. The shaded region in the case 2 corresponds to the cases where some of the OD matrix estimates returned negative values implying there is no solution for these cases. These estimates were not considered when computing the mean OD matrix errors.

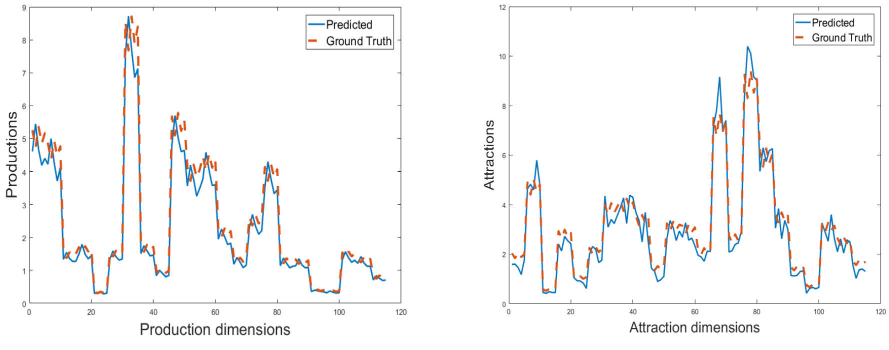
A look at the three different scenarios of link count availability in Figure 6.7 shows that case 1 and 3 outperform case 2. This is because the top shortest paths between the zones all use the links in the outer ring. This implies that the counts in the inner ring links are not used for solving the equations. However, case 2 also fairs better when $N^* = 5$, since this route set will now include links in the inner ring. When the number of shortest paths is 1 for case 2, some of the scenarios did not provide a non-negative solution. This is depicted by the shaded region in Figure 6.7. In all the scenarios non-negative solutions were found for the OD estimation. That case 3 (link counts in both inner and outer ring) does not provide better results than case 1 is easily explained. The number of link counts are not as important as having counts at relevant locations. The key lies in the number of linearly independent equations that can be constructed, i.e. in the rank of matrix C in equation (6.16).

The error distribution of the 198 scenarios can be inferred from Figure 6.7 and shows that approximately 34% of the scenarios have 100% estimation error. We did not explore these error regions in this chapter. However, a 100% error for a particular OD flow may not always be as detrimental as it sounds. Particularly for very small OD flows, of which in the Santander network there are many and in the toy network still quite a few, a 100% relative error may involve a very small absolute error. An alternative assessment method would be feeding the estimated OD matrices back into the simulation to check if it produces similar traffic conditions as the original matrix. Given all other variables (e.g. random seeds, driving and route choice parameters, control timings, etc.) kept equal, this may be a better verification method for OD estimation than simply assessing the OD error itself.

6.4.3 Santander case study

Finally, we apply the framework on a larger network of the city of Santander. We did not perform an extensive sensitivity analysis to estimate the optimal number of 3D zones and aggregation level for production-attraction prediction. We used an aggregation of 20 mins (which was optimal in the Toy network case). An initial attempt to use 10 3D zones was unsuccessful as the neural network was not able to converge. The traffic dynamics were perhaps aggregated too much and the neural net may not have been able to separate the resulting patterns. However, by using 20 clusters, the neural network was able to achieve a median accuracy of $20-22\%$ for both production and attraction. A look at one of the (favorable) predictions in Figure 6.8 shows that the neural network was able to capture the high-dimensional pattern accurately. In the figure we see for one time period the 115 values of the production vector (a) and the attraction vector (b) for one time period.

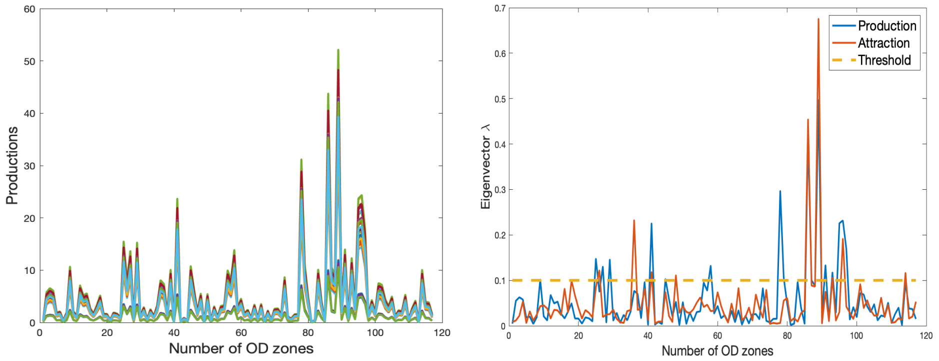
In the Santander case, we apply the PCA method of section 6.2.3 to construct a reduced solution space. This is indeed necessary, because the full matrix equality (6.16) could not be solved (was quite severely under determined). It turned out that only a single principal component was needed to explain $\approx 99\%$ of the variance in the production and attraction over a single day. The explanation is that the (calibrated) OD matrix for this network is created by scaling the entire OD matrix by time dependent factor $\tau(t)$ and thus the original dimension varies in the same direction. This is illustrated in Figure 6.9(a) which shows the production dynamics for all the time periods for a particular day and it is evident that only the magnitude varies for different periods while the demand distribution between zones remains unchanged. Thus, we consider only a single principal component to extract the reduced OD matrix. Note that deriving this single principal component is an emergent result of the method and provides us with confidence that the dimension reduction strategy is effective. However, in reality, OD matrices likely have varying structural similarity throughout the day (different $\tau(t)$ for each OD) and applying the method with such more complex dynamical OD matrices may provide additional evidence of the efficacy of the approach, or, possibly, limitations in applying a linear method such as PCA to sufficiently capture OD dynamics.



(a) *(b)*

Figure 6.8: Single instance of production and attraction prediction for 20 mins time horizon respectively

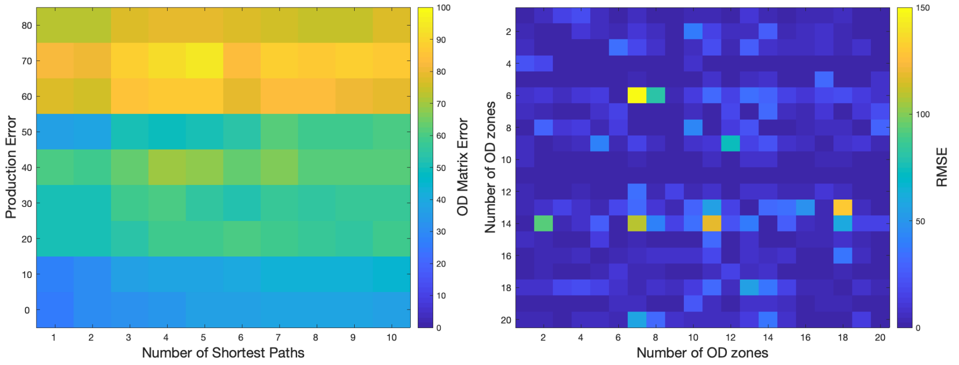
Figure 6.9(b) shows the eigenvector λ_1 of the production and attraction time series w.r.t. to the first principal component. Clearly, some of the dimensions (zones) have high correlation with the principal component. We use a cutoff threshold of 0.1 to select a total of 20 zones (as a consequence of the union of 16 production and 9 attraction zones, of which 5 overlap). This leads to a reduced OD space of 20×20 (400 unknowns)—a reduction in dimensionality of 97%(!) compared to the original 115×115 OD matrix (13225 unknowns).



(a) *(b)*

Figure 6.9: (a) Production dynamics for 48 time periods for a particular day (b) Eigenvector of production and attraction for the first principal component.

We use the 8 steps in section 6.2.3 to estimate the full OD matrix, with the reduced OD matrix used as upper bounds for 400 of in total 13,225 unknowns. Figure 6.10(a) presents the OD estimation error for the Santander network in the same way as in Figure 6.7(a). The optimal number of shortest paths for the Santander network is $N^* = 1$ which has an overall average error of 26% for $0-10\%$ production error. There are many possible reasons for this - the alternative routes do not provide much additional travel time gain under low or high traffic congestion; the alternative routes have large path overlap; the alternative routes might contain topologically inferior road stretches. Therefore, for a majority of the OD flows, the top shortest path is the only valid route. This is also the reason for the marginal influence of the number of shortest paths on the OD estimation error. From figure 6.10(a), the error only varies from 26% to 37% for $N^* = 1$ to $N^* = 10$. To illustrate the error distribution in the actual OD matrix, Figure 6.10(b) shows the RMSE of the reduced OD matrix for a single peak period for $N^* = 1$ (Due to the sparsity of the OD matrix, we use the reduced OD matrix to visualise the error in detail) The full OD matrix has a mean error of 22% with a mean RMSE value of 1.5 and a maximum of 150 vehicles/hour.



(a) *(b)*

Figure 6.10: (a) Mean OD matrix estimation error for the 110 scenarios of Santander network (b) OD estimation error for the reduced matrix for $N^* = 1$.

6.5 Conclusion and discussion

In this chapter we make an argument for a new data driven OD estimation method for cases where a supply pattern in the form of speeds and flows is available. We show that with these input data, an iterative traffic loading procedure with many assumptions on average (or individual) driving behavior; individual route choice trade-offs, and the type of equilibrium network traffic state that emerges *is not needed to solve the OD estimation problem*. The ingredients of this data driven method are (a) a method to estimate / predict production and attraction time series; (b) a method to compute the N^* shortest paths from each OD zone to the next; and (c) assumptions on the magnitude of N^*, and on the proportionality of pathflows between these origins and destinations, respectively. The latter constitutes the only behavioral assumption in our method, and we choose a proportionality factor that is inversely proportional to realized travel time, where we do incorporate a penalty for path overlap. For large networks, these ingredients may not be sufficient to find a unique OD matrix. We show that with a relatively simple and well-known feature space reduction technique (PCA: principal component analysis) we can derive additional constraints directly from the data to solve this underdeterminedness, and as a result a full OD matrix.

From the results on a small toy network and a large real network (of the Spanish city of Santander) we can draw the following conclusions. Most importantly, our simplified OD estimation method works, given a reasonable choice of N^* and a supply pattern (speed in the network) from which all route travel times between all ODs can be estimated. Second, we show that production and attraction time series can be predicted with a coarse 3D supply pattern that includes network topology. This makes the method scalable both in terms of network size and for different topologies, and for cases when the available data is coarse. Third, in the toy network case, we find that OD estimation quality is linearly related to production/attraction prediction quality. With a 10% P/A prediction error, the OD estimation error is about 20% with an additional 10% for every additional 10% P/A prediction error. In the full network we applied the PCA method and were indeed able to estimate the full OD matrix with a mean error of 22% and associated RMSE of 1.5 vehicles.

There are many paths to further explore, refine, and improve the properties of the method. Here we provide just a few main ideas. The method can be validated against OD matrices with different structural similarity for large networks to test the robustness of the approach. For large networks, we can refine the "8-step PCA procedure" proposed in computing also lower bounds (instead of just upper bounds). This would necessitate an iterative procedure, but may improve estimation accuracy considerably. Thirdly, a key point is that we chose a single assumption on the (average) number of shortest paths N^* for the entire network. Particularly for larger networks, literature (e.g. [215, 259]) suggests this assumption is likely too crude. The method, however, allows the analyst to diversify different assumptions on N^* for different (groups of) OD pairs, and similarly, add other factors than just travel time that may play a role here (costs, topological considerations, etc). Another avenue of future research is to identify parameters for the method such as the number of clusters, $N*$, for a given transport network, etc. without extensive sensitivity analysis. We believe this strongly depends on the topology of the network, choice set per OD patterns, etc. A fifth direction of further research relates to predicting the production and attraction patterns, which in the method is essentially an exchangeable building block. We see many research avenues here, because there are many aspects with which we may (systematically) vary. For example, what if we can actually (partially) observe production and attraction patterns? Perhaps we do not need supervised prediction models altogether? In this chapter, we use a neural network model, which implies a ground-truth data set is required (including production-attraction time series). However, there are straight-forward paths for alternative approaches. For example, by using and fusing demographic data with specific cordon counts, household surveys or movement traces. These latter sources could also be used as extra (soft) constraints on the spatial structure of the predicted production and attraction patterns and/or the estimated OD matrix.

Finally, the framework would fit very nicely in an online (DTA) modeling and management framework (with or without assumptions on equilibrium, response to ITS/information, etc), since the OD is now estimated *independently* from any disaggregated route choice or (micro/mes/macro) traffic assumptions. We are excited this idea turned out so fruitful, and hope it will open up many avenues for new research and applications within our field!

Chapter 7

Nationwide traffic predictions

Various methods have been proposed for representing the traffic states in the previous chapters. The shape-based method is proven to meaningfully represent the traffic states of highways. In this chapter, we further extend the shape-based approach to reveal regularity between daily network patterns and compare it against the partitioning-based approach. We also demonstrate the method on the entire Dutch highway network. Thus the method is adapted for maximum computational efficiency. Furthermore, this study provides a test bed to measure the performance of the proposed methods in terms of scalability for large-scale networks.

This chapter is based on the following paper that is currently under review:

Panchamy Krishnakumari, Oded Cats and Hans van Lint. "A compact and scalable representation of network traffic dynamics using shapes and its application.", submitted 2019-11-26.

7.1 Introduction

Understanding traffic dynamics has been one of the main research topics in the transportation sciences since the early 1950s. A large number of different traffic flow models has been developed by researchers for this purpose from the field of transportation, physics, and mathematics, as advanced techniques emerge and better data sources become available. There are many reviews of these modeling approaches, with different ways to categorize them based on their level of detail and aggregation [7, 260–262]. The focus of many of these studies is on describing and understanding the longitudinal dynamics of traffic on corridors. Some of the main approaches can be categorized as car-following models [8, 9], gas-kinetic models [10, 11], cellular automata [12], first-order traffic flow models [13] and higher-order traffic flow models [14, 15].

However, the literature on modeling network-level traffic dynamics is still limited. Simulating traffic on this coarser level offers many opportunities for better understanding and managing traffic flows in large urban networks. The idea of a macroscopic fundamental diagram (MFD) [16, 17], along with the empirical evidence of its existence [18–20], provided a breakthrough in modeling such network dynamics [93, 263]. It was found that details at the individual link level are not needed to describe the congestion dynamics of cities but can be instead defined based on homogeneous regions of a city. The main characteristic of traffic is that congestion propagates both in space and time with some finite speed and is spatially correlated to the adjacent roads. A key assumption in partitioning a network in homogeneous reservoirs is that also over time the partitioned regions remain homogeneous in terms of density and speed. However, this is in many cases not how network congestion dynamics work. Knoop et al (2015) for example show how congestion typically nucleates at specific bottlenecks [264]. Subsequently, the spatiotemporal extent of these congested areas may become larger (and later on smaller again) and over the course of time even move over a network. Such dynamics do suggest that it might be possible to identify homogeneous regions over space and time in networks (3D regions), in which speed and density remain approximately constant.

This is indeed what is found in [189], where time is incorporated for defining these homogeneous regions. They generalized the 2D partitioning efforts using *snakes* [188] to 3D and proposed using more computationally efficient data point clustering methods such as k-means, Growing Neural Gas (GNG) and DBSCAN, followed by a post-treatment methodology to ensure that the clusters are connected in space and time [189]. Moreover, these partitioning techniques was used to synthesize the network of Amsterdam over 35 days into 4 so-called consensual patterns with each 9 homogeneous 3D subnetworks and show that with these patterns the travel time of 84% of all trips between any OD in the network can be predicted with an error margin below 25% [62]. Thus, [62] demonstrates that investigating higher-level clustering over space and time can efficiently reduce dimensionality. However, the consensus partition is proved to be a NP-complete problem [265], which implies that the computational complexity, and thus time, of the algorithm increases rapidly as the size of the problem grows. Consequently, the consensus clustering becomes non-viable for large-scale urban networks.

To this end, we propose a new scalable approach, taking inspiration from human vision. The inspiration comes from the fact that humans use shapes, color and different physical characteristics for detecting faces and expressions. Instead of processing all possible shapes, we only need a general shape to detect a face. However, for face recognition, we might need more detailed physical attributes. Thus, depending on the application, we only need a general shape of the congestion propagation to distinguish between the different types of congestion. In [266], the variation of two archetype shapes, identified based on well-established literature [161], was proven to be sufficient for distinguishing between different types of congestion in corridor-level analysis with a 70% prediction accuracy rate. However, in network-level analysis, we do not have extensive knowledge of the different congestion patterns. Hence, in this work, we identify the recurrent shapes in network-wide congestion patterns and use these to distinguish between the traffic dynamics of a large-scale network. The main contribution of this work is a compact customized feature vector based on these identified shapes to define the daily network traffic state. This feature vector can easily be extended to include more context information without further increasing the computational complexity.

The chapter is organized as follows: section 7.2 describes the overall methodology including the data preparation, feature vector formulation, and its applications. In section 7.3, we outline how we assess the method. We demonstrate the method for two networks: a small network for which we compare the approach with the state-of-the-art consensus method and a larger network to demonstrate its scalability. We also briefly reiterate the consensus clustering approach proposed in [62] in section 7.3. The results are presented in section 7.4. We then offer conclusion and a discussion on further research avenues in section 7.5.

7.2 Method

Figure 7.1 gives an overview of our methodological framework. There are two main data sources: network in the form of a directed graph denoted by $G(V,E)$ and the supply patterns in the form of link speeds. The latter can be enriched by including additional data sources such as link flows, contextual data and weather information. The overall idea behind the framework stems from two assumptions: (1) that we can use physical features to extract information about the traffic dynamics and (2) that only higher abstraction levels of these features are needed to reveal the regularity between traffic dynamics of a large-scale urban network. However, the abstraction level greatly depends on the application. For example, we only need the information of a general shape, color and other high-level physical attributes for detecting faces whereas, for face recognition, we need much more detailed features for distinguishing between two different people. We extend this theory to traffic and hypothesize that, depending on the application, we only require high-level features such as the general shape of the congestion propagation to distinguish between different daily dynamics. To extract and use these high-level features, we need to prepare the network and traffic variable to incorporate the spatiotemporal relationship of traffic using 3D maps as indicated in figure 7.1(a). Thereafter, we extract the congestion shapes from these 3D daily maps. From these shapes, we identify and model the general shapes of the congestion and use this to construct a compact feature vector representation as shown in figure 7.1(b). Finally, we apply these feature vectors to reveal the regularity within the daily traffic patterns of an urban network as indicated in figure 7.1(c).

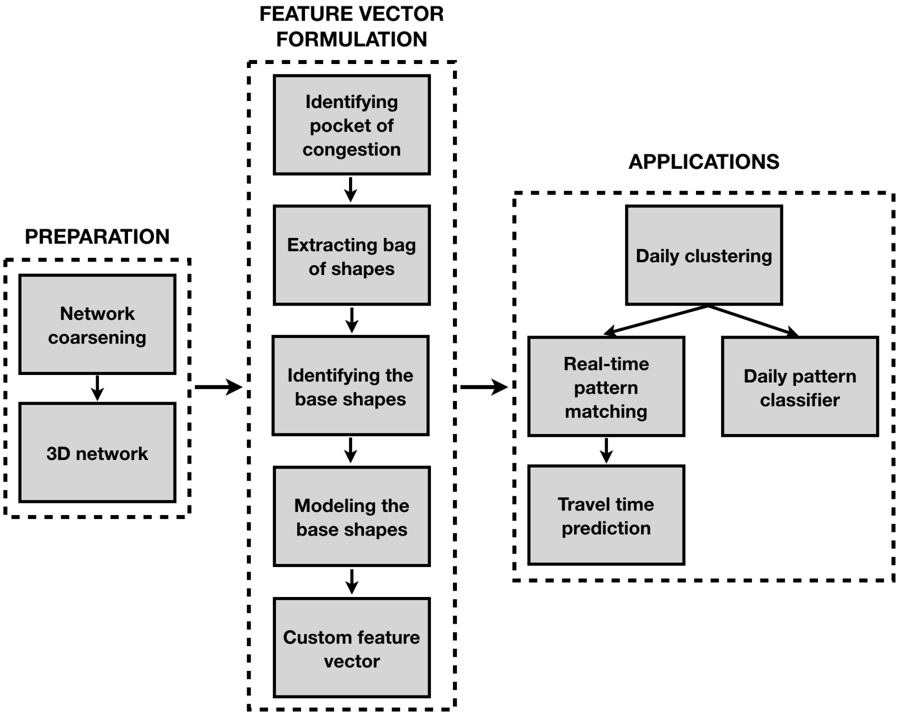


Figure 7.1: Methodological framework

7.2.1 Network and data preparation

We use the standard notation in graph theory to represent the transportation network. Here, the graph $G = (V,E)$ is a weighted directed graph where V is the set of nodes and E is the set of ordered pairs of edges or links. Each of the links is associated with a weight $w(u,v)$, where $(u,v) \in E$ and $u,v \in V$. In general, the weight can correspond to link length, width, flow or speed. However, since we aim to identify the pocket of congestion, hereby link weights correspond to speed.

Network coarsening

As part of network preparation, we first reduce the complexity of the network as this is the first roadblock in a network-wide study. Network complexity can sometimes determine the viability of a method for a particular city. There are various coarsening schemes to reduce the complexity of graphs from the field of experimental algorithms [48, 68, 80]. A recent study focused on transport network offers a coarsening scheme more tailored for transportation network [267]. The general idea is that, given graph $G(V,E)$ with n nodes, a more compact representation of the original graph, $G'(V',E')$, with a smaller number of nodes can be found which preserves its topological properties such as shortest path and network length. This is achieved by collapsing together nodes that have links attached to it with similar weights. A variance threshold is used to control the range of this similarity. A threshold of 0 implies that only links with the same weight are collapsed together. For more details on the coarsening algorithm, we refer the interested reader to [267]. The authors also offer an open-source implementation of the algorithm [268]. In this chapter, we use the link speed for collapsing the node and a variance threshold of 0 is used to preserve the data as it is and remove any data aliasing due to collapsing nodes connected to links with varying speed, which leads to a smoothing of the link speed in the coarsened graph.

3D network

Given the coarsened network $G'(V',E')$ and the associated link speed, we can incorporate another important characteristic of traffic - its temporal dynamics. Thus, instead of static graphs $G'(V',E')$, we need to construct time-dependent graphs $G'_t(V',E')$ to represent the network traffic dynamics in both space and time. The link speed is inherently time-dependent and can be represented as $w_t(u,v)$, where $u,v \in E'$. The resolution of the time slice t depends on the aggregation of the link speed. To construct this 3D map, directed weighted graphs for different time t are stacked together with unidirectional virtual links between the time slices, which only goes forward in time and not backward as shown in figure 7.2. The result is a very compact representation of the traffic dynamics in space and time.

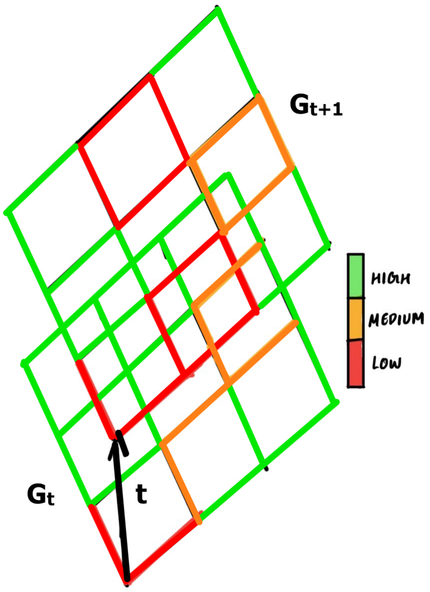


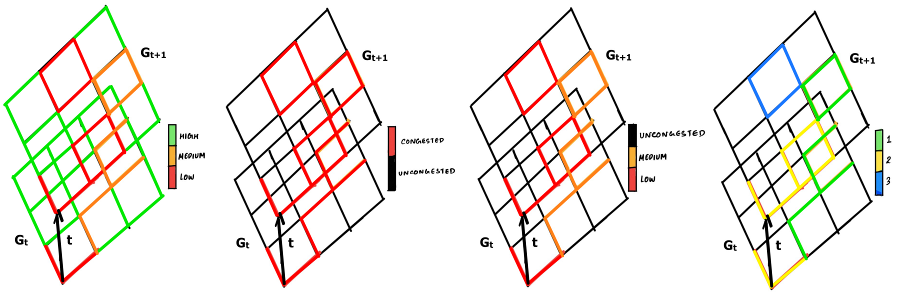
Figure 7.2: 3D network with unidirectional virtual links connecting the 2D networks and representing time t where different colors display the edge weight - link speed - ranging from low to high speed.

7.2.2 Feature vector formulation

Customized high-level physical features coupled with traffic variables and other contextual information have proven to be successful in revealing regularity in corridor-based traffic congestion patterns [266]. In this work, we extract the congestion shapes, identify the recurrent shapes and use this knowledge to build a feature vector that encompasses the network dynamics of an entire day of a large scale urban network. An overview of the steps for extracting these high-level features is given in figure 7.1(b).

Identifying pockets of congestion using conditional 3D partitioning

In traffic networks, congestion propagates over space and time. Our main idea is to use these congestion regions to represent the dynamics of a specific day. For this, we first identify low speed regions from the weighted 3D network $G'_t(V',E')$. At any given time, most likely, a large portion of the network will not be congested (except in the case of severe gridlock). Therefore, to reduce the dimensionality of the problem, we do not consider the free-flow regions to define the network dynamics. To this end, we use a speed threshold, v^0, to differentiate between congested and free-flow regions. In [266], a threshold of 65 km/hr was used as the threshold for highways. So, any link with a speed lower than 65 km/hr will be considered as a congestion region. Based on the case study area, this threshold can be adjusted. This value, which heavily depends on the case study area, can be made relevant for any road type by using a relative link speed ratio instead of an absolute link speed. The speed ratio of a link at a given time t is the ratio between the speed drop (difference between the speed of the link at time t and its speed limit) and the speed limit of the link. Thus, a threshold of 0.5 on the speed ratio implies a speed drop of 50 % relative to its speed limit, which is significant and can be objectively categorized as a congestion region. However, this requires additional data regarding the speed limit of the links. Thus, depending on data availability, we can use a threshold v^0 on either speed or speed ratio to distinguish between congested and free-flow conditions as illustrated in figure 7.3(b).



(a) (b) (c) (d)

Figure 7.3: Steps for identifying pockets of congestion from 3D speed maps (a) 3D network with colors representing link speed ranging from low to high speed (b) links categorised as congested and uncongested (free-flow) (c) congested links clustered into different unconnected zones based on their speed (d) extract the three connected zones which corresponds to the three pockets of congestion.

Given that we have identified the free-flow and congested regions of the 3D network, we can now identify the pockets of congestion in the network. We define a *pocket of congestion* as a connected congestion region in the network, which is most probably, caused by a common bottleneck(s) or incident [264]. For this, we can partition the heterogeneous congested regions in the network into homogeneous congestion zones. 3D network partitioning has been studied before in [62, 189], which proposes different 3D clustering techniques and post-treatment methodology for partitioning the entire network into homogeneous connected zones. However, they aim to partition the entire network to extract N connected zones from unconnected clusters, whereas we aim to find all the connected congested zones. Hence, we do not require any optimization in the post-treatment methodology to make sure congestion regions are connected. This improves the efficiency of the method exponentially. To this end, we extract the pocket of congestion by:

- Clustering the spatiotemporal link speeds in the congested regions.
- Connecting the resultant clusters in space and time.
- Each connected 3D cluster is represented by a single traffic variable, which can either be the mean speed or mean speed ratio of the 3D cluster depending on data availability.

The vectorised link speed $w_t(u,v)$, for all $u,v \in E'$ and for all t on a given day, is clustered into N unconnected clusters using k-means [269]. This clustering identifies different types of congestion within a network, for example, light, medium or heavy congestion as illustrated in figure 7.3(c). Since k-means does not incorporate spatial connectivity when clustering into different groups, we incorporate a simplified post-treatment methodology to extract zones that are connected in both space and time. This is done by finding connected components (CC) from the resultant k-mean clusters using depth-first search algorithm [85]. A connected component is a sub-graph in which any two nodes are connected to each other by at least one path. Given the N clusters from the k-means clustering, there might be more than one CC for each cluster as shown in figure 7.3(c). All CCs within each cluster are identified and each of the identified CC is assigned a new cluster number. Thus, each of the identified CC is a pocket of congestion represented by a single variable - the average speed of the links that belongs to that cluster. The result is a very compact representation of the same spatio-temporal dynamics contained in the 3D network, but now in the form of 3D pockets of congestion represented by zone variables (average speeds). An example of the identified pocket of congestion is shown in figure 7.3(d).

Extracting bag of shapes

We extract 3D shapes of each pocket of congestion in a daily pattern by creating a convex hull around each of these congested zones. Each pocket of congestion in a daily pattern is represented by 4 variables (x,y,t,s) where x and y are the geographical coordinates of the locations inside the zone, t is the time extent of that zone and s is the cluster number of the zone. The convex hull of a zone represents the boundary of the zone as shown in Figure 7.4(a). Because of the complexity of a 3D shape, we project the 3D shape on a 2D plane by creating a convex hull around (x,t) and (y,t) to create two 2D shapes for each zone.

Thus, if there are p 3D pockets of congestion in a daily pattern, there will be $2p$ 2D shapes. By collecting the $2p$ shapes from all the daily patterns, we can create a comprehensive *bag of shapes* that represents the entire data space as shown in Figure 7.4(b).

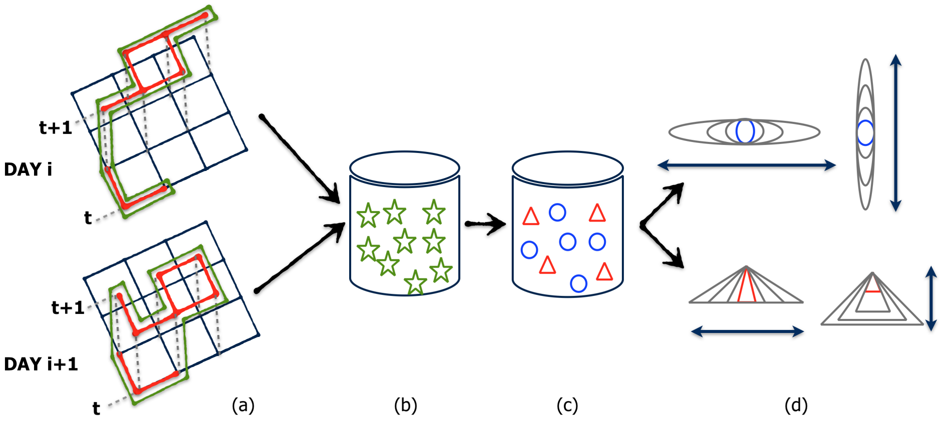


Figure 7.4: Shape-based clustering (a) 3D convex hull from zones of all days (b) Bag of shapes (each star represents a 2D shape) (c) Clustering into base shapes (d) Base shape models where blue and red shapes correspond to mean shape and the gray corresponds to variations of the shape along their principle components.

Some of the congestion pockets are relatively small, i.e. the congestion spans only a single link or over a single time period. We can remove such small congestion regions by setting a lower bound for the area of the shape using a^0 so that we only consider congestion of significant spatial and/or temporal persistence.

Identifying the base shapes

The bag of shapes can increase rapidly with the increasing availability of data from more days. In [266], archetype shapes, also known as base shapes, have been identified for corridor-based traffic patterns based on literature to solve this problem. *Base shapes* are the generic shapes that recurrently prevail in the data, whose variations can be used to represent other shapes in the data. For network-wide patterns, the shapes differ based on the network structure and underlying supply and demand conditions and hence, a straight forward manual archetype identification is not possible. Therefore, in this work, we introduce a data-driven approach to identify base shapes. This is done by defining a shape similarity index (SSI) for a pair of n-dimensional shapes x_1 and x_2 as:

$$SSI(x_1,x_2) = \sum_x \sum_y ||\hat{x_1} - \hat{x_2}|| \tag{7.1}$$

where $\hat{x_1}$ and $\hat{x_2}$ are the superimposed shapes respectively, x and y the 2D coordinate space of the shapes. The shapes are superimposed on each other using Procrustes analysis [270] which translates, rotates and scales the shapes optimally to minimize the sum of squared distances, also known as Procrustes distance. The shape similarity index of all shape pairs in the bag of shapes is found. Thus, if there are $2p$ shapes in the bag of shapes, each shape is represented by $2p$ Procrustes distances and thus creating a $2p \times 2p$ distance matrix for all the shapes. To estimate how many base shapes are required to approximate any shape in the bag of shapes, Principal Component Analysis (PCA) is applied to the distance matrix [271]. PCA gives the Eigenvectors and Eigenvalues of the distance matrix and we can compute how many principal components are required to explain $v\%$ of the variances in the shapes. In this work, we set v to 95%. The number of principal components, denoted by k, needed to explain 95% variance is used as the number of archetype shapes needed to represent the entire bag of shapes. Thus, the entire bag of shapes is clustered into k archetype shapes using k-means clustering as illustrated in Figure 7.4(c). Consequently, we can create a separate bag of shapes for each archetype shapes based on the clustering resulting in k bag of shapes for k archetype shapes.

Modeling the base shapes

To model the variances within an archetype shape, an Active Shape Model (ASM) is constructed for each base shape. The ASM is introduced in [170] based on the principle that deformations of a shape can be represented by a so-called Statistical Shape Model (SSM) which contains all the parameters that are needed to define that shape. The SSM comprises of a mean shape and variations of that shape, analogously to a scalar value and statistical variations (variance) around that value. An example of mean shape and a couple of variations, also known as SSM components is illustrated in Figure 7.4(d). Initially, a set of landmarks/points in the new shape is defined, after which the shape defined by these landmarks is deformed to provide the best fit available within the SSM. The deformation is based on finding correspondences between the new shape and the shape defined by different SSM components and iteratively minimizing a cost function for the fit. The allowed degree of deformation is constrained by the variations defined in the SSM. If the SSM includes large variances, large deformations are possible and vice versa. A more detailed explanation of ASM can be found in [170].

An overview of the ASM method for building a shape model for the archetype shapes are given below:

1. To build a shape model, all the shapes need to have the same number of dimensions. For this, we downsample all the shapes in the archetype shapes to the minimum number of dimensions in the corresponding bag.

2. The downsampled shapes are aligned to the first shape in the bag as an initialization step using Procrustes analysis. An initial mean shape is generated from the aligned shapes.

3. The downsampled shapes are realigned to the mean shape and a new mean shape is generated from the aligned shapes. Repeat this step until convergence to generate the SSM mean shape.

4. Apply PCA on the final aligned shapes to compute the Eigenvectors and Eigenvalues which constitutes the SSM components and their variances respectively. Thus,

any shape x can be approximated using the SSM mean shape $\bar{x}$, SSM principal components P and the variances b that deforms the shapes according to the components as

$$x \approx \bar{x} + Pb \tag{7.2}$$

5. Repeat the steps for all archetype shapes.

Thus, we identify the base shapes from the bag of shapes and build an SSM model for each of them.

Custom feature vector

We now have the necessary ingredients to define a compact yet insightful feature vector to represent the network traffic dynamics of a daily pattern. Given the $2p$ 2D shapes of the daily patterns, we can use the shape model to categorize these shapes into the base shapes to create a multidimensional feature vector $(n_i, n_{i+1}, n_{i+2}, ...n_{i+m})$; where n_i represents the number of occurrences of base shape i in that daily pattern and m is the total number of base shapes identified. The 2D shapes are categorized into i^{th} base shape by minimizing the ASM fitting error between the shape models of all the base shapes. The ASM fitting process is as follows:

1. Given a new shape Y', we downsample the shape to the same dimension as the mean shape $\bar{x}$
2. Assign shape parameter b as 0.
3. Initialise the model point x based on b using $x = \bar{x} + Pb$
4. Align the new shape Y' to x using Procrustes analysis.
5. Update b in (7.2) to match the new shape Y' by solving $Pb = Y' - \bar{x}$
6. Compute the error metric by finding the difference between the mean shape $\bar{x}$ and the ASM fitted shape Y'
7. Repeat steps 3 to 6 until convergence. The error metric is used as the convergence criteria.

The new shape is fitted to all the base shapes and the corresponding fitting error of each of the base shape is computed. The base shape that minimizes the fitting error is chosen as the base shape that the new shape belongs to. Additionally, we also use the traffic variables within the shapes (average speed) to define the traffic state of that network. Instead of using the average speed, we define k clusters which represent k ranges of speed in order to be consistent between different days and also to interpret the feature vector easily. To be consistent, we assign k as the number of clusters N used for clustering the congested regions into different speed regions in section 7.2.2. The final feature vector that fully defines the dynamics of a day based on the congested regions is defined as follows:

$$FV_{day} = [n_{i+1}, ...n_{i+k}, ...n_{i+m+1}, ...n_{i+m+k}, s_1, ...s_k, v_1, ...v_k] \tag{7.3}$$

where k is the number of speed clusters, m is the number of base shapes, n_{i+1} is the number of occurrences of base shape i in cluster 1 in a 3D pattern of day, s_1 is the average speed within speed cluster 1 and v_1 is the variance of speed within cluster 1. This feature vector is computed for all days.

7.2.3 Applications

There are different applications for representing traffic data using these compact feature vectors. We reveal the regularity between different days and use these insights to extract the essence of these days into a set of representative 3D speed maps. Here, we present two such applications. The first application is using these 3D maps for short-term predictions of congestion propagation, which can be used for the travel time predictions. The travel time prediction is used to evaluate the performance of our method against the benchmark method. The second application is to reveal the representativeness of these feature vectors in revealing the day-to-day regularity by investigating the classification accuracy of the identified daily clusters. This application is also used to evaluate the scalability of our approach. In the following we first describe the daily clustering procedure before elaborating on the two aforementioned applications.

Daily clustering

K-means is used to cluster the days, represented using the custom feature vector, into M classes to reveal regularity between days. There are different approaches to find a representative 3D network for each class. We used the centroid of the 3D speeds maps of the days that belong to each of the classes as follows:

$$\lambda_m = \frac{1}{n}\sum_{i=1}^{n}\lambda_i \tag{7.4}$$

where λ_i is the single ordered vector of traffic observations in the 3D network whose values can either be the link speed or speed ratio, n is the number of days in class m. Thus, λ_m is calculated for all classes and we obtain a 3D map for all classes.

Real-time pattern matching and travel time prediction

Given the current speed observation, we match the observed speed observations to these representative 3D maps to predict to which class is the test day resembles most. This is done by matching the current traffic state to the 3D representative models to predict the next traffic state. For the matching, we need to find the class that best fits the new data. For this, depending on the size of the available real-time speed data, the model is truncated and a similarity matrix is computed between the real-time speed and truncated model of each class. The model that maximizes the similarity is chosen as the predicted speed map of the new data. Since the model of each class λ_m is mean speed values, we use the Euclidean distance between the truncated model and the current speed for computing the similarity matrix. The complete 3D model of the matched class is the predicted traffic state of the test data which can be used for travel time predictions. We evaluate the quality of the clustering and the matching process using this prediction error. For the evaluation, we conduct leave-one-out validation. For a given day, the following steps are performed without considering that test day:

- cluster the feature vector of all days other than the test day into M classes
- create the mean model of these classes based on the shape-based feature vector
- fit the speed profile of the day under consideration to the mean model to predict the future speed profile
- the predicted speed profile model is used to generate travel time for predefined routes

These steps are repeated for each day in the dataset. The prediction accuracy is evaluated by computing the travel time of the predefined routes through the fitted speed profiles and comparing the results to benchmark methods and ground truth travel time. The travel time of the predefined routes is computed every t minutes in the 3D map so as to get a more representative sample. An exhaustive analysis of the travel time error is done to evaluate the prediction using basic performance indicators such as Mean Absolute Percentage Error (MAPE) and the Root Mean Squared Error (RMSE).

Daily pattern classifier

The aim of this application is to show that the clusters of daily patterns extracted based on the custom feature vector encompasses the entire data set. This is done by classifying a new network traffic pattern into one of the M predefined daily pattern clusters and investigate the classification accuracy. The feature vectors of the entire data set is clustered into M classes and this class labels is used as the ground truth. For estimating the classification accuracy, we used a leave-one-out strategy and the following steps are performed for a given test day:

- Cluster the feature vector of the days other than the test day into M classes using k-means.
- Train a simple k-means classifier with input as the feature vectors and the output as the estimated class number from the previous step.
- Estimate the class label of the test day by classifying the feature vector of the test day into M classes using the trained classifier.

The classification accuracy is measured as:

$$accuracy = \frac{number\ of\ correctly\ labelled\ days}{total\ number\ of\ days} * 100\ \% \tag{7.5}$$

We also use a confusion matrix [175] to investigate the performance of each class in order to get insights into these classes.

7.3 Experimental Setup

7.3.1 Data

We demonstrate our methodology on two network. First, a relatively small network of the major streets within the city of Amsterdam (excluding the freeways), which consists of 7512 links and 6528 nodes. The network data are obtained from OpenStreetMaps. Second, our method is applied for the entire Dutch highway network with 131 994 links and 116 305 nodes, which is openly accessible from the Dutch road authority through NWB [272]. These case studies show the viability of our approach for both small-scale and large-scale networks as well as for both urban and highway road network.

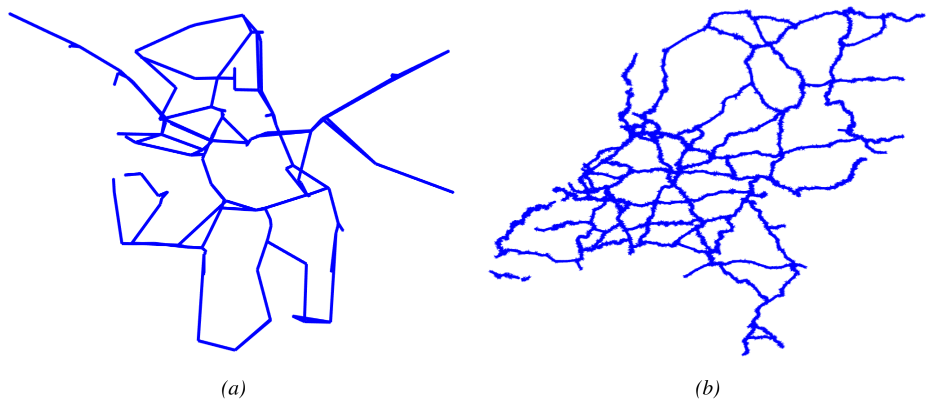
The traffic information available for the city of Amsterdam is comprised of mean speeds available every 10 minutes between 7 am and 3 pm ($\approx$ 8 hours) for all links during 35 days from 23 February 2015 to 4 April 2015 (except Sundays). This information is derived from license plate recognition system at different critical points of the network. The methodology to derive link speed data from trajectory data; creating the mapped network; and reconstruct missing data can be found in [189]. For the entire Dutch highway network, there are more than 10 000 detectors across the network which are placed approximately 500 meters apart. Most of these detectors collect both speeds and flow every 1 minute. We use the well-known adaptive smoothing method introduced by Treiber and Helbing [156] to fill in the missing speed information between detectors which results in filtered data every 30s. Since the speed limit of the highway network is also available, we use the speed ratio instead of absolute speed to represent the dynamics of the highway network. We collected these data for 45 days from 1 June 2018, to 15 July 2018, between 03:00 and 23:59 ($\approx$ 21 hours).

For both case studies, we use coarsening to reduce network complexity. The urban network of Amsterdam is coarsened using the link speed at 16:00 (peak period) as the edge weights. The coarsened network contains 411 links. However, some of these links have missing data for some days. By taking the intersection of all links that have data for the 35 days, the final coarsened network contains 208 links and 214 nodes (a complexity reduction of 97%) forms a single component as shown in figure 7.5(a). Each day is represented by a 3D network of dimension 208×48 (number of links $\times$ time periods). For the entire Dutch highway network, we use the entire speed ratio vector of the link as the link weight for coarsening. The final coarsened network contains 38 662 links and 34 655 nodes with 14 connected components as shown in figure 7.5(b), which is a $\approx 71\%$ reduction in the number of links with a 0 variance threshold. Thus, each day is represented by a 3D network of dimension 38662×2519, which implies $\approx$ 97 million data points to represent a single day.

7.3.2 Evaluation

To evaluate our method, we use the current state-of-the-art in network-wide analysis - consensus method, which was found able to synthesis 35 days of data into 4 consensual patterns [62]. We briefly explain the consensus method here, but for further details refer to [62]. For extracting regularity between the daily patterns using the consensus method, there are several pre-processing steps:

- Constructing 3D network describing link-based traffic states



(a) (b)

Figure 7.5: Coarsened network of (a) Amsterdam with a single connected component (b) Entire dutch highway network with 14 connected components

- Constructing 3D patterns describing zone-based traffic states
- Clustering 3D daily patterns using a consensus approach
- Real-time pattern matching for short-term travel time predictions.

The first step is the same as our data preparation module. The second step includes k-means clustering followed by a full-fledged post-treatment methodology which involves dividing the entire 3D network into k zones, which has a complexity of $O(N!)$ where N is the number of connected components in the 3D network after the initial clustering whereas our identification of pocket of congestion only has a complexity of $O(N)$.

The third step involves clustering the daily 3D patterns representing the zones. In [62], a similarity measure known as normalized mutual information (NMI) has been used for the clustering. NMI has been extensively used for assessing the similarity between two clustering results [273]. NMI for comparing two 3D zones π_i and π_j of two different days is denoted by $NMI(\pi_i, \pi_j)$. π_i is a single ordered vector of all observations in the 3D zones whose values are the cluster ID. The pairwise NMI for all days is calculated and the Ncut algorithm [274] is used on this similarity measure to cluster the days into different classes. For finding a representative 3D median partition (consensus partition) for each class, the following two steps are performed:

- A representative partition m is defined for each class that maximizes the total similarity TS with all the other days belonging to the same cluster.

$$TS = \sum_{k=1}^{a} NMI(\pi_m, \pi_k) \tag{7.6}$$

  where a is the number of days in a given cluster, π_m and π_k is the representative partition and partition of every other day in the same cluster, respectively.

- Refining the partition m by randomly changing the label of the partition m and investigating whether that improves the TS. This is done using one element move (OEM) algorithm [275].

The details on clustering the individual days and creating the consensus partition are given in [62].

The final step is the real-time pattern matching where the consensus model is matched with the current speed observation. In order to match the truncated consensus partition of each class to the new speed values, the following steps are applied:

- compute the speed vector to define the consensus partition of a class as follows:

$$\hat{s}(m,z) = \frac{1}{M*N*T} \sum_{i=1}^{M} \sum_{j=1}^{N} \sum_{k=1}^{T} s(i,j,k) \tag{7.7}$$

  where $\hat{s}(m,z)$ is the mean speed of zone z for the consensus partition of class m, M is the number of historical days in class m, N is the links that belongs to that zone, T is the truncated time period and $s(i,j,k)$ is the speed of link j at time period k for the i^{th} day in class m.

- compute a new speed vector of the current traffic state as follows:

$$\bar{s}(m,z) = \frac{1}{N*T} \sum_{i=1}^{N} \sum_{j=1}^{T} s_{test}(i,j) \tag{7.8}$$

  where $\bar{s}(m,z)$ is the mean speed of zone z for the consensus partition of class m for the new test day and $s_{test}(i,j)$ is the speed of link i at time j.

- compute the euclidean distance between $\hat{s}(m,z)$ and $\bar{s}(m,z)$ for all zones and all classes.

Thus, the class that fits the new data is the m that minimizes the Euclidean distance for all zones z.

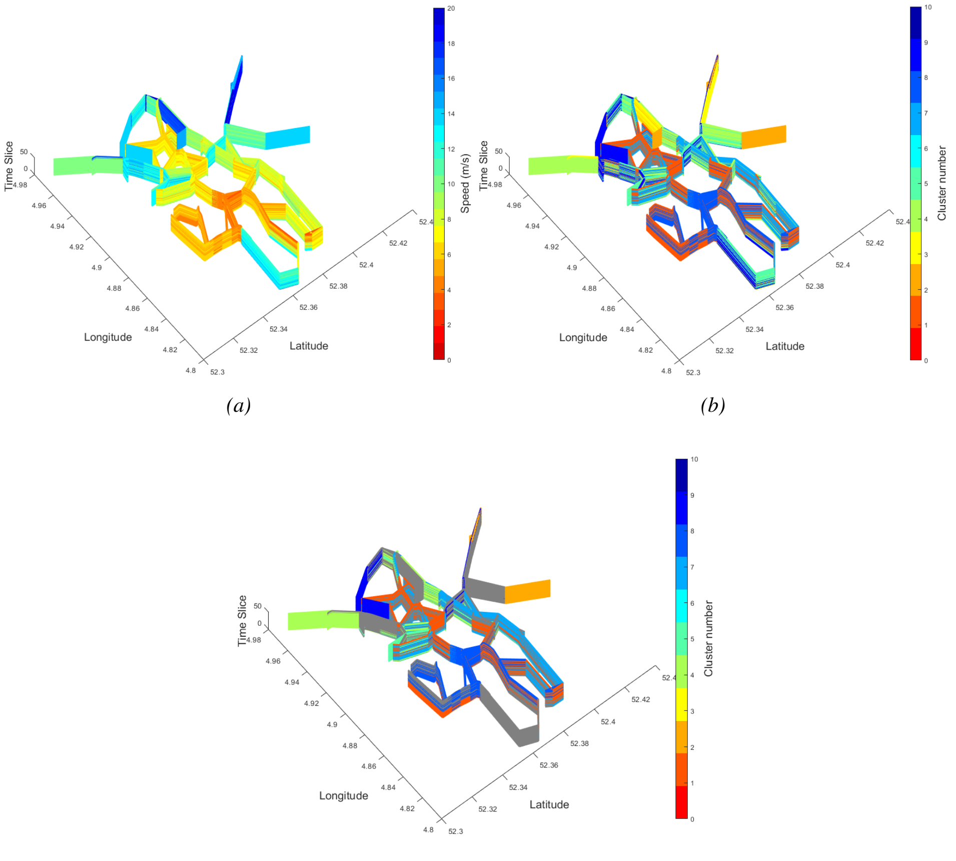
We conduct a comparison study only for the urban network of Amsterdam as processing the data for the entire Dutch highway network using consensus partition will lead to an unreasonable amount of running time ($\approx$ months). This is due to two reasons - the post-treatment methodology has a complexity of $O(N!)$ and the consensus partitioning was proved to be a NP-complete problem. Consequently, we demonstrate the first application on the Amsterdam network and the second application on the entire dutch network.

7.4 Results

In this section, we present the results of the shape-based clustering for the two case study networks. We demonstrate the performance of the method by comparing the travel time prediction with the consensus method. Then, we apply the method on the entire Dutch highway to reveal the regularity of network patterns.

7.4.1 Application 1 - Comparison with consensus models

The Amsterdam network is used for comparing our approach to the consensus method. An example of a 3D speed map of Amsterdam is shown in figure 7.6(a). To allow for a meaningful comparison of the two approaches, we used the same number of zones to partition the network, which is set to 10. We use 10 3D zones to describe an individual day of 8 hours for the consensus method. Since we do not have information about the road type and speed limit, we cannot set an informative v^0 to distinguish between congested and uncongested regions. Thus, we consider all the links as congested, which is the worst-case scenario in terms of computational time for the proposed method. The congested links are partitioned using k-means clustering with $k = 10$ as shown in figure 7.6(b). The pockets of congestion are identified from these unconnected clusters. The top 10 connected pockets of congestion for a given day are shown in figure 7.6(c). The top 10 refers to the largest



(a)

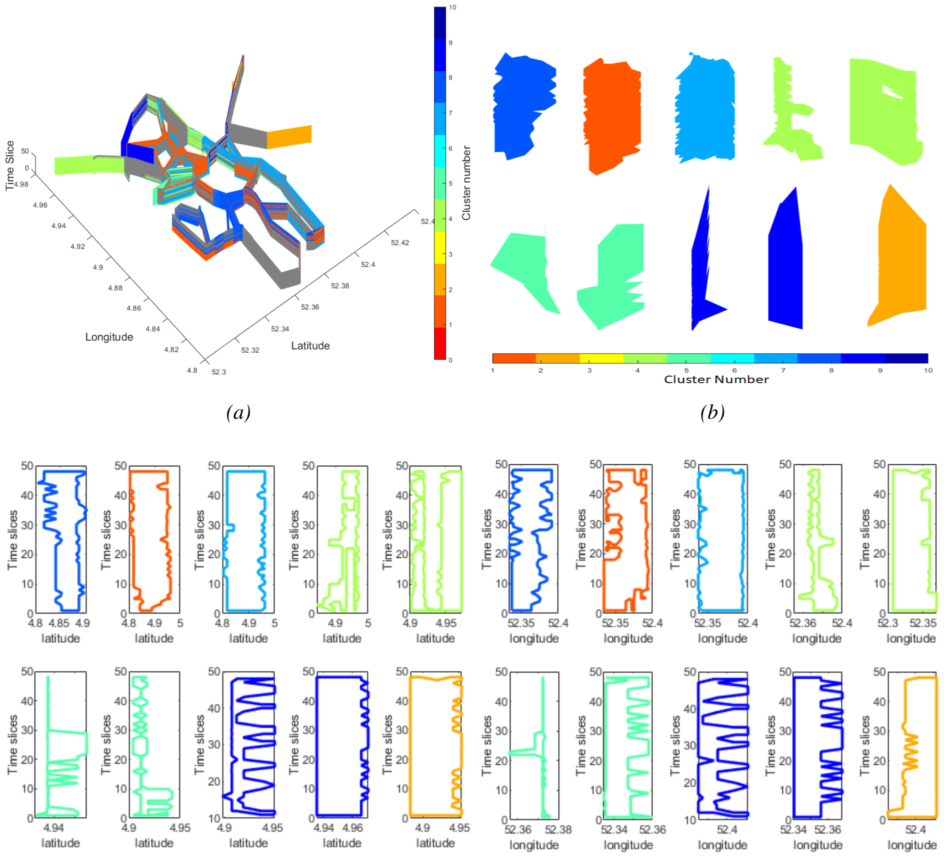
(b)

(c)

Figure 7.6: Identifying pockets of congestion (a) 3D speed map (red implies low speed, blue high speed) (b) k-means clustering to extract unconnected clusters (c) Extract connected clusters to identify the top 10 pocket of congestion.

connected components in terms of size with the lowest speeds.

The 3D shapes are extracted from the pocket of congestion as shown in figure 7.7(b). The projected 2D shapes with latitude × time and longitude × time are shown in Figure 7.7(c) and (d), respectively. From the figure, we can see that there is regularity within the 2D shapes. Put simply, many shapes have similar geometries. There are a lot of recurrent shapes that can easily be described by a shape model rather than having separate shape models.



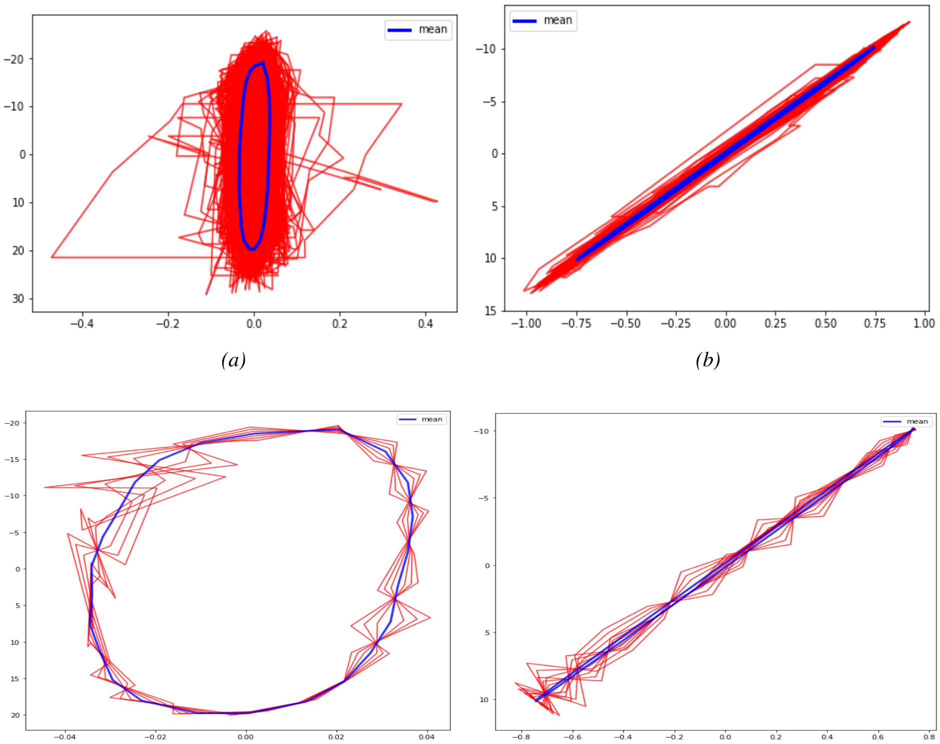
(a) *(b)*

(c) *(d)*

Figure 7.7: Representing the top pockets of congestion for a given day as 2D shapes for the Amsterdam case study (a) top 10 pockets of congestion for that day (b) corresponding 10 3D shapes of the pockets of congestion (c) projected 2D shape representing the boundary between latitude and time (d) projected 2D shape representing the boundary between longitude and time.

The 2D shapes from the 35 days are collected together to form the bag of shapes for Amsterdam. It was found that 95% of the shape variations in the bag of shapes can be represented by 2 base shapes and its shape models. Based on the similarity distance matrix, we clustered the bag of shapes into the two base shapes leading to the first base shape with

258 shapes and second base shape with 606 shapes. The shape model of each base shape contains a mean shape and the corresponding SSM components that can explain 95% of the variance within the shape. Base shapes 1 and 2 have 10 and 11 principal components, respectively. The corresponding mean shape and one of the SSM components of each base shape are shown in Figure 7.8.



(a)

(b)

(c)

(d)

Figure 7.8: Two base shapes of the Amsterdam case study derived from the projected 2D shapes with the mean shape highlighted in blue. The red shapes are the aligned bag of shapes in each base shape (a) and (b). (c) and (d) show one of the principal components of base shape 1 and 2, respectively.

The shape models are used to construct the feature vector which along with additional traffic variable information is used for clustering the daily patterns into different groups. Since k is 10 and the number of base shapes is 2 for Amsterdam use case, the feature vector dimension is 40. Thus, a single day can be represented using 40 values instead of 9984 (number of links × time resolution), leading to a dimensionality reduction of more than 99%. These feature vectors can be clustered to reveal regularity between days. In [62], 4 consensus models are used to represent the 35 days. In this chapter, we, therefore, use the same number of clusters to conduct a fair comparison for travel time prediction. Thus, the feature vectors are clustered into 4 classes and the mean speed model of each class using the shape-based approach is shown in Figure 7.9.

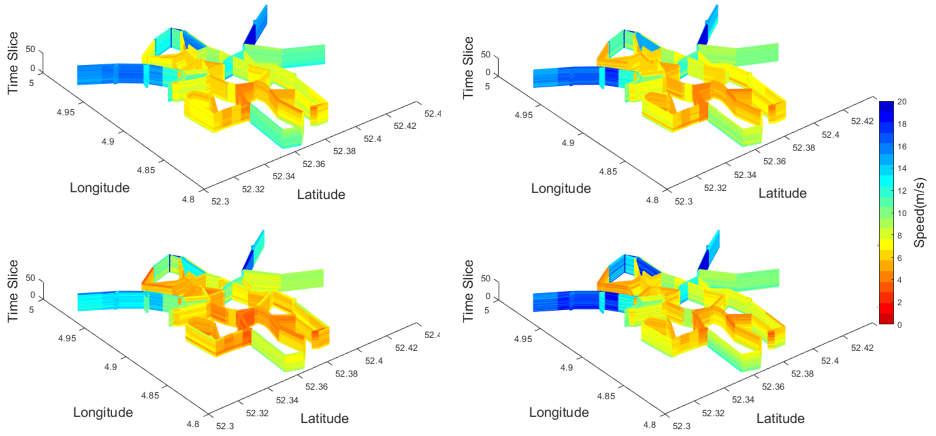


Figure 7.9: Four shape-based speed models for Amsterdam

The results of the travel time prediction based on 4 consensus and shape-based models are presented here for the Amsterdam case. We conducted a leave-one-out strategy for the travel time prediction, so that 34 days are used for training the classifier of the daily patterns and the remaining day is used for testing. This is repeated for all 35 days. For validating the travel time accuracy, 10 routes are randomly generated from the network. The ground truth is the travel time computed from the observed link speed from the 3D speed maps and we compare this against our results and against the travel time computed using the consensus models. Figure 7.10 shows the 10 routes and an example result for a given day with the travel time estimated using the observed ground-truth speed, our shape method, and the benchmark consensus method.

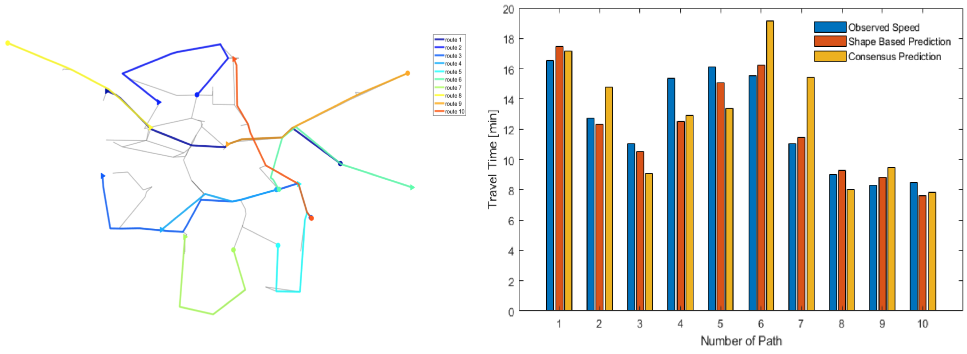


Figure 7.10: The 10 routes used for validation with their corresponding travel time estimated from ground truth observed speed, predicted using the proposed shape-based method and based on the consensus method for a particular instance as an illustration.

It can be seen that half of the travel times are underestimated for the routes, even though the error is marginal. This is also evident from Figure 7.11(a) which shows a more detailed error distribution of the 10 routes for all 35 days. 64% of the travel time error is because of underestimated travel time (positive error time) and thus the method seems to be able to encompass congestion dynamics slightly better than free flow conditions. This is acceptable as it is more important to have better predictions in case of an incident or traffic jam.

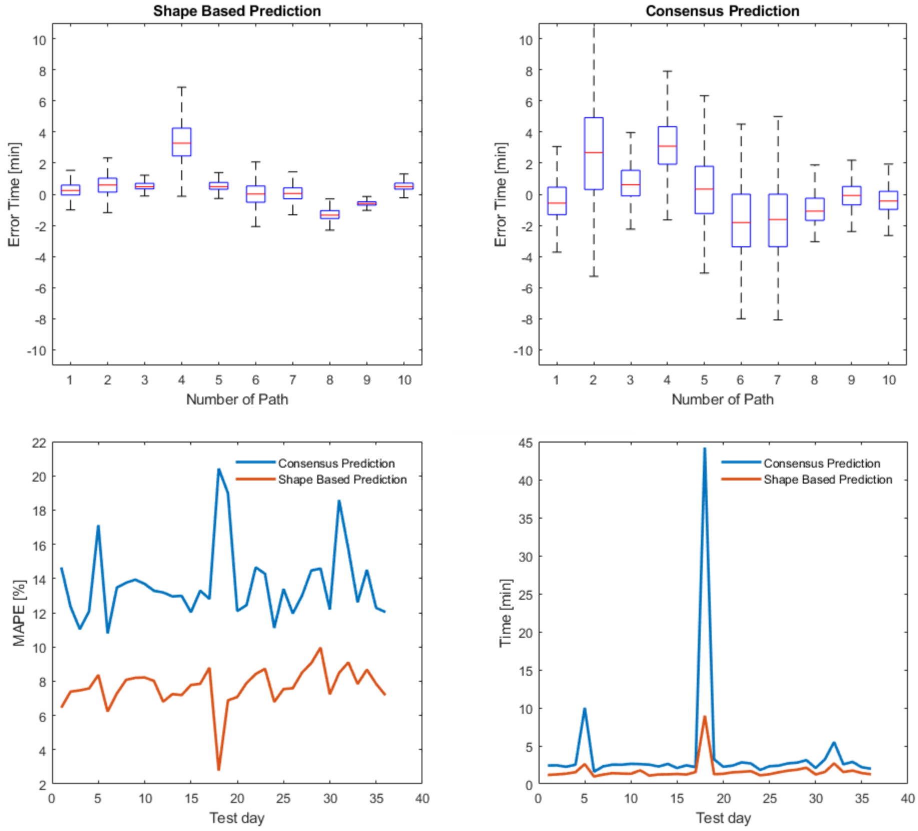


Figure 7.11: Leave-one-out validation results for travel time prediction using consensus models and shape based speed models for 35 days. The distribution of travel time error for the 10 routes within the 35 days for shape based (a) and consensus (b) models. MAPE comparison results (c) and RMSE comparison results (d).

Figure 7.11(c) and (d) show the MAPE and RMSE result of the 10 routes for the 35 days, respectively. The shape-based method has a mean prediction error of approximately 8% compared to 14% for consensus which is an improvement of around 44%. 95% of the travel time prediction has less than 9% error for the shape-based approach compared to 19% for consensus. Some of the days perform worse than others. There are three well-defined peaks in Figure 7.11(c) and (d). This is probably because of non-recurrent incidents or special events that occurred on these days. A well-defined incident database can be used to provide more insight into these irregularities in the traffic dynamics which can ultimately be used for improving travel time predictions under such conditions.

As for the computational time for the shape-based and consensus methods, the steps that differ between the two methods are the post-treatment and building the models themselves. The post-treatment for all 35 days took approximately 1 second for the shape-based and 8 seconds for the consensus method. The computation time for building the consensus model is approximately 6 minutes whereas the shape-based method is less than 4 minutes. The computational improvement of the shape-based method seems insignificant for the Amsterdam use case. However, the computation times for both modules - building and post-treatment - increase rapidly with an increase in network size for the consensus method as we will illustrate in the next section.

7.4.2 Application II - Nationwide analysis

In this section, we present the results for the nationwide analysis. Since the speed limit of the road sections is available, we use the speed ratio instead of the absolute speed. Thus, a high ratio implies congestion (large speed drop) compared to low ratio. An example 3D speed ratio map of the entire dutch highway for a certain time period is shown in figure 7.12.

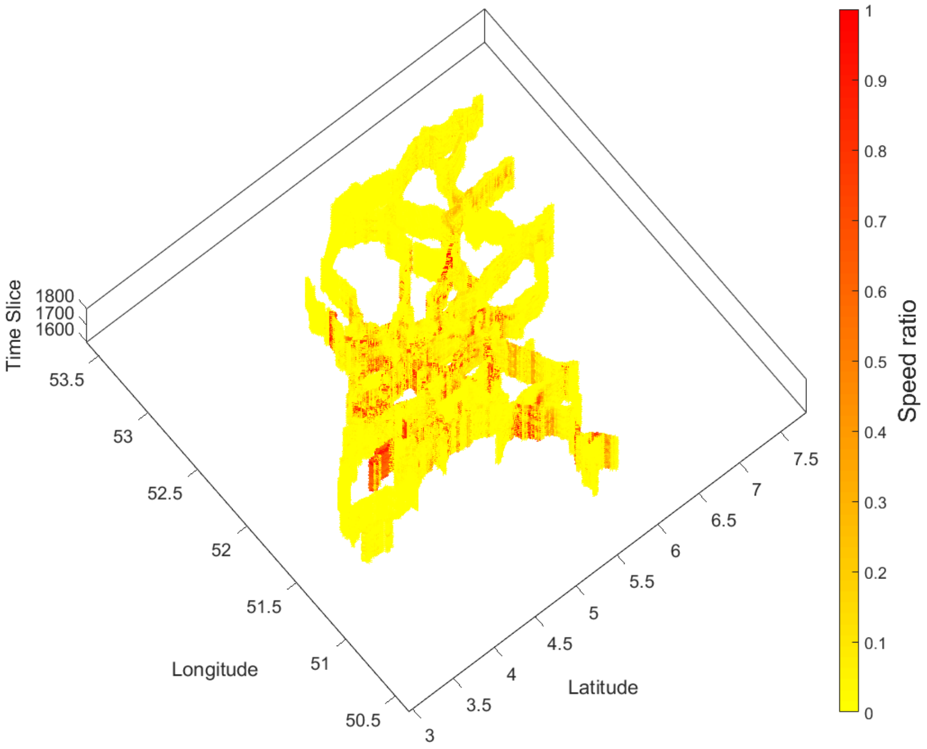
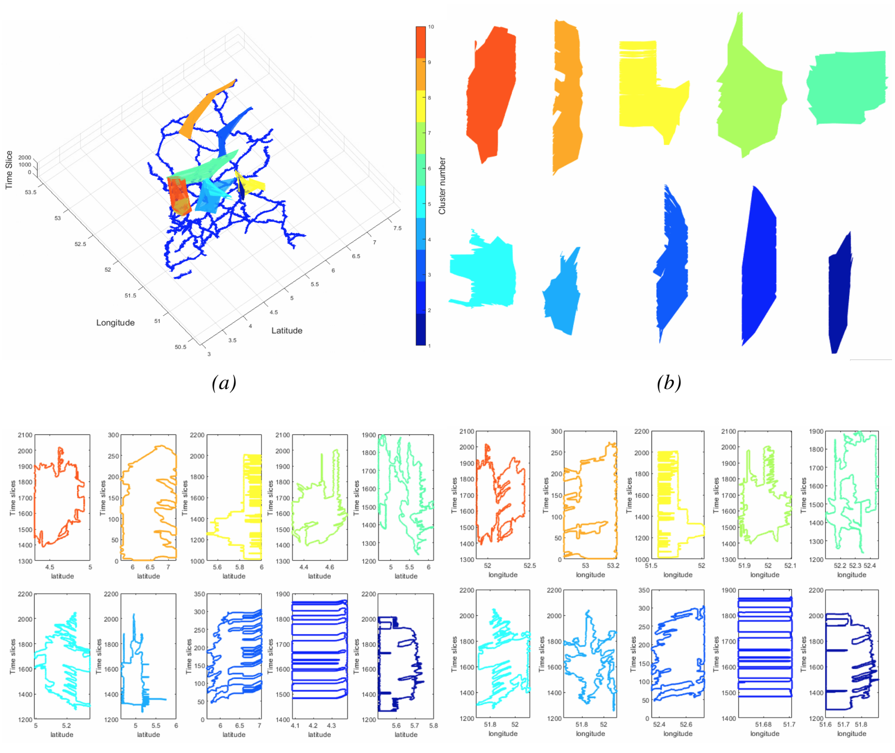


Figure 7.12: 3D speed map of the entire dutch highway during the evening peak period from 16:30 to 18:00 (red implies congestion, yellow implies free-flow), which is ≈ 7 million speed ratio values.

Since the speed limit is provided, we used $v^0 = 0.1$. Thus, all links with a speed ratio of less than 0.1 are considered as an uncongested region and hence are not clustered. We use k=5 to cluster the congested links into different levels of congestion. We identified the pocket of congestion from these clusters. An example of the top 10 pockets of congestion for a single day is shown in figure 7.13(a). The identified pockets of congestion is well-known bottlenecks within the Dutch highway network including the busy highway that connects Amsterdam to South Holland and Amsterdam to Utrecht. The corresponding 3D shapes of the pocket of congestion and the projected 2D shapes are shown in figure 7.13.



(a) *(b)*

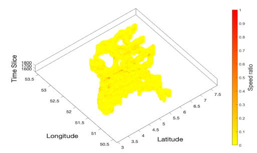
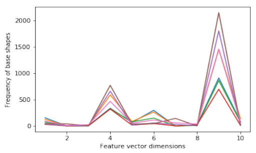
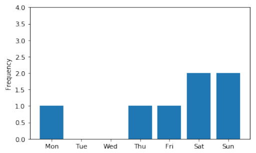
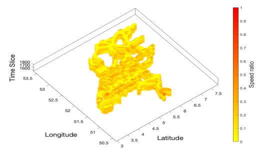
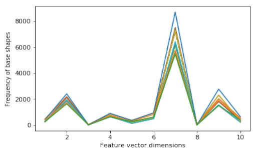
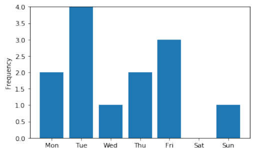
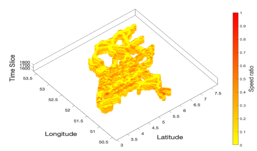
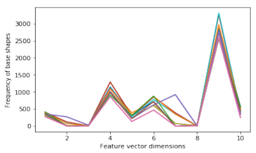
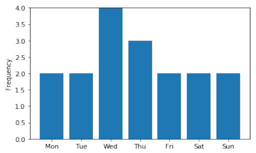
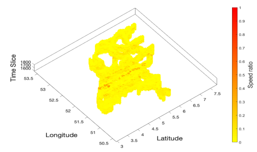
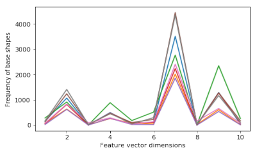
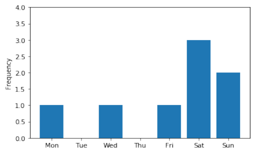
(c) *(d)*

Figure 7.13: Representing the top pockets of congestion for a given day as 2D shapes of the entire Dutch highway network (a) top 10 pocket of congestion for that day (b) corresponding 10 3D shapes (c) projected 2D shape representing the boundary between latitude and time (d) projected 2D shape representing the boundary between longitude and time.

The projected shapes constitute the bag of shapes. In this case, again only 2 base shapes are needed for representing 95% of the variance within the bag of shapes. Thus, the bag of shapes is clustered into 2, in which the first base shape contains $\approx$ 4000 shapes and the second base shape $\approx$ 18000 shapes. The shape model of the first base shape has 16 principal components whereas the second is composed of 18 principal components. This shape model can be used to build a compact custom feature vector with a dimension of 20 per day, given k = 5 and the number of base shape = 2.

The feature vector of the 45 days can be clustered to reveal regularity between the days as outlined in table 7.1. It includes the 4 shape-based speed models, the distribution of base shapes in the feature vector and the distribution of days in each class. The first 5 values of the feature vector dimension correspond to the occurrence of base shape 1 in each cluster *k* and the consecutive 5 values correspond to the frequency of the second base shape in each cluster. From the mean model of classes 1 and 4, we can see that they reflect relatively low congestion days which is in line with most of the weekdays categorized into these two classes. For all the classes, base shape 2 occurs more frequently than base shape 1. Only 1 to 2% of the network is congested (speed ratio > 0.1) in class 1 and 4 respectively whereas $\approx 41\%$ of the network is in congestion for classes 2 and 3.

Table 7.1: Description of Dutch highway network traffic states.

Class	**Mean model**	**Distribution of base shapes in each class**	**Distribution of days in each class**
1			
2			
3			
4			

We conduct a leave-one-out validation of the feature vectors to analyze if these daily variation are recurrent and hence, predictable. It achieves a prediction accuracy of 93% for the 45 days. A breakdown of the prediction is provided using a confusion matrix as shown in table 7.2. Like in the Amsterdam case, the congestion classes seem to perform better than the relatively low congestion classes. One day from class 1 is wrongly classified as 4 and two day from class 4 are wrongly classified as either 2 or 3. This is attributed to not embedding information about the space and time extent of the pocket of congestion into the feature vector. With the current feature vector, we do not differentiate between large and small congestion conditions if they have the same shape. It will be classified into the same class.

Notwithstanding, the feature vector is still able to successfully distinguish between different classes. However, the usefulness of the classes heavily depends on the application. If the aim is to identify the evolution of different traffic states at the local level, we need to incorporate more information in the feature vector. The current feature vector provides only a global picture of the traffic dynamics. The feature vector can be further extended to incorporate spatial correlation of the base shapes for different parts of the network to provide more details at the local level. Furthermore, by incorporating the extent of the variation of each base shape in relation to the mean shape, we can account the extent of the congestion in the feature vector. However, the compact nature of the feature vector specified in this study allows incorporating more information about the network dynamics at a low computational cost.

Table 7.2: Leave-one-out classification results

	PREDICTED			
KNOWN	Class 1	Class 2	Class 3	Class 4
Class 1	0.86	0	0	0.14
Class 2	0	1	0	0
Class 3	0	0	1	0
Class 4	0	0.08	0.07	0.75

Finally, we present the computational cost of building shape-based daily clusters for the entire Dutch network. The steps that differ between consensus and shape-based approach are the post-treatment and the model building steps. The post-treatment for the shape-based approach for a single day costs $\approx$ 20 minutes and processing the entire 45 days of data only takes around 15 hours. However, we estimate that it would take months to apply the post-treatment methodology on a single day of data when using the consensus method. As for building the models, building the shape models took $\approx$ 11 hours. The main computational component in this is building the distance matrix between the bag of shapes to estimate the number of base shapes. However, this can easily be parallelized as these are just pairwise distances. In contrast, given that each day contains $\approx$ 97 million values, running OEM (one element move) and optimizing over 45 days for the consensus method, the computational complexity is in the order of magnitude of months and not hours. Hence, we do not compute the consensus model for the nationwide analysis. In short, processing 45 days of data for the entire Dutch highway network using our approach costs less than 2 days on a 64-bit machine without any parallelization, which is very promising for large-scale network dynamics analysis. This allows conducting in-depth analysis and interpretation of these shapes and extending the feature vector to further understand the different aspects of a network exhibiting regular patterns.

7.5 Conclusion

In this chapter, we propose an efficient method for representing the traffic dynamics of a large-scale network using shapes. To this end, we first identify the pockets of congestion (i.e. connected sub-networks with low speed over space and time) and extract 3D shapes from these pockets. These 3D shapes are used to build a representative bag of shapes for the network. This bag essentially contains all the different shapes of congestion that occurs within this network. We extract the essence of these shapes as base shapes and build a Statistical Shape Model (SSM) for each of them. The SSM combined with the traffic variable is used to define custom feature vector for a daily pattern. These feature vectors are then used for accurate network travel time predictions by clustering feature vectors for a historical data set, and then matching the prevailing traffic conditions to one of the historical clusters. We also show how these feature vectors can be used to classify a new daily pattern into one of the historical clusters, which indicates the representativeness of the historical cluster and the feature vector.

We demonstrate our approach on an urban network (Amsterdam) and on a large-scale network (entire Dutch highway network) with promising results. We achieve a mean travel time prediction accuracy of 8%, which is $\approx 44\%$ improvement compared to the benchmark consensus method for the Amsterdam network. We are able to achieve this improvement with just two base shapes and a feature vector of dimension 40, which is a 99% data dimensionality reduction. For the nationwide analysis, we use the shape-based feature vectors to reveal the regularity between the days. The clustering provides meaningful network patterns with different types of congestion - low, mild, heavy congestion patterns. There are also significant differences between weekday and weekend patterns, with a majority of the weekends clustered under low to mild congestion classes, except for a few exemptions. These network pattern clusters are then used to build a classifier, which is able to achieve a classification accuracy of 93%. This implies that the feature vector is able to generalise and successfully uncover the difference between the daily patterns contained in the historical data set.

From the results of the two networks, we draw the following conclusions. We conclude that using shapes for understanding network traffic dynamics offers an efficient, insightful and accurate method. We have shown that defining pockets of congestion in a traffic network using shapes can reveal regularity between the daily patterns. The compact nature of the feature vector coupled with using tangible attributes allows to generate interpretable outcomes. We also found that for both case study networks the feature vector is able to encompass the congestion dynamics better than the free flow dynamics. In the first application, the travel times were underestimated and in the second application, the relatively low congestion days were the ones that were classified wrongly.

We postulate that there are many possible paths to further explore, refine, and improve the properties of the method. Here, we present key directions for further research. First, the feature vector can be extended to enrich it as well as make it more interpretable by including information about the size of the shape (which corresponds to the extent of congestion) or location as to where the shape occurred within a network. Moreover, the time dimension can be incorporated into the feature vector so that one can differentiate between the occurrence of certain congestion shapes in the morning and evening period and thus the feature vector can provide more details about the congestion propagation. Second, the evolution of shapes in different regions can be used to analyse local congestion propagation. This will allow associating certain shapes to certain bottlenecks and correlate the bottleneck with other bottlenecks of the network. Third, more sophisticated representative speed maps for each class can be built, other than just the mean model. Since, each day is represented by a set of shapes, we can consider the problem as a jigsaw puzzle of shapes and solve it accordingly. By decomposing the network dynamics problem into a set of shapes, the well-studied problem can be re-imagined as a vision problem, which is compelling for humans to grasp and it also opens up many exciting avenues of future research.

Chapter 8

Network passenger delay estimation

The previous chapters have focused on road traffic networks. We assert that the methods can be generalised for any problem with spatiotemporal data. However, for this the data needs to be converted to represent meaningful spatiotemporal maps. In this chapter, we propose a framework for estimating the network-wide passenger delay of a metro network. This is challenging and a new area of research as the data is unique and relatively hard to obtain. Furthermore, since the public transportation network is a two-tier system, it requires both the infrastucture and service networks to fully represent the network state.

This chapter is based on the following paper that is currently under review:

Panchamy Krishnakumari, Oded Cats and Hans van Lint. ”Estimation of Network Passenger Delay from Individual Trajectories.”, submitted 2019-09-30.

8.1 Introduction

Service reliability is known to be one of the most important determinants of transit performance, ridership and user satisfaction. Traditionally, service reliability has been focused on vehicles rather than customers using measures such as headway adherence or schedule punctuality [276]. However, there is a strong shift towards using passenger-focused measures for quantifying the transit performance [277, 278]. The data with which many of these quantities can be directly or indirectly estimated are already collected by many transport authorities through different sensors and information sources. Some of the well-known and increasingly used data sources for public transport network are infrastructure (stations and track segments) and service network (line) information, timetable data, automatic vehicle location (AVL) data which contains real-time locations of transit vehicles [279] and by implication the realization of the schedule and automatic fare collection (AFC) or smart card data with origin-destination specific information of passengers [46].

An increasing body of literature is available on the use of smart card data in public transit; reviews of which can be found in e.g.[46, 280]. Most of these studies focus on origin-destination matrix estimation [281, 282], extracting passenger patterns [283–285], network performance analysis [286, 287] and passenger route choice determinants [288]. AFC systems can be classified based on: (i) whether they include tap-in only or tap-in and tap-out records; (ii) whether fare validation is performed upon boarding (and possibly alighting) the transit vehicle or upon entering (and possibly leaving) a transit station. Depending on the two former aspects, destination (for boarding only) and vehicle (for station validation) inferences might be applied. Methods have been developed to infer the alighting station of a given tap-in record [289, 290]. In addition, new methods have been proposed for the train inference of each passenger, referred as passenger-train assignment [291, 292].

By contrasting the timetable and AVL data, the vehicle delay of the individual transit vehicles can be directly determined. However, in a shift towards passenger-focused measures, it is more interesting and relevant to investigate the *passenger delay* for a given line, network segment or transfer station as it encompasses the delay incurred by a passenger within the entire public transit system and not just the on-board delay. Clearly, with the increasing availability of AFC data and progress in passenger-train assignment research, it is possible to estimate and quantify the delay experienced by each passenger. However, it remains unknown how much of the passenger delay can be attributed to individual network elements. In particular, in the context of transit, delays can be associated with different travel time segments - initial waiting time, on-board time and transfer times.

To this end, we study passenger delays on individual network elements by fusing AVL and AFC data. The key contribution of this chapter thus is a new data-driven method to derive PT network delays from individual trajectories. The methodology has similarities with the data-driven OD estimation method we propose in [293], in that we construct a solvable system of equations utilizing all the information at hand without making more assumptions than strictly needed. The usage of passenger-train assignment data makes this study unique because this type of information is relatively new. To the authors' knowledge, this is the first study to explore the potential application of this unique data set. In this work, we distinguish two different types of passenger delay in relation to the public transit network: average passenger delay and total passenger delay. We define the *average passenger delay* as the delay incurred by a passenger while traversing a track segment (link), station (node) or trajectory. The *total passenger delay* is the total delay experienced by all the passengers that traverse that link, node or trajectory. Thus, the total passenger delay is a function of the number of passengers that traverse that network element during a given time period.

In the remainder of this chapter, we show that realized passenger-trajectories (resulting from AFC data and passenger-train assignment) and schedule information are sufficient for estimating average and total passenger delays for all the different network elements. The estimation improves by adding a constraint derived from AVL data, because these are—in our case—readily available. With these constraints for each passenger and each vehicle, a solvable system of equations can be formulated. There are various applications for the resulting network indicators, such as identifying key bottlenecks and critical network elements, prioritizing investments and maintenance of assets such as switches, modeling delay propagation through the network and an automated disruption detection. We demonstrate the estimation framework for the metro network of Washington DC using one year of data.

The chapter is organized as follows: section 8.2 describes the overall estimation framework; in section 8.3 we apply the framework on the Washington DC metro network. We outline the network and data used in this section and present the estimation results. We offer conclusions and a discussion on further research avenues in section 8.4.

8.2 Methodology

For convenient reference, the notation used for recurrent variables in the methodology is first presented as follows:

$\mathbf{G}(\mathbf{S},\mathbf{E},\mathbf{L},\mathbf{S^{trans}})$	Directed graph representing public transport network
$\mathbf{S}$	Set of stations $\{s_1, s_2, ...\}$
$\mathbf{E}$	Set of ordered pair of stations representing the track segments between the stations $\{(s_1,s_2),(s_2,s_3),...\}$
$\mathbf{L}$	Set of ordered pair of stations that defines a directed public transport service $\{l_1,l_2,...\}$
$\mathbf{S^{trans}}$	Set of transfer or interchange stations where transfer between lines occur $\{s_i,s_{i+1},...\}$, where $S^{trans} \subset S$
$\mathbf{s_o}$	Origin station of a passenger journey
$\mathbf{s_d}$	Destination station of a passenger journey
$\mathbf{r_{s_o,s_d}}$	Shortest trajectory between origin station s_o and destination station s_d, where $s_o, s_d \in S$
$\mathbf{r_{s_o,s_d,v}}$	Trajectory traversed by a transit vehicle v between origin station s_o and destination station s_d, where $s_o, s_d \in S$ and $v \in \{1...V\}$
$\mathbf{r_{s_o,s_d,n}}$	Inferred trajectory of a passenger n between origin station s_o and destination station s_d, where $s_o, s_d \in S$ and $n \in \{1...N\}$

$\widetilde{\mathbf{t}}_{\mathbf{s_o,s_d,k}}$ Scheduled travel time between origin station s_o and destination station s_d departing at period k

$\widetilde{\mathbf{t}}^{\mathbf{veh}}_{\mathbf{s_i,s_{i+1},k}}$ Scheduled transit vehicle run-time and dwell time between track segment (s_i, s_{i+1}) departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$\mathbf{h_{l,k}}$ Headway between transit vehicles for a given line $l \in L$ departing within period k

$\mathbf{t}^{\mathbf{walk}}_{\mathbf{s_i}}$ Walking time between gate and platform or vice-versa for each station $s_i \in S$ and if $s_i \in S^{trans}$, the walking time refers to walking time between line directions

$\mathbf{t_{s_o,s_d,k,n}}$ Observed travel time between origin station s_o and destination station s_d for passenger n departing at period k

$\mathbf{t}^{\mathbf{veh}}_{\mathbf{s_i,s_{i+1},k,v}}$ Observed transit vehicle run-time and dwell time between track segment (s_i, s_{i+1}) for a vehicle v departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$\mathbf{d_{s_o,s_d,k,n}}$ Estimated average passenger delay between origin station s_o and destination station s_d for passenger n departing at period k

$\mathbf{d}^{\mathbf{veh}}_{\mathbf{s_o,s_d,k,v}}$ Estimated transit vehicle delay between origin station s_o and destination station s_d departing at period k for a given vehicle v servicing a line

$\mathbf{d}^{\mathbf{on-board}}_{\mathbf{s_i,s_{i+1},k}}$ Estimated transit vehicle-run or on-board delay between track segment (s_i, s_{i+1}) departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$\mathbf{d}^{\mathbf{wait}}_{\mathbf{s_o,k}}$ Estimated initial waiting time delay at origin station s_o departing at period k

$\mathbf{d}^{\mathbf{trans}}_{\mathbf{s_i,k}}$ Estimated transfer delay at transfer station s_i departing at period k

8.2.1 Problem formulation

A passenger trajectory or journey $r_{s_o,s_d,n}$ between origin station s_o and destination station s_d of passenger $n \in \{1, ...N\}$, where N is the total number of passengers, is defined based on two sets, one of stops and one of lines, each of which are a subset of S and L, respectively. The combination of which allows one to define a third set: the set of track segments (subset of E) that is traversed by the passenger. As a result, we obtain the initial stop (first element in the stop set), the intermediate transfer stops (second to one before last element in the stop set) and the link set. The stop set is defined as $\{s_1, .., s_i, s_{i+1}, .., s_m\}$, where $s_2, .., s_i, ..s_{m-1}$ are the m transfer stations for that journey, in which s_1 is the origin station s_o and s_m is the destination station s_d. The line set for the same journey is defined as $\{l_1,..,l_i,..l_{m-1}\}$.

In the following, we assume that the AFC system is a tap-in-tap-out system with the passenger-train assignment provided. We also assume that the origin and destination station, and the inferred trajectory $r_{s_o,s_d,n}$ of each passenger are known. Thus, the passenger delay $d_{s_o,s_d,k,n}$ is defined as:

$$d_{s_o,s_d,k,n} = \begin{cases} t_{s_o,s_d,k,n} - \widetilde{t}_{s_o,s_d,k} & if\ t_{s_o,s_d,k,n} > \widetilde{t}_{s_o,s_d,k} \\ 0 & otherwise \end{cases} \tag{8.1}$$

where $t_{s_o,s_d,k,n}$ is the observed travel time for passenger n departing at time period k and can be obtained by finding the difference between the tap-in and tap-out time of that passenger. Any negative delay $d_{s_o,s_d,k,n}$ is set to 0 as shown in (8.1), since this implies the passenger reached the destination earlier than expected.

The maximum scheduled travel time $\widetilde{t}_{s_o,s_d,k}$ for a journey r_{s_o,s_d} can be defined in several ways. In this work, the following definition has been adopted:

$$\widetilde{t}_{s_o,s_d,k} = \sum_{l \in r_{s_o,s_d}} \sum_{s_i \in l} \widetilde{t}^{veh}_{s_i,s_{i+1},k} + \sum_{l \in r_{s_o,s_d}} h_{l,k} + \sum_{s_i \in r_{s_o,s_d}} \sum_{s_i = s_1}^{s_{m-1}} t^{walk}_{s_i} \tag{8.2}$$

where $\widetilde{t}_{s_o,s_d,k}$ is composed of scheduled running times and dwell times $\widetilde{t}^{veh}_{s_i,s_{i+1},k}$ assigned to track segments connecting subsequent stations s_i and s_{i+1}, headway $h_{l,k}$ between successive services for the lines l in that journey and walking time $t^{walk}_{s_i}$ at the origin station and transfer stations denoted by s_i. The headway between transit services is included in calculating the schedule travel time due to the definition of on-time journey considered in this study. Note that a passenger is considered to be on-time with regards to the maximum scheduled journey time even he/she just missed a transit vehicle but was able to catch the next service of that line and the headway between the services account for that fallback time.

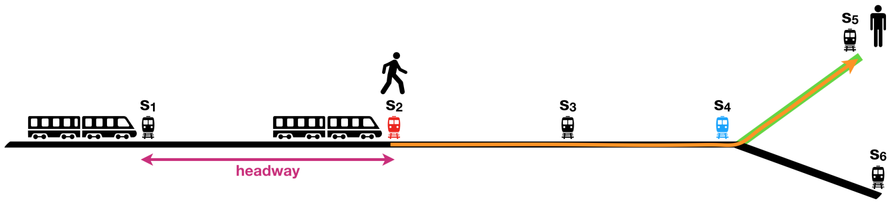
Given the individual trajectories of the passengers, the aim is to decompose the delay experienced on a given passenger journey into the corresponding network elements they traversed along their trajectory. We assume that the observed travel time of a passenger $t_{s_o,s_d,k,n}$ comprises of travel time that can be attributed to one of the following network elements of the transit service:

1. Time spent at the origin stop of the journey
2. On-board a transit vehicle along one of the segments
3. Time spent at a transfer station

Similarly, the delay experienced by a passenger is comprised of the delays occurring at these network elements. The composition of the scheduled and observed passenger travel time is illustrated in figure 8.1. Based on this schematic representation, the passenger delay between stations (s_o, s_d) for a given departure time k can be defined as:

$$d_{s_o,s_d,k,n} \geq d^{wait}_{s_o,k} + \sum_{l \in r_{s_o,s_d,n}} \sum_{s_i \in l} d^{on-board}_{s_i,s_{i+1},k} + \sum_{s_i \in r_{s_o,s_d,n}} \sum_{s_i = s_1}^{s_{m-1}} d^{trans}_{s_i,k} \tag{8.3}$$

where $d^{wait}_{s_o,k}$ is the initial waiting delay at the origin station s_o, $d^{on-board}_{s_i,s_{i+1},k}$ is the on-board delay between track segment (s_i, s_{i+1}) and $d^{trans}_{s_i,k}$ is the transfer delay at transfer stop s_i in



scheduled travel time = on-board time + headway + walking time

observed passenger travel time = waiting time + on-board time + transfer time

passenger delay = observed passenger travel time - scheduled travel time

Figure 8.1: Schematic picture of scheduled and observed travel time between station s_2 and s_5.

journey $r_{s_o,s_d,n}$. The inequality in (8.3) is due to potential *non-observable* personal travel components which may add additional delays such as performing an activity at one of the stations within the gated area. Formulating the relationship between personal delays and the three network related delay components by means of an inequality constraint allows us to perform the delay inference without ignoring unobserved personal delay components. We define this unobserved error term as a slack variable for each passenger; thus (8.3) can be reformulated as:

$$d_{s_o,s_d,k,n} = d^{wait}_{s_o,k} + \sum_{l\in r_{s_o,s_d,n}} \sum_{s_i\in l} d^{on-board}_{s_i,s_{i+1},k} + \sum_{s_i\in r_{s_o,s_d,n}} \sum_{s_i=s_1}^{s_{m-1}} d^{trans}_{s_i,k} + \varepsilon_n \tag{8.4}$$

An additional constraint can also be formulated, that pertains to the on-board delay component, which must be equal to the transit vehicle delay between the corresponding trip segments. This vehicle delay can be directly inferred from the AVL data and the schedule information as follows:

$$d^{veh}_{s_o,s_d,k,v} = \sum_{l\in r_{s_o,s_d,v}} \sum_{s_i\in l} \begin{cases} t^{veh}_{s_i,s_{i+1},k,v} - \tilde{t}^{veh}_{s_i,s_{i+1},k} & if\ t^{veh}_{s_i,s_{i+1},k,v} > \tilde{t}^{veh}_{s_i,s_{i+1},k} \\ 0 & otherwise \end{cases} \tag{8.5}$$

This results in an additional set of equations to solve for the unknowns in (8.4) which reads

$$d^{veh}_{s_o,s_d,k,v} = \sum_{l\in r_{s_o,s_d,v}} \sum_{s_i\in l} d^{on-board}_{s_i,s_{i+1},k} \tag{8.6}$$

where $d^{veh}_{s_o,s_d,k,v}$ is the delay for transit vehicle v servicing the line that starts at s_o and ends at s_d within time k and $r_{s_o,s_d,v}$ is the trip composed of track segments visited by transit vehicle v. The initial waiting time and transfer delay component are irrelevant for a transit vehicle trajectory. The slack variable can also be set to 0 as there is no unobserved delay component for the vehicle delay.

8.2.2 Formulating a solvable system of equations

We decompose each of the passenger delay into delay at three network elements - origin stations, track segments and transfer stations and each transit vehicle delay into delay per track segments. With eq.8.4, now each passenger trajectory can be written as a linear combination of these three passenger delay component types with an additional slack variable for the error. Our main hypothesis is that formulating (8.4) for all passenger trajectories leads to a potentially solvable system of equations, since each passengers trip between k and $k+t_{s_o,s_d,k,n}$ serves as a constraint for all other trips that traverse one or more common network elements during this trip.

We demonstrate this point with an example. Consider the toy network shown in figure 8.1. If a passenger traveling from s_1 to s_5 experiences no delay while a passenger traveling from s_1 to s_6 experiences a delay of say 5 minutes, then this delay is probably due to delay occurring between stations s_4 and s_6. Thus, given a sufficient number of passengers, there exists a bounded solution for (8.4).

Given that $d_{s_o,s_d,k,n}$ and $r_{s_o,s_d,n}$ are known, we can expand each delay in (8.4) to a linear combination of single network elements. Thus, the initial waiting time can be expanded as a linear combination of station elements as follows:

$$d^{wait}_{s_o,k} = \sum_{s_i \in S} \alpha_{s_i,k,n} d^{wait}_{s_i,k} \tag{8.7}$$

where

$$\alpha_{s_i,k,n} = \begin{cases} 1 & s_i = s_o \; \forall \; s_o \in r_{s_o,s_d,n} \\ 0 & \text{otherwise} \end{cases} \tag{8.8}$$

The on-board time can be reformulated based on track segment elements:

$$\sum_{l \in r_{s_o,s_d,n}} \sum_{s_i \in l} d^{on-board}_{s_i,s_{i+1},k} = \sum_{(s_i,s_{i+1}) \in E} \beta_{s_i,s_{i+1},k,n} d^{on-board}_{s_i,s_{i+1},k} \tag{8.9}$$

where

$$\beta_{s_i,s_{i+1},k,n} = \begin{cases} 1 & (s_i,s_{i+1}) \in r_{s_o,s_d,n} \\ 0 & \text{otherwise} \end{cases} \tag{8.10}$$

The transfer time can be defined based on the transfer stations:

$$\sum_{s_i \in r_{s_o,s_d,n}} \sum_{s_i = s_i}^{s_{m-1}} d^{trans}_{s_i,k} = \sum_{s_i \in S^{trans}} \gamma_{s_i,k,n} d^{trans}_{s_i,k} \tag{8.11}$$

where

$$\gamma_{s_i,k,n} = \begin{cases} 1 & s_i \in r_{s_o,s_d,n} \\ 0 & \text{otherwise} \end{cases} \tag{8.12}$$

Note that the stations and transfer stations are directed for public transport network. The number of station directions depends on the number of its outgoing neighbors. Thus, for station node s_4 in figure 8.1, there are three directions possible, either towards s_3, s_5 or s_6. This implies that the number of directed stations in a public transport network is the same as the number of directed edges. We thus refine (8.7) and (8.11) based on these directed travel nodes as follows:

$$d^{wait}_{s_o,k} = \sum_{(s_i,s_{i+1})\in E} \alpha_{s_i,s_{i+1},k,n} d^{wait}_{s_i,s_{i+1},k} \tag{8.13}$$

where

$$\alpha_{s_i,s_{i+1},k,n} = \begin{cases} 1 & s_i = s_1 = s_o \ \forall \ (s_1,s_2) \in r_{s_o,s_d,n} \\ 0 & \text{otherwise} \end{cases} \tag{8.14}$$

$$\sum_{s_i \in r_{s_o,s_d,n}} \sum_{s_i=s_i}^{s_{m-1}} d^{trans}_{s_i,k} = \sum_{(s_i,s_{i+1})\in E} \gamma_{s_i,s_{i+1},k,n} d^{trans}_{s_i,s_{i+1},k} \tag{8.15}$$

where

$$\gamma_{s_i,s_{i+1},k,n} = \begin{cases} 1 & (s_i,s_{i+1}) \in r_{s_o,s_d,n} \ \forall \ s_i \in S^{trans} \\ 0 & \text{otherwise} \end{cases} \tag{8.16}$$

Next, we can reformulate the passenger delay given in (8.4) as:

$$\begin{aligned} d_{s_o,s_d,k,n} = \sum_{(s_i,s_{i+1})\in E} \alpha_{s_i,s_{i+1},k,n} d^{wait}_{s_i,s_{i+1},k} + \sum_{(s_i,s_{i+1})\in E} \beta_{s_i,s_{i+1},k,n} d^{on-board}_{s_i,s_{i+1},k} \\ + \sum_{(s_i,s_{i+1})\in E} \gamma_{s_i,s_{i+1},k,n} \delta^{trans}_{s_i,s_{i+1},k} + \varepsilon_n \end{aligned} \tag{8.17}$$

Similarly, we can formulate the transit vehicle delay given in (8.6) as:

$$d^{veh}_{s_o,s_d,k,v} = \sum_{(s_i,s_{i+1})\in E} \beta_{s_i,s_{i+1},k,v} d^{on-board}_{s_i,s_{i+1},k} \tag{8.18}$$

where

$$\beta_{s_i,s_{i+1},k,v} = \begin{cases} 1 & (s_i,s_{i+1}) \in r_{s_o,s_d,v} \\ 0 & \text{otherwise} \end{cases} \tag{8.19}$$

Based on the example network from figure 8.1, for a given passenger 1 with his or her journey defined by link set $\{(s_2,s_3),(s_3,s_4),(s_4,s_5)\}$ and stop set $\{s_2,s_4,s_5\}$ where s_4 is a transfer station, we can write the expanded form of (8.17) for passenger 1 as:

$$d_{s_o,s_d,k,1} = d^{wait}_{s_2,s_3,k} + d^{on-board}_{s_2,s_3,k} + d^{on-board}_{s_3,s_4,k} + d^{on-board}_{s_4,s_5,k} + d^{trans}_{s_4,s_5,k} + \varepsilon_1 \tag{8.20}$$

Similarly, for a given transit vehicle 1 with its journey defined using link set $\{(s_1,s_2),(s_2,s_3),(s_3,s_4),(s_4,s_5)\}$, we can write the expanded form of (8.18) for vehicle 1 as:

$$d^{veh}_{s_o,s_d,k,1} = d^{on-board}_{s_1,s_2,k} + d^{on-board}_{s_2,s_3,k} + d^{on-board}_{s_3,s_4,k} + d^{on-board}_{s_4,s_5,k} \tag{8.21}$$

We can generalize (8.17) and (8.18) to N passengers and V vehicles respectively to build a system of equations and thus a matrix equation as:

$$C\mathbf{x} + \varepsilon = B, \ \ \mathbf{x} \geq 0; \tag{8.22}$$

where

$$C = \begin{bmatrix} 0 & 1 & \dots & 0 & 1 & \dots & 0 & 0 & \dots \\ \vdots & \ddots & & & & & & & \\ \alpha_{s_1,s_2,k,n} & \alpha_{s_1,s_2,k,n} & \dots & \beta_{s_1,s_2,k,n} & \beta_{s_2,s_3,k,n} & \dots & \gamma_{s_1,s_2,k,n} & \gamma_{s_2,s_3,k,n} & \dots \\ \vdots & \ddots & & & & & & & \\ 0 & 0 & \dots & 1 & 1 & \dots & 0 & 0 & \dots \\ \vdots & \ddots & & & & & & & \\ 0 & 0 & \dots & \beta_{s_1,s_2,k,v} & \beta_{s_2,s_3,k,v} & \dots & 0 & 0 & \dots \\ \vdots & \ddots & & & & & & & \end{bmatrix};$$

$$\mathbf{x} = \begin{bmatrix} d^{wait}_{s_1,s_2,k} \\ d^{wait}_{s_2,s_3,k} \\ \vdots \\ d^{on-board}_{s_1,s_2,k} \\ d^{on-board}_{s_2,s_3,k} \\ \vdots \\ d^{trans}_{s_1,s_2,k} \\ d^{trans}_{s_2,s_3,k} \\ \vdots \end{bmatrix}; \ \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \end{bmatrix} \text{ and } B = \begin{bmatrix} d_{s_o,s_d,k,1} \\ \vdots \\ d_{s_o,s_d,k,n} \\ \vdots \\ d^{veh}_{s_o,s_d,k,1} \\ \vdots \\ d^{veh}_{s_o,s_d,k,v} \\ \vdots \end{bmatrix} \tag{8.23}$$

The **x** is shorthand for the delay attributed to each network element; C contains ones or zeros (LHS of equations (8.17) and (8.18)); ε is the slack variable for each passenger and B corresponds to the passenger and vehicle delay (RHS of equations (8.17) and (8.18)), respectively. A zero in the C matrix implies that the corresponding network element does not contribute to the respective passenger or vehicle delay in B and a value of one imply that the corresponding network element is part of that passenger's or vehicle's journey and hence contributes to the delay. Thus, equations (8.17) and (8.18) now constitute a solvable system of equations in which $d^{wait}_{s_i,s_{i+1},k}$, $d^{on-board}_{s_i,s_{i+1},k}$ and $d^{trans}_{s_i,s_{i+1},k}$ are the unknowns $\forall\ (s_i,s_{i+1}) \in E$ and $d_{s_o,s_d,k,n}$ and $d^{veh}_{s_o,s_d,k,v}$ are known $\forall\ n \in \{1,...N\}$ and $v \in \{1,...V\}$ respectively. The matrix equality given in equation (8.22) can be solved using a constrained linear least squares solution [249] with the lower bound set to 0 to ensure a non-negative solution as follows:

$$\min_{\mathbf{x}} \frac{1}{2} ||C\mathbf{x} - B||_2^2 \tag{8.24}$$

In case a non-negative solution does not exist, an ordinary least square solution is computed and the negative values of the delay matrix are ignored when computing the estimation error.

8.2.3 Evaluation metrics

An optimal solution is reached when ε is minimized. Thus, we use the slack variable given in equation (8.25) to evaluate the estimation results. The delay estimates **x** of the individual network elements can be used to recalculate the individual passenger delay by multiplying C and $\mathbf{x}$. This can be used to find the estimation error or the slack variable as follows:

$$\varepsilon = B - C\mathbf{x} \tag{8.25}$$

Based on our problem formulation, a positive slack variable is favored as this implies that our estimates are less than or equal to the observed delay of the passenger. This is reasonable as some of the delay may be attributed to their individual choice of departure time or other personal activities rather than the associated network element delay. Conversely, a negative value for slack implies that our estimation attributes to individual network elements a total delay that exceeds the delay experienced by a passenger.

8.3 Application

In this section, we demonstrate our estimation method on a real-world application. For this, we first explain the data and network used (section 8.3.1). Thereafter, we provide some useful descriptive statistics of the data (section 8.3.2) and finally, we present the results of the estimation (section 8.3.3) and analyze its performance (section 8.3.4).

8.3.1 Data

We demonstrate the estimation approach on a data set of smart card data from the Washington DC Area Metro Transit Authority (WMATA) in the United States. The data is composed of one year of smart card data from 19 August 2017 to 28 August 2018 for the entire metro network of Washington DC which contains the passenger-train assignment outputs derived from an application of the so-called ODX method described in [290]. In addition, the dataset also includes the rail movement data, schedule information and disruption log file. The metro network is comprised of 6 lines, 91 stations, 186 links and 9 transfer stations as shown in Figure 8.2.

8.3.2 Descriptive statistics

The network has an average ridership of $\approx$ 438 000 rides per day and a total of $\approx$ 157 million rides during the entire study period with an average journey time of 28 minutes per passenger. Of these, 14% of the passengers experience delay, with a mean delay of 6 minutes given a delay. Moreover, 39% of passenger trips include transfers with an average of 1.14 trip legs per passenger journey. A breakdown of the number of passengers for different time periods is shown in figure 8.3. There are two distinct peaks in the passenger distribution - morning and afternoon peak.

8.3.3 Estimation results

In this section, we present the results of the delay estimation for the Washington metro network. We chose a temporal aggregation of 30 minutes for k in (8.17), since the maximum headway between transit vehicles is 20 minutes and choosing a temporal aggregation smaller than that would imply that there would be no vehicles between some OD pairs,

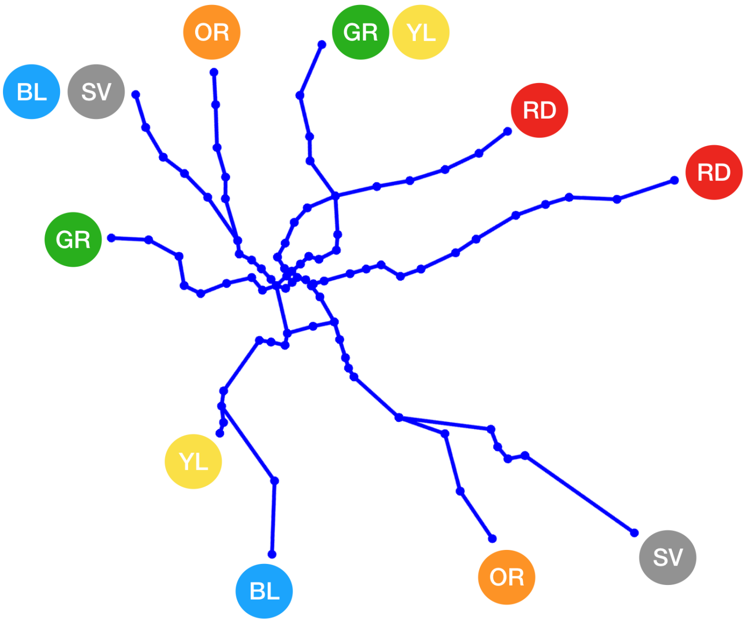


Figure 8.2: Washington metro network

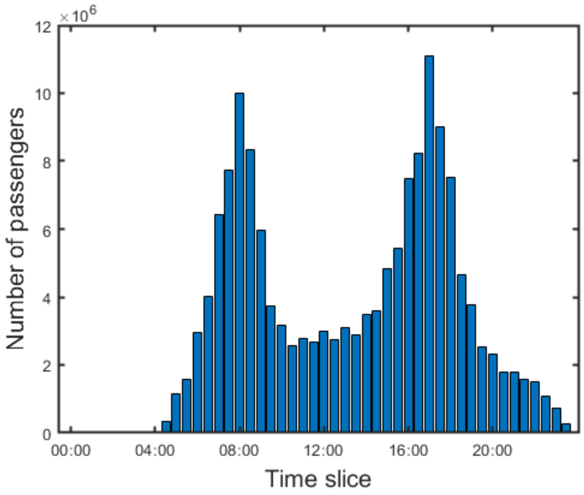
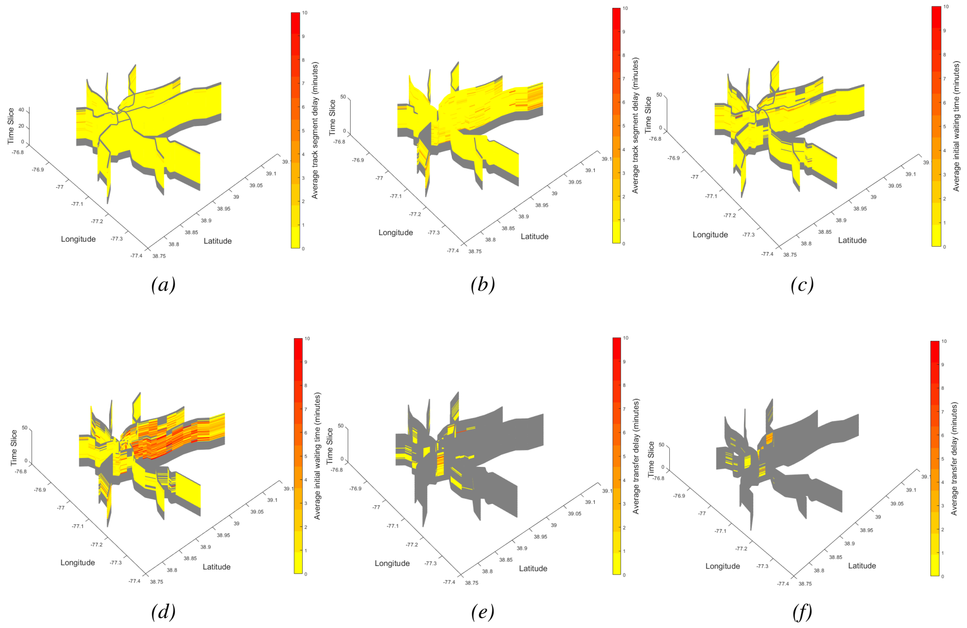


Figure 8.3: Passenger distribution of all days for different time periods

hence no passengers and consequently no system of equations. Having an aggregation of 30 minutes ensures that at least one transit vehicle per time period is included in the system of equations as represented in (8.17).

Figure 8.4 shows the estimation results of the average passenger delay for the three network elements for a selected weekday (Thursday), i.e. March 1, 2018, with a temporal aggregation of 30 minutes. There is no significant track segment delay for this particular day. The waiting time and transfer delay are mapped on a link rather than a node as we incorporate directionality of the node to distinguish journeys between different lines. This allows us to build these three compact 3D graphs for visualizing the delay propagation of each day for each of the delay components. This can be used for evaluating the performance of the metro network on a given day or over a long period of time or estimate the passenger delay incurred between any given origin-destination (OD) pair. Moreover, there were ≈ 596 000 rides for this particular weekday and we were able to represent the dynamics of the network using these three 3D networks with the dimensions 3 × 48 × 186 (3 × temporal aggregation × number of links), thus leading to a dimensionality reduction of about 95%.

A 3D graph of a weekend day (Saturday), i.e. March 3, 2018, in the same month is also shown in figure 8.4. This exhibits a significant delay in the red line compared to a normal weekday. From the disruption log file of WMATA, we are able to attribute this delay to late track clearing in the morning which had a cascading effect on the rest of the line. One of the main findings from our delay estimation is that not all the delays have a corresponding explanation in the disruption log file of WMATA, which is maintained manually, and vice-versa. Thus, our estimation can be used to enrich the log file with additional incidents that caused passenger delay as well as quantify the consequences of these incidents in terms of the number of passengers affected, the average passenger delay and the spatial extent of the



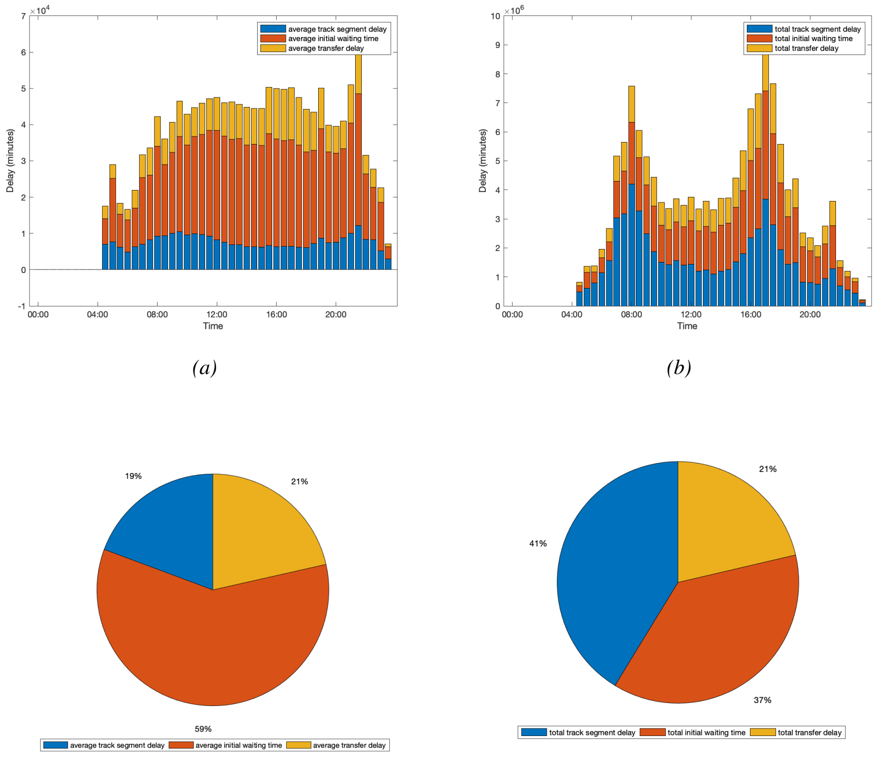
(a) (b) (c)

(d) (e) (f)

Figure 8.4: Estimation results of average track segment delay, average initial waiting time and average transfer delay for a weekday dated 01-03-2018 is shown in (a), (c) and (e) respectively and for a weekend day dated 03-03-2018 is shown in (b), (d) and (f) respectively

impact of the incident.

We applied the estimation framework on the 359 days of smart card data with around 157 million rides and compressed each day into the three 3D delay networks, leading to an overall dimensionality reduction of 94%. We also used the estimates to explore the distribution of the average and total passenger delays for the entire analysis period decomposed into different network elements and different time periods as shown in figure 8.5. The average passenger delay, shown in figure 8.5(a), is stable compared to the morning and evening peak in the total passenger delay distribution in figure 8.5(b). Thus, delays occur at all time periods, whereas more passengers are affected during the peak periods as can be expected. Figures 8.5(c) and (d) show the contribution of each network element delay to the overall delay. In the case of average passenger delay, 59% of the delay is associated with the initial waiting time. However, when the number of passengers affected is considered, track segment delay contributes the most with 41% of the delay.



(a)

(b)

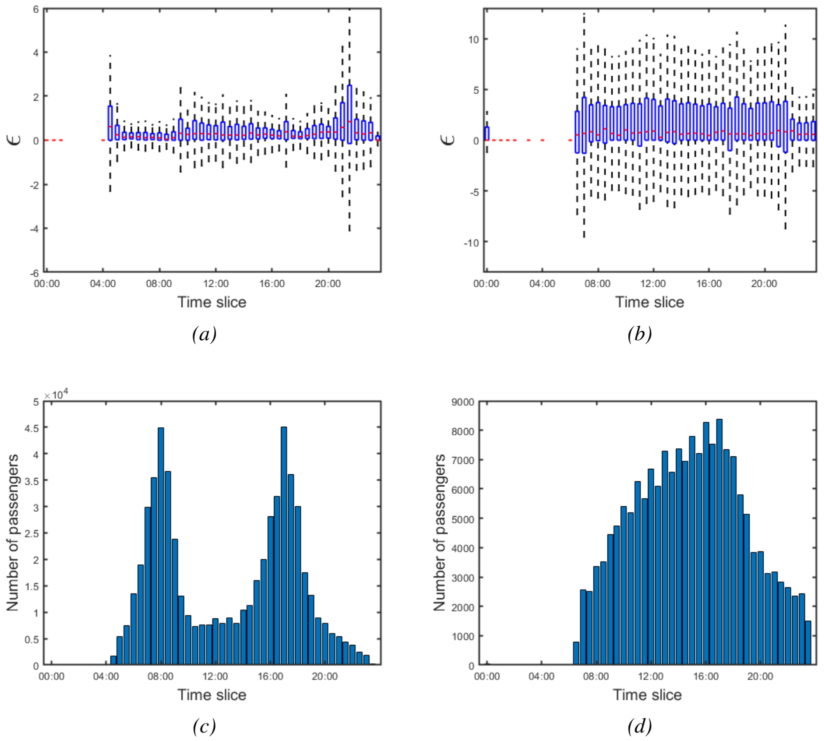
(c)

(d)

Figure 8.5: Delay distribution of the whole dataset across the three network elements for different time periods based on (a) average passenger delay and (b) total passenger delay. Overall delay distribution of the whole dataset across the three network elements based on (c) average passenger delay and (d) total passenger delay

8.3.4 Validation

We evaluate the validity of the estimation by reconstructing the individual passenger delay and calculating the slack variable. A slack value of 0 implies that the delay experienced by a passenger based on the AFC data is the same as obtained by summing over our estimates for the respective journey travel components. We do except a slight variation in the slack value of each passenger due to heterogeneity in user behavior. This is confirmed by inspecting the distribution of the slack variable using boxplot for the above mentioned selected weekday and weekend day shown in figures 8.6(a) and (b).

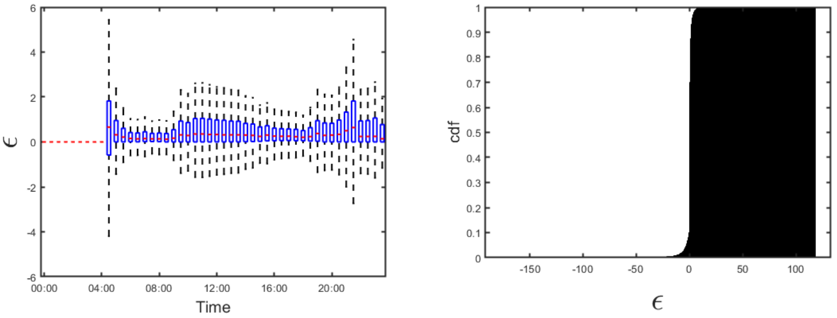


(a) *(b)* *(c)* *(d)*

Figure 8.6: (a) and (b) Slack value distribution per time period which also shows the dispersion within the time period for a weekday and a weekend respectively; (c) and (d) Passenger distribution per time period for a weekday (01-03-2018) and a weekend day (03-03-2018), respectively

In figures 8.6(a) and (b), a compact boxplot with small variation is desired as this implies that the delay estimated for all passengers at the time period falls within that small range of values. Furthermore, the dispersion of the slack variable increases with a decrease in the number of passengers for a given time period which can be seen in figure 8.6 as slack distributions in figures 8.6(a) and (b) are complemented with passenger ridership distributions in figures 8.6(c) and (d). This is expected as more passengers mean more equations and thus more confidence in the estimates and thus a lesser dispersion. The effect of the number of passengers is also evident from the distribution of the slack for the weekday. The weekday has a smaller number of passengers relative to the weekday and thus the boxplot is also relatively compact. Moreover, the distinctively different passenger distribution than the weekday connect to the slack variable result.

Another important finding from figures 8.6(a) and (b) is that some passengers have a negative slack. This is presumably primarily due to the time aggregation. Our estimation is aggregated for a certain k time interval. Assume that a disruption occurred at $k+i$ time whereas a passenger departed at time $< k+i$, thus avoiding the disruption. However, due to the level of aggregation, that trip will be categorized as being affected by the disruption, since most of the passengers in that time period experienced that delay.



(a)

(b)

Figure 8.7: (a) Overall slack distribution of all passengers across all days for different time period (b) Cumulative distribution function of the slack values for all passengers.

The slack distribution for all the days across different time periods is shown in figure 8.7. As can be seen from the figure, most of the values are distributed around zero and our estimation framework achieves a mean slack of 0.072 minutes across all the passengers which is promising. Additionally, only 10.76% of all passengers had a negative slack value, with a mean of -5 minutes. Moreover, when complemented with the passenger distribution shown in figure 8.3, it is evident that the boxplot at a time period with a large number of passengers are more compact with less dispersion compared to the time period with less number of passengers. This is, also, in line with the finding from the individual weekday and weekend.

8.4 Conclusion

In this study, we propose a new estimation method to map passenger delay into network elements. The outputs of our method can aid in measuring network performance for any given origin-destination pair of the public transportation network and in prioritizing measures for improving service robustness. We decompose the delay along a passenger trajectory into its corresponding track segment delay, initial waiting time and transfer delay. We demonstrate the method using one-year data from the Washington metro network.

Our method estimates how passenger delays are distributed across network elements and can thus assess the contribution of each network element to system performance. We were able to achieve a dimensionality reduction of 94% by representing these individual trajectories as 3D networks. The estimation results show that the confidence of the estimation, measured based on the compactness of the boxplot, increases with an increase in the number of passengers. Overall, our estimation framework achieved a mean slack of 0.072 minutes or less than 5 seconds per passenger, which is very promising.

These network element estimates are used to analyze the average and total passenger delay of the metro network. The average passenger delay is more or less stable throughout the day whereas total passenger delay, which accounts for the number of passengers affected, contains two distinct peaks. The two peaks correspond to the morning peak and evening peak with the evening peak associated with greater passenger delay than the morning peak. The initial waiting time contributes the most to the average passenger delay with 59% whereas the track segment delay contributes the most to the total passenger delay with 41% of the total delay being attributed to it.

The estimates generated by the method proposed in this chapter open new avenues for future research. The estimation framework can be adapted to allow non-uniform temporal aggregation to reduce the share of passenger delay records for which the slack variable value is negative. The delay estimates at the individual network element level can be used to reveal hourly, daily, weekly or seasonal delay patterns across the metro network. Moreover, the estimation approach is easily transferable to other locations. For this, it is necessary to ensure that the system of equations is solvable and that a different temporal aggregation might be needed depending on the headway between the transit vehicles. Furthermore, it can help in understanding the delay propagation through a network and potentially contribute to the prediction of such delays and their spill-over impacts. Such advancements can ultimately help in supporting decision making in relation to improved service robustness at both the tactical level (e.g. locating switches to allow for short-turning) and operational level (e.g. disruption mitigation strategies including information provision and resource allocation).

Chapter 9

Conclusion

This chapter discusses the significance of our studies and some of the opportunities to extend this research. We start with a summary of our key findings. Next, we discuss the implications of this work for practitioners and society. Finally, we propose recommendations for the future research in this area.

We identified the current challenges of doing network-wide analysis of large-scale traffic patterns, arising from the limitations of model-based approaches and the failure of data-driven approaches to incorporate space and time jointly. Recent developments in the macroscopic fundamental diagram provided new insights into traffic patterns by looking at the patterns at a higher level. We have improved upon this finding to build efficient data-driven methods for analyzing and interpret the traffic dynamics of large-scale metropolitan networks. The methods presented in this thesis improve the efficiency of understanding network dynamics in several ways:

- by reducing the dimensionality of the network through coarsening [Chapter 2]
- by reducing the dimensionality of the data through clustering [Chapter 3]
- by reducing the dimensionality of the data using high-level features such as shapes to define the traffic patterns [Chapter 4]
- by reducing the dimensionality of the data using image features [Chapter 5]
- by exploring the application of such network dynamics representation in OD matrix estimation [Chapter 6]
- by extending the methods to be applicable for nationwide analysis [Chapter 7]
- by paving the way for extending the methods to other modes by proposing an estimation approach to represent passenger delay dynamics in a public transit network as a compact 3D map. [Chapter 8]

We can, therefore, conclude that these contributions provide substantial improvement in efficiently analyzing network-wide traffic for practitioners and researchers.

9.1 Key findings

We now briefly summarize each chapter's findings to answer the proposed research questions.

In **chapter 2**, we developed a heuristic method for automatically generating multi-scale graph representations without significantly compromising their topological properties. This makes the resulting graphs widely applicable for efficient network-wide analysis. We demonstrated the method on the open street map (OSM) network of Amsterdam with four different application cases. The results show that the method successfully reduces the Amsterdam network by up to 96% of its original size at a computation time of no more than 15 minutes with a limited loss of information, indicated by the preservation of key network characteristics. For example, the method maintains trip length distributions and limits the maximum shortest path deterioration between any major origin and destination nodes to no more than 0.025% for the coarsest graph. Moreover, by setting its parameters it can preserve important network elements or entire sub-networks, which is of special importance in multiscale traffic modeling and simulation. To support further research, an open-source implementation of the algorithm is made available.

The massive spatio-temporal data collected from sensors in the urban cities are addressed in **chapter 3**. This data can provide comprehensive traffic state conditions for an urban network and a particular day. However, they are often too numerous and too detailed to be of direct use. Hence, 3D speed cluster maps are used to represent the spatio-temporal relations of this data in a compact form. Three partitioning techniques are used to build these cluster maps: Normalized cut, DBSCAN and Growing Neural Gas (GNG). A new post-treatment methodology is introduced for DBSCAN and GNG, which are based on data point clustering, to generate connected zones. Furthermore, we investigate the day-to-day regularity of urban congestion patterns using these 3D clusters. The days with similar patterns are clustered together, and then we use the consensus method to produce a unique global pattern that fits multiple days, uncovering the day-to-day regularity. We show that the network of Amsterdam over 35 days can be synthesized into only 4 consensual 3D speed maps with 9 clusters. This paves the way for a cutting-edge systematic method for travel time predictions in cities. By matching the current observation to historical consensual 3D speed maps, we design an efficient real-time method that successfully predicts 84% of trip's travel times with an error margin below 25%.

Instead of ploughing through the raw detector data, we have shown that it is possible to search through higher-level traffic congestion patterns. In **chapter 4**, we further expand on this idea by using shapes as the high-level features to represent the traffic congestion patterns. We extract and store individual traffic congestion patterns as images. The images are manually classified by experts based on the type of congestion and it's extent in space and time. We then extract contours that represent the congestion pattern from these classified images. These contours are then used to build a Statistical Shape Model for each class. Finally, we use a multiclass Active Shape Model algorithm coupled with sequential classification to classify these patterns. We demonstrate the method using one month of loop detector data for two corridors in the Netherlands. The method achieved a classification accuracy of 70% by using just two archetype shapes and simple logistic classifiers.

Chapter 5 proposes another approach for representing the traffic pattern, with inspiration from human vision, where we encode traffic information as images. We used a deep learning model to extract information from the encoded images due to its popularity in artificial vision. We employed a pre-trained deep convolutional neural network as a feature extractor to determine whether the features learned from natural images are sufficient to represent traffic states as well. Since our objective was exploration and not accuracy rates, we used transfer learning to fully exploit the pre-trained models. We demonstrate the method on a large-scale urban network of Amsterdam with one month of travel time data. Experimental results show how the extracted feature vectors cluster naturally into meaningful network traffic states where low, medium and high congestion are distinctly recognized; using these network states for traffic state prediction obtained an accuracy rate of 58% using a naive classification approach.

Leveraging the compact representation of the massive data over the entire network using 3D maps, we proposed three applications in the next three chapters. Firstly, we used the 3D supply patterns to estimate the origin-destination (OD) matrix. The fundamental challenge of the OD matrix estimation problem is that it is severely under-determined. In **chapter 6**, we proposed a new data-driven OD estimation method for cases where a supply pattern in the form of speeds and flows is available. The method consists of three main ingredients: (a) a method to estimate/predict production and attraction time series; (b) a method to compute the N shortest paths from each OD zone to the next; and (c) two behavioral assumptions about the magnitude of N and the proportionality of path flows between these origins and destinations. For large networks, these ingredients may be insufficient to solve the resulting system of equations. Thus, we show how additional constraints can be derived directly from the data by using principal component analysis, with which we exploit the fact that temporal patterns of production and attraction are similar across the network. In this work, we used the 3D supply patterns for predicting the production and attraction patterns, thus revealing a correlation between demand and supply. Moreover, we used these 3D maps to obtain the travel time at different time periods for the whole network to compute the path proportionality. Experimental results on a toy network and a large city network (Santander, Spain) show that our OD estimation method provides a minimum error of $\approx 20\%$ for 0-10% error in production and attraction accuracy; this is promising, given that no equilibrium assignment or network loading model was needed and that there is only a minimum number of assumptions involved in the approach.

Secondly, we extend the shape-based approach to understand the network-wide traffic dynamics. The biggest challenge of analysing network traffic dynamics of large-scale networks is its complexity and pattern interpretability. In **chapter 7**, we present a new computationally efficient method, inspired by human vision, to create a compact, scalable and interpretable custom feature vector to represent network traffic dynamics. This is done by extracting pockets of congestion that encompass connected 3D subnetworks as 3D shapes. We then parameterize these 3D shapes as 2D projections and construct parsimonious feature vectors from these projections. There are various applications of these feature vectors such as revealing the day-to-day regularity of the congestion patterns and building a classification model that allows us to predict travel time from any origin to any destination in the network. We demonstrate that our method achieves a 44% accuracy improvement when compared against the benchmark consensus method for travel prediction of an urban network of Amsterdam. Furthermore, we demonstrate the scalability of the approach by applying the method on the entire Dutch highway network and show that the feature vector was able to encapsulate the network dynamics with a 93% prediction accuracy. There are many paths to further refine and improve the method. The compact form of the feature vector allows us to efficiently enrich it with more information such as context, weather and event without increasing the computational complexity.

Finally, we extend the applications of such 3D maps to other modes in **chapter 8**. The aim was to represent the network dynamics of a transport service in a similar 3D compact form, which can be exploited to reveal patterns within the data. Smart card data enables the estimation of passenger delays throughout the public transit network. However, this delay is measured per passenger trajectory and not per network component. The implication is that it is currently not possible to identify the contribution of individual system components – stations and track segments – to overall passenger delay and thus prioritize investments and disruption management measures accordingly. To this end, we propose a novel method for attributing passenger delays to individual transit network elements from individual passenger trajectories. We decompose the delay along a passenger trajectory into its corresponding track segment delay, initial waiting time and transfer delay. Using these delay components, we construct a solvable system of equations, using which the delays on each network component can be computed. The estimation method is demonstrated on one year of data from the Washington DC metro network. Our approach produces promising results by compressing millions of individual trajectories into 3D networks, leading to a dimensionality reduction of 94%. Moreover, the mean slack variable value (that can be interpreted as proxies for estimation errors) is smaller than five seconds per passenger, and has the desired positive sign for almost 90% of all travelers. Applications using the estimation results include revealing network-wide recurrent delay patterns, modeling delay propagation and detecting disruptions.

9.2 Scientific contributions

We have grouped the main contributions of the thesis into several topics.

- *Insights on networks.* Networks are the cornerstone of any transportation system. We proposed a coarsening approach in **chapter 2** to reduce the complexity of these networks, as they are the first roadblock to a network-wide data-driven analysis of these systems, especially for large urban metropolitan networks, as shown in **chapters 3** and **7**. This was further confirmed in **chapter 6**, where the biggest bottleneck of the OD estimation approach was the computation of N^* shortest paths in the network. Coarsened multiscale networks would improve this computation exponentially. Moreover, the importance of incorporating space to represent the traffic dynamics was substantially proven in this thesis. By adding topology information, we were able to achieve higher production and attraction prediction accuracy within the OD estimation framework in **chapter 6** than using simple vectorized traffic information. Moreover, using feature vectors that incorporate space was proven to be scale-invariant in **chapter 7** and hence highly relevant for large-scale applications, unlike vectorized information where the dimensionality increases with an increase in network size.

- *Insights on feature selection.* Feature selection was proven to be extremely important in the machine learning approach. In this thesis, we further cement their importance for transportation and for interpretability. In **chapter 3**, we used clustering results to distinguish between different days, whereas in **chapters 4** and **7**, we incorporated domain knowledge to define the feature vector by using shapes (combined with the speed and flow values of the given shape) to define different congestion patterns. Furthermore, in **chapter 5**, we used features from computer vision to define traffic patterns. Some of these features are not scalable with respect to the network size; however, all of them have the advantage of being explainable. This makes it possible to ask why certain specific days are clustered together and why some of them are not based on the features we selected. In **chapter 5**, we showed the success of the pre-trained deep learning network in distinguishing between different types of network traffic states. This paves the way for identifying additional relevant features from computer vision to define these traffic states, other than shape. Moreover, in **chapters 6** and **7**, we showed that simple and scale-invariant feature vectors perform better than vectorized time series, thus confirming the power of choosing appropriate features.

- *Insights on traffic patterns and their predictability.* We have proposed different methods to define and reveal traffic patterns. The new concept of consensual 3D speed maps proposed in **chapter 3** allows us to extract the essence from large amounts of link speed observations; as a result, it reveals a global and previously hidden picture of traffic dynamics at the whole city scale, which may be more regular and predictable than expected. The new method of using shapes allows us to look at high-level scale-invariant features to define traffic dynamics, as shown in **chapters 4** and **7**. The success of using such features also paved the way for using other high-level features inspired by human vision to define and distinguish between different network dynamics, which was also found to be fruitful in **chapter 5**. This way of compressing the spatio-temporal data of one day into 3D maps, and thus compressing multiple days of data to reveal daily or weekly patterns, is a giant leap from data to information and ultimately to insights about how the city moves. The studies revealed a strong recurrence of traffic patterns when we look at a higher abstraction level, leading to the conclusion that these patterns are predictable.

- *Insights on demand and supply relationship.* Previous research has not identified a relationship between traffic supply and demand. In **chapter 6**, however, we proposed a simple supervised neural net which revealed that there is a strong correlation between supply and demand; thus, we were able to predict production and attraction patterns from supply patterns represented using 3D speed and flow maps. This opens up many avenues of research into deducing the exact relationship, whether it be linear or non-linear. This is extremely useful since supply information on road networks can be observed directly while demand data is harder to measure.

- *Insights on estimation.* In this thesis, we have shown that given sufficient data, we can formulate the problem in such a way that it is solvable for unknown variables. In **chapter 6**, we estimated the OD matrix by constructing equations using the travel time whereas, in **chapter 8**, we used the passenger trajectory from smart card data to estimate the passenger delay for different network elements. This method of problem formulation can be used to solve data estimation, especially where sufficient data is available. We have also shown in **chapter 8** that limited availability of data leads to more unreliability in the estimation. However, we circumvented the problem of under-determinedness for large-scale networks by incorporating domain knowledge. In **chapter 6**, we used PCA to estimate the OD matrix for large scale networks in a step-wise fashion. Thus, given sufficient data and good problem formulation, we can estimate traffic variables with reasonable confidence. This can be extended for the estimation of unknown variables for other modes as well.

9.3 Practical contributions

As described in chapters 6,7 and 8, our pattern recognition methods have been applied to several real-world cases with different levels of detail, for different modes, road types and network sizes. This demonstrates that the above-mentioned contributions perform well and are mature enough to be applied in practice. Some of the contributions that can be directly used in practice are as follows:

- *Multiscale networks.* The open-source implementation of our proposed coarsening approach aids researchers and practitioners in building multiscale networks without compromising key topological characteristics. This is applicable for any problem that requires complexity reduction and that can be defined using a graph representation, such as road, freight, marine or power grids. The applications for such multiscale graphs range from visualization to hybrid modeling.

- *3D maps.* The 3D maps proposed in this work can be used to represent and visualize spatio-temporal data in a compact form for different modes. We have shown how this can be done for road and public transit networks. However, it can also be used for any spatio-temporal data such as active mode, freight or on-demand service. It can be used for monitoring and analyzing the traffic data ex-post. Moreover, the 3D maps can be used as a carrier to share information between different modes, and thus they are extremely relevant for multimodal studies.

- *Traffic patterns.* We compressed multiple days into a handful of classes that reveal regular and irregular patterns. This can be incorporated into online traffic management services as the methods are efficient and fast for travel time predictions, as already demonstrated. Furthermore, the network traffic patterns that we have identified through clustering can be used as scenarios for traffic control strategies, as the network states in these classes are representative of the city dynamics across different days.

- *Custom feature vectors.* We used various high-level features to define the network traffic states. These customized feature vectors can easily be extended to include context information. Thus, they can be used for querying through a large amount of traffic data instead of ploughing through raw data. This can help build a new type of intelligent traffic database, in which each day is labeled using the custom features coupled with the context information and different days with similar feature vector can be identified rapidly.

- *Data-driven OD estimation framework.* The data-driven OD estimation approach fits in nicely with an online DTA modeling and management machinery, as it does not require iterative use of simulation models to reach equilibrium assignment.

- *Passenger delay.* The 3D passenger delay can be used as a network performance indicator that has applications such as estimating the passenger delay for any trajectory in the network for compensation purposes, detecting disruptions and predicting delay propagation. The 3D representation also leads to data compression as instead of saving individual passenger trajectories, the 3D map is sufficient to represent a day of data. Furthermore, our study revealed daily and weekly patterns, that can be used to plan and manage transit schedules based on data with the aim of improving customer satisfaction.

- *Hierarchical data structure.* The multiscale networks and the custom feature vector formulation, which can easily be extended to include more details, make it possible to build a hierarchy of intermediate results for both road and public transit networks. This allows for analyzing the network at different levels of detail, which is especially important for online applications.

9.4 Recommendations

We believe that the approaches and insights from this thesis provide fruitful directions for continuing research in the context of large-scale network traffic analysis. To conclude this thesis, we present a set of recommendations for future work in this field:

- *Evolution of the data-driven models.* The network traffic patterns analyzed in this work are based on historical data. For the models to evolve to new data, the methods proposed in this work need to be adapted. One of the easiest ways is to retrain the model with the new data. However, this implies that the insights obtained from the old model are completely discarded. A more intelligent method needs to be built to adapt the old model with the new data rather than retraining the model again. I believe evolutionary algorithms and reinforcement learning can aid in building such data-driven models.

- *Incorporating irregular days into the models.* One of the disadvantages of aggregating multiple days into different groups is that we generalize the findings. Thus, the irregular days represented in the data are considered as outliers and will bias the clustering. I believe that we can use probabilistic classification to detect such irregular days and treat them differently than the regular days. Further research is needed to support this theory and to determine whether such a step will improve the performance of the clustering in terms of homogeneity within classes.

- *Class interpretability.* We have extensively looked into various approaches to extract patterns from the data and cluster them into different groups. However, we have not analyzed these classes in detail to draw any conclusions about what they represent. We were able to visually explain the classes found in the transfer learning approach. However, the 3D zones and 3D shapes in each class were not analyzed in detail to understand what these zones or shapes in different classes represent.

- *Feature interpretability.* The custom feature vector formulation incorporates domain knowledge to define the network state. Thus, these feature vectors are self-explanatory. However, the features used in the pre-trained deep learning neural net that was able to successfully distinguish different traffic states were not studied considerably, as it was a black-box method. I believe that by disentangling dominant features from the neural net that make the distinction between different traffic states, we can extract high-level features other than shapes to define a network traffic state. The progress in model explainability research of deep neural nets is a promising avenue that can aid in this.

- *Applying results in practice.* In Part III, we have explored several applications of extracting the essence of the traffic patterns, such as predicting travel time and revealing the demand and supply relationships. However, there are many promising avenues to apply these research results for different networks and modes. Especially for the 3D passenger delay, straightforward applications include disruption detection and delay prediction.

- *Extending for multimodal networks.* We showed the potential of applying the 3D maps for public transit networks. This opens up quite a few possibilities for combining the 3D supply maps from road networks and the 3D passenger delay from public transport networks to build a multimodal network where we can define the interaction between different modes using virtual links or nodes. If we can extend these 3D maps to other modes, we can further enrich these multimodal networks.

- *Diving into data-driven OD estimation.* We only scratched the surface with our work in OD estimation. There are many promising avenues of research to further enrich and improve the proposed framework. Some of the key research directions are looking into OD-specific N^*, improving production and attraction predictions, and employing better verification measures for OD matrix estimation.

- *Augmenting traffic data with socio-economic data.* In the age when the full power of data, both its positive and negative aspects, is being revealed, it is extremely important to consider the ethical implications of any data-driven methods that are relevant for practice. When we analyze data, we only see numbers without realizing those numbers are actually people, especially in the field of transportation. As researchers, we need to be vigilant in complementing the insights derived from the data with demographic and socioeconomic context so that it is a fully rounded analysis. This is especially relevant to OD estimation, where using the OD estimates blindly without additional information might lead to less investment for areas that actually need it.

With the recent advances in processing power, combined with the increasing availability of data in the transportation field and the widespread application of pattern recognition outside the computer science domain, the time seems ripe for using these approaches to convert these data into insights. There is already an increased acceptance of using machine learning as a worthy and effective method for network-wide analysis of traffic patterns, and this thesis can propel that forward.

Appendix A

Derivation

The system of equations given in (6.13), (6.14), and (6.15) is obtained from equations (6.6), (6.7), (6.11), and (6.12). Here, we give a detailed derivation of the system of equations and formulate it as a matrix equality for solving it. First, we reiterate the equations used:

$$P_{ik} = \sum_j \sum_n x_{ijk}^n \tag{A.1}$$

$$A_{jk} = \sum_i \sum_n x_{ijk}^n \tag{A.2}$$

$$x_{ijk}^n = \beta_{ijk}^n x_{ijk}, \forall, i, j, k, \text{and } n \in \{1, \ldots, N_{ijk}^*\} \tag{A.3}$$

$$y_k^m = \sum_i \sum_j \sum_{p_{ijl}^n \in P_k^m} x_{ijl}^n \tag{A.4}$$

Equation A.1 can be expanded as follows:

$$P_{ik} = \sum_n x_{i1k}^n + \sum_n x_{i2k}^n + \cdots + \sum_n x_{ijk}^n + \ldots \tag{A.5}$$

Equation A.5 can be redefined as sum of OD flows instead of path flows based on equation A.3 as follows:

$$P_{ik} = \sum_n \beta_{i1k}^n x_{i1k} + \sum_n \beta_{i2k}^n x_{i2k} + \cdots + \sum_n \beta_{ijk}^n x_{ijk} + \ldots \tag{A.6}$$

Since the route proportion β_{i1k}^n is summed across all routes n ranging from 1 to N_{ijk}^*, it can be reformulated as follows:

$$\sum_n \beta_{ijk}^n = \begin{cases} 0 & TT_{ijk}^n = \infty \\ 1 & \text{otherwise} \end{cases} \tag{A.7}$$

where $TT_{ijk}^n = \infty$ implies that no path exists between origin i and destination j. Thus, equation A.6 can be written as:

$$P_{ik} = x_{i1k} + \cdots + x_{ijk} + \ldots \tag{A.8}$$

with example $\sum_n \beta^n_{i1k} = 1$, $\sum_n \beta^n_{i2k} = 0$ and $\sum_n \beta^n_{ijk} = 1$.

Equations A.5 and A.6 are reproduced for equation A.2 as given below:

$$A_{jk} = \sum_n x^n_{1jk} + \sum_n x^n_{2jk} + \cdots + \sum_n x^n_{ijk} + \ldots \tag{A.9}$$

$$A_{jk} = \sum_n \beta^n_{1jk} x_{1jk} + \sum_n \beta^n_{2jk} x_{2jk} + \cdots + \sum_n \beta^n_{ijk} x_{ijk} + \ldots \tag{A.10}$$

Using equation A.7, A.10 can rewritten as:

$$A_{jk} = x_{1jk} + \cdots + x_{ijk} + \ldots \tag{A.11}$$

with example $\sum_n \beta^n_{1jk} = 1$, $\sum_n \beta^n_{2jk} = 0$ and $\sum_n \beta^n_{ijk} = 1$.

The link count given in A.4 can be expanded as:

$$y^m_k = \sum_{p^n_{11l} \in P^m_k} x^n_{11l} + \sum_{p^n_{12l} \in P^m_k} x^n_{12l} + \cdots + \sum_{p^n_{ijl} \in P^m_k} x^n_{ijl} + \ldots \tag{A.12}$$

A.12 can rewritten in terms of OD flows using equation A.3 as follows:

$$y^m_k = \sum_{p^n_{11l} \in P^m_k} \beta^n_{11l} x_{11l} + \sum_{p^n_{12l} \in P^m_k} \beta^n_{12l} x_{12l} + \cdots + \sum_{p^n_{ijl} \in P^m_k} \beta^n_{ijl} x_{ijl} + \ldots \tag{A.13}$$

This term is considered if and only if (a) link e_m is not congested in period k, and (b) none of the links upstream of e_m in the set of paths $p^n_{ijl} \in P^m_k, l \leq k - \overline{TT}^{n|m}_{ijk}, \forall i, j, n$ (i.e. all paths that traverse e_m during period k) were congested. $\overline{TT}^{n|m}_{ijk}$ depicts the partial arrival travel time (i.e. up to link e_m) along path p^n_{ijl} when traversing link e_m in period k. Thus,

$$\sum_{p^n_{ijl} \in P^m_k} \beta^n_{ijl} x_{ijl} = \begin{cases} 0 & \text{links upstream of } e_m \in p^n_{ijl} \text{ is congested} \\ \beta^n_{ijk} x_{ijk} & \text{otherwise} \end{cases} \tag{A.14}$$

Based on A.14, A.13 can be rewritten as:

$$y^m_k = \beta^n_{11k} x_{11k} + \cdots + \beta^n_{ijk} x_{ijk} + \ldots \tag{A.15}$$

assuming $\sum_{p^n_{12l} \in P^m_k} \beta^n_{12l} x_{12l} = 0$

Combining the expanded form of production, attraction and link counts given in A.8, A.11 and A.15 respectively, the system of equations can be formulated as:

$$x_{i1k} + \cdots + x_{ijk} + \ldots = P_{ik} \tag{A.16}$$

$$\vdots$$

$$x_{1jk} + \cdots + x_{ijk} + \ldots = A_{jk} \tag{A.17}$$

$$\vdots$$

$$\beta^n_{11k} x_{11k} + \cdots + \beta^n_{ijk} x_{ijk} + \ldots = y^m_k \tag{A.18}$$

$$\vdots$$

We can reformulate the system of equations as a matrix equation as:

$$C\mathbf{x} = B, \;\; \mathbf{x} \geq 0; \tag{A.19}$$

where

$$C = \begin{bmatrix} \sum_n \beta_{11k}^n & \cdots & 1 & \cdots & \sum_n \beta_{j1k}^n & \cdots & 1 & \cdots \\ \vdots & \ddots & & & & & & \\ \sum_n \beta_{11k}^n & \cdots & \sum_n \beta_{i1k}^n & \cdots & 1 & \cdots & 1 & \cdots \\ \vdots & \ddots & & & & & & \\ \beta_{11k}^n & \cdots & \sum_{p_{i1l}^n \in P_k^m} \beta_{i1l}^n & \cdots & \sum_{p_{1jl}^n \in P_k^m} \beta_{1jl}^n & \cdots & \beta_{ijk}^n & \cdots \\ \vdots & \ddots & & & & & & \end{bmatrix}; \mathbf{x} = \begin{bmatrix} x_{11k} \\ \vdots \\ x_{i1k} \\ \vdots \\ x_{1jk} \\ \vdots \\ x_{ijk} \\ \vdots \end{bmatrix}; \text{and } B = \begin{bmatrix} P_{ik} \\ \vdots \\ A_{jk} \\ \vdots \\ y_k^m \\ \vdots \end{bmatrix} \tag{A.20}$$

Nomenclature

$G(S,E,L,S^{trans})$	Directed graph representing public transport network
x^n_{ijk}	path flows between $v_i, v_j \in Z$ for travelers departing in period k with $n = 1,\ldots,N_{ijk}$ paths
S	Set of stations $\{s_1, s_2, ...\}$
x_{ijk}	OD flows between $v_i, v_j \in Z$ for travelers departing in period k
E	Set of ordered pair of stations representing the track segments between the stations $\{(s_1,s_2),(s_2,s_3),...\}$
P_{ik}	production (sum of all outgoing OD flows) of zone i during period k
A_{jk}	attraction (sum of all incoming OD flows) of zone j during period k
L	Set of ordered pair of stations that defines a directed public transport service $\{l_1, l_2, ...\}$
S^{trans}	Set of transfer or interchange stations where transfer between lines occur $\{s_i, s_{i+1}, ...\}$, where $S^{trans} \subset S$
TT^n_{ijk}	travel time for vehicles traversing path n between node i, j departing in period k
β^n_{ijk}	route proportion
s_o	Origin station of a passenger journey
s_d	Destination station of a passenger journey
y^m_k	link count for link e_m in period k
FV	speed-flow based (SF) feature vector
$\hat{FV}$	speed-flow-topology (SFT) based feature vector
r_{s_o,s_d}	Shortest trajectory between origin station s_o and destination station s_d, where $s_o, s_d \in S$
$\widetilde{t}_{s_o,s_d,k}$	Scheduled travel time between origin station s_o and destination station s_d departing at period k

$\widetilde{t}^{veh}_{s_i,s_{i+1},k}$ Scheduled transit vehicle run-time and dwell time between track segment (s_i, s_{i+1}) departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$h_{l,k}$ Headway between transit vehicles for a given line $l \in L$ departing within period k

$t^{walk}_{s_i}$ Walking time between gate and platform or vice-versa for each station $s_i \in S$ and if $s_i \in S^{trans}$, the walking time refers to walking time between line directions

$r_{s_o,s_d,v}$ Trajectory traversed by a transit vehicle v between origin station s_o and destination station s_d, where $s_o, s_d \in S$ and $v \in \{1...V\}$

$r_{s_o,s_d,n}$ Inferred trajectory of a passenger n between origin station s_o and destination station s_d, where $s_o, s_d \in S$ and $n \in \{1...N\}$

$t_{s_o,s_d,k,n}$ Observed travel time between origin station s_o and destination station s_d for passenger n departing at period k

$t^{veh}_{s_i,s_{i+1},k,v}$ Observed transit vehicle run-time and dwell time between track segment (s_i, s_{i+1}) for a vehicle v departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$d_{s_o,s_d,k,n}$ Estimated average passenger delay between origin station s_o and destination station s_d for passenger n departing at period k

$d^{veh}_{s_o,s_d,k,v}$ Estimated transit vehicle delay between origin station s_o and destination station s_d departing at period k for a given vehicle v servicing a line

$d^{on-board}_{s_i,s_{i+1},k}$ Estimated transit vehicle-run or on-board delay between track segment (s_i, s_{i+1}) departing at period k, where $(s_i, s_{i+1}) \in E$ and $s_i, s_{i+1} \in S$

$d^{wait}_{s_o,k}$ Estimated initial waiting time delay at origin station s_o departing at period k

$d^{trans}_{s_i,k}$ Estimated transfer delay at transfer station s_i departing at period k

Bibliography

[1] *Road crash costs*. SWOV Fact sheet, July 2017. The Hague.

[2] Yufei Yuan. Lagrangian multi-class traffic state estimation. 2013.

[3] Eleni I Vlahogianni, John C Golias, and Matthew G Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews*, 24(5):533–557, 2004.

[4] JWC Van Lint and CPIJ Van Hinsbergen. Short-term traffic and travel time prediction models. *Artificial Intelligence Applications to Critical Transportation Issues*, 22(1): 22–41, 2012.

[5] Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Short-term traffic forecasting: Where we are and where were going. *Transportation Research Part C: Emerging Technologies*, 43:3–19, 2014.

[6] Simon Oh, Young-Ji Byon, Kitae Jang, and Hwasoo Yeo. Short-term travel-time prediction on highway: a review of the data-driven approach. *Transport Reviews*, 35 (1):4–32, 2015.

[7] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of modern physics*, 73(4):1067, 2001.

[8] Denos C Gazis, Robert Herman, and Renfrey B Potts. Car-following theory of steady-state traffic flow. *Operations research*, 7(4):499–505, 1959.

[9] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):105–111, 1981.

[10] Ilya Prigogine and Robert Herman. Kinetic theory of vehicular traffic. Technical report, 1971.

[11] Dirk Helbing and Martin Treiber. Gas-kinetic-based traffic model explaining observed hysteretic phase transition. *Physical Review Letters*, 81(14):3042, 1998.

[12] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229, 1992.

[13] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 229(1178):317–345, 1955.

[14] Harold J Payne. Model of freeway traffic and control. *Mathematical Model of Public System*, pages 51–61, 1971.

[15] GB Whitham. *Linear and nonlinear waves*. Modern Book Incorporated, 1975.

[16] JW Godfrey. The mechanism of a road network. *Traffic Engineering & Control*, 8 (8), 1969.

[17] H Mahmassani, James C Williams, and R Herman. Performance of urban traffic networks. In *Proceedings of the 10th International Symposium on Transportation and Traffic Theory*, pages 1–20. Elsevier Amsterdam, The Netherlands, 1987.

[18] Nikolas Geroliminis and Carlos F Daganzo. Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings. *Transportation Research Part B: Methodological*, 42(9):759–770, 2008.

[19] Ludovic Leclercq, Nicolas Chiabaut, and Béatrice Trinquier. Macroscopic fundamental diagrams: A cross-comparison of estimation methods. *Transportation Research Part B: Methodological*, 62:1–12, 2014.

[20] Mehmet Yildirimoglu, Mohsen Ramezani, and Nikolas Geroliminis. Equilibrium analysis and route guidance in large-scale networks with mfd dynamics. *Transportation Research Procedia*, 9:185–204, 2015.

[21] Ibai Lana, Javier Del Ser, Manuel Velez, and Eleni I Vlahogianni. Road traffic forecasting: recent advances and new challenges. *IEEE Intelligent Transportation Systems Magazine*, 10(2):93–109, 2018.

[22] Alireza Ermagun and David Levinson. Spatiotemporal traffic forecasting: review and proposed directions. *Transport Reviews*, 38(6):786–814, 2018.

[23] Jaimyoung Kwon and Kevin Murphy. Modeling freeway traffic with coupled hmms. Technical report, Technical report, Univ. California, Berkeley, 2000.

[24] John Rice and Erik Van Zwet. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3): 200–207, 2004.

[25] CP IJ van Hinsbergen and JWC Van Lint. Bayesian combination of travel time prediction models. *Transportation Research Record*, 2064(1):73–80, 2008.

[26] Mohammed S Ahmed and Allen R Cook. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Number 722. 1979.

[27] Jianhua Guo, Wei Huang, and Billy M Williams. Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, 43:50–64, 2014.

[28] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[29] Yufei Yuan, Aurélien Duret, and Hans Van Lint. Mesoscopic traffic state estimation based on a variational formulation of the lwr model in lagrangian-space coordinates and kalman filter. *Transportation Research Procedia*, 10:82–92, 2015.

[30] Herman Sutarto, Endra Joelianto, and Taufik Sugian. Estimation and prediction of road traffic flow using particle filter for real-time traffic control. In *Conference on Control, systems and Industrial Informatics (ICCSII 2013)*. IEEE, 2013.

[31] Yanyan Xu, Qing-Jie Kong, Reinhard Klette, and Yuncai Liu. Accurate and interpretable bayesian mars for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2457–2469, 2014.

[32] Yuliang Cong, Jianwei Wang, and Xiaolei Li. Traffic flow forecasting by a least squares support vector machine with a fruit fly optimization algorithm. *Procedia Engineering*, 137:59–68, 2016.

[33] Xingxing Xing, Xiabing Zhou, Haikun Hong, Wenhao Huang, Kaigui Bian, and Kunqing Xie. Traffic flow decomposition and prediction based on robust principal component analysis. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2219–2224. IEEE, 2015.

[34] D Nikovski, N Nishiuma, Y Goto, and H Kumazawa. Univariate short-term prediction of road travel times. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 1074–1079. IEEE, 2005.

[35] Shiliang Sun and Changshui Zhang. The selective random subspace predictor for traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, 8 (2):367–373, 2007.

[36] Gary A Davis and Nancy L Nihan. Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering*, 117(2):178–188, 1991.

[37] Zuduo Zheng and Dongcai Su. Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm. *Transportation Research Part C: Emerging Technologies*, 43:143–157, 2014.

[38] Brian L Smith and Michael J Demetsky. Short-term traffic flow prediction models-a comparison of neural network and nonparametric regression approaches. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 1706–1709. IEEE, 1994.

[39] JWC Van Lint, SP Hoogendoorn, and Henk J van Zuylen. Accurate freeway travel time prediction with state-space neural networks under missing data. *Transportation Research Part C: Emerging Technologies*, 13(5-6):347–369, 2005.

[40] Matthew G Karlaftis and Eleni I Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3):387–399, 2011.

[41] Fabio Moretti, Stefano Pizzuti, Stefano Panzieri, and Mauro Annunziato. Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling. *Neurocomputing*, 167:3–7, 2015.

[42] N Polson and V Sokolov. Deep learning predictors for traffic flows. *arXiv preprint arXiv:1604.04527*, 2016.

[43] Nicholas G Polson and Vadim O Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 2017.

[44] Yi Hou and Praveen Edara. Network scale travel time prediction using deep learning. *Transportation Research Record*, 2672(45):115–123, 2018.

[45] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *arXiv preprint arXiv:1802.07007*, 2018.

[46] Marie-Pier Pelletier, Martin Trépanier, and Catherine Morency. Smart card data use in public transit: A literature review. *Transportation Research Part C: Emerging Technologies*, 19(4):557–568, 2011.

[47] George Karypis and Vipin Kumar. Multilevel graph partitioning schemes. In *ICPP (3)*, pages 113–122, 1995.

[48] Cédric Chevalier and Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *International Conference on Learning and Intelligent Optimization*, pages 191–205. Springer, 2009.

[49] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[50] Peter Sanders and Dominik Schultes. Engineering highway hierarchies. *Journal of Experimental Algorithmics (JEA)*, 17:1–6, 2012.

[51] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. *Algorithmics of large and complex networks*, 5515:117–139, 2009.

[52] Zhiguang Cao, Hongliang Guo, Jie Zhang, Dusit Niyato, and Ulrich Fastenrath. Finding the shortest path in stochastic vehicle routing: A cardinality minimization approach. *IEEE Transactions on Intelligent Transportation Systems*, 17(6):1688–1702, 2016.

[53] Bruce N Janson. Dynamic traffic assignment for urban road networks. *Transportation Research Part B: Methodological*, 25(2):143–161, 1991.

[54] Ehsan Jafari and Stephen D. Boyles. Improved bush-based methods for network contraction. *Transportation Research Part B: Methodological*, 83(Supplement C): 298–313, 2016.

[55] Richard D. Connors and David P. Watling. Assessing the demand vulnerability of equilibrium traffic networks via network aggregation. *Networks and Spatial Economics*, 15(2):367–395, 2015.

[56] Stephen D. Boyles. Bush-based sensitivity analysis for approximating subnetwork diversion. *Transportation Research Part B: Methodological*, 46(1):139–155, 2012.

[57] R. D. Connors and D. P. Watling. Aggregation of transport networks using sensitivity analysis. *European Transport Conference 2008; Proceedings*, page 13p, 2008.

[58] Mahtab Joueiai, Ludovic Leclercq, Hans van Lint, and Serge P. Hoogendoorn. Multiscale traffic flow model based on the mesoscopic lighthill-whitham and richards models. *Transportation Research Record*, (2491):98–106, 2015.

[59] Y. Xu, W. Cai, H. Aydt, and M. Lees. Efficient graph-based dynamic load-balancing for parallel large-scale agent-based traffic simulation. volume 2015-January, pages 3483–3494, 2015. doi: 10.1109/WSC.2014.7020180. cited By 4.

[60] Moshe Ben-Akiva, Michel Bierlaire, Didier Burton, Haris N Koutsopoulos, and Rabi Mishalani. Network state estimation and prediction for real-time traffic management. *Networks and spatial economics*, 1(3-4):293–318, 2001.

[61] Yufei Yuan, JWC Van Lint, R Eddie Wilson, Femke van Wageningen-Kessels, and Serge P Hoogendoorn. Real-time lagrangian traffic state estimator for freeways. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):59–70, 2012.

[62] Clélia Lopez, Ludovic Leclercq, Panchamy Krishnakumari, Nicolas Chiabaut, and Hans Lint. Revealing the day-to-day regularity of urban congestion patterns with 3d speed maps. *Scientific Reports*, 7(1):14029, 2017.

[63] Clélia Lopez, Panchamy Krishnakumari, Ludovic Leclercq, Nicolas Chiabaut, and Hans Van Lint. Spatiotemporal partitioning of transportation network using travel time data. *Transportation Research Record: Journal of the Transportation Research Board*, (2623):98–107, 2017.

[64] Xiqun Chen, Shuaichao Zhang, Li Li, and Liang Li. Adaptive rolling smoothing with heterogeneous data for traffic state estimation and prediction. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–12, 2018.

[65] S.D. Dimitrov and A. Ceder. A method of examining the structure and topological properties of public-transport networks. *Physica A: Statistical Mechanics and its Applications*, 451:373–387, 2016. doi: 10.1016/j.physa.2016.01.060. cited By 1.

[66] Dominik Schultes. Route planning in road networks. In *Ausgezeichnete Informatikdissertationen*, pages 271–280, 2008.

[67] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm Engineering*, pages 19–80. Springer, 2016.

[68] Achi Brandt and Dorit Ron. Multigrid solvers and multilevel optimization strategies. *Multilevel Optimization in VLSICAD*, 14:1, 2013.

[69] Ilya Safro, Dorit Ron, and Achi Brandt. Multilevel algorithms for linear ordering problems. *Journal of Experimental Algorithmics (JEA)*, 13:4, 2009.

[70] Achi Brandt. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis*, 10(1):21, 2000.

[71] Sabine Storandt. Contraction hierarchies on grid graphs. In *Annual Conference on Artificial Intelligence*, pages 236–247. Springer, 2013.

[72] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. Customizable contraction hierarchies. *Journal of Experimental Algorithmics (JEA)*, 21(1):1–5, 2016.

[73] Carl Ollivier-Gooch. Coarsening unstructured meshes by edge contraction. *International Journal for Numerical Methods in Engineering*, 57(3):391–414, 2003.

[74] Alexander Gutfraind, Ilya Safro, and Lauren Ancel Meyers. Multiscale network generation. In *2015 18th International Conference on Information Fusion (Fusion)*, pages 158–165. IEEE, 2015.

[75] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. Fast exact shortest path and distance queries on road networks with parametrized costs. *CoRR*, abs/1509.03165, 2015.

[76] Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Customizable route planning in road networks. *Transportation Science*, 51(2):566–591, 2017. doi: 10.1287/trsc.2014.0579.

[77] Stefan Funke, André Nusser, and Sabine Storandt. On k-path covers and their applications. *The VLDB Journal*, 25(1):103–123, Feb 2016.

[78] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[79] Bas Fagginger Auer and Rob H Bisseling. Graph coarsening and clustering on the gpu. *Graph Partitioning and Graph Clustering*, 588:223, 2012.

[80] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:2–2, 2015.

[81] Nikolaj Tatti and Aristides Gionis. Density-friendly graph decomposition. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1089–1099. ACM, 2015.

[82] Luis Moreira-Matias, Joao Mendes-Moreira, Joao Gama, and Michel Ferreira. On improving operational planning and control in public transportation networks using streaming data: A machine learning approach. *ECML/PKDD 2014 PhD Session*, 2014.

[83] Openstreet maps highway tag wiki. http://wiki.openstreetmap.org/wiki/Key:highway.

[84] Wrongly-noded graph in openstreet maps. https://www.gaia-gis.it/fossil/spatialite-tools/wi\ki?name=graphs-intro.

[85] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[86] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[87] Marc Barthelemy. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):163–168, 2004.

[88] Marc Barthélemy. Spatial networks. *Physics Reports*, 499(1):1–101, 2011.

[89] Krzysztof Choromański, Michał Matuszak, and Jacek Mikisz. Scale-free graph with preferential attachment and evolving internal vertex structure. *Journal of Statistical Physics*, 151(6):1175–1183, 2013.

[90] Yuxuan Ji and Nikolas Geroliminis. On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological*, 46(10):1639–1656, 2012.

[91] Mohammadreza Saeedmanesh and Nikolas Geroliminis. Clustering of heterogeneous networks with directional flows based on snake similarities. *Transportation Research Part B: Methodological*, 91:250–269, 2016.

[92] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[93] Jack Haddad, Mohsen Ramezani, and Nikolas Geroliminis. Cooperative traffic control of a mixed network with two urban regions and a freeway. *Transportation Research Part B: Methodological*, 54:17–36, 2013.

[94] Mohsen Ramezani, Jack Haddad, and Nikolas Geroliminis. Dynamics of heterogeneity in urban networks: aggregated traffic modeling and hierarchical control. *Transportation Research Part B: Methodological*, 74:1–19, 2015.

[95] Ying-Ying Ma, Yi-Chang Chiu, and Xiao-Guang Yang. Urban traffic signal control network automatic partitioning using laplacian eigenvectors. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–5. IEEE, 2009.

[96] Srinivas Peeta and Athanasios K Ziliaskopoulos. Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and spatial economics*, 1 (3-4):233–265, 2001.

[97] JW Godfrey. The mechanism of a road network. *Traffic Engineering & Control*, 8 (8), 1969.

[98] H Mahmassani, James C Williams, and R Herman. Performance of urban traffic networks. In *Proceedings of the 10th International Symposium on Transportation and Traffic Theory*, pages 1–20. Elsevier Amsterdam, The Netherlands, 1987.

[99] Nikolas Geroliminis and Carlos F Daganzo. Existence of urban-scale macroscopic fundamental diagrams: Some experimental findings. *Transportation Research Part B: Methodological*, 42(9):759–770, 2008.

[100] Christine Buisson and Cyril Ladier. Exploring the impact of homogeneity of traffic measurements on the existence of macroscopic fundamental diagrams. *Transportation Research Record*, 2124(1):127–136, 2009.

[101] Mehmet Yildirimoglu, Mohsen Ramezani, and Nikolas Geroliminis. Equilibrium analysis and route guidance in large-scale networks with mfd dynamics. *Transportation Research Procedia*, 9:185–204, 2015.

[102] Carlos F Daganzo. Urban gridlock: Macroscopic modeling and mitigation approaches. *Transportation Research Part B: Methodological*, 41(1):49–62, 2007.

[103] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[104] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[105] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in neural information processing systems*, pages 625–632, 1995.

[106] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[107] Thomas Martinetz, Klaus Schulten, et al. A" neural-gas" network learns topologies. 1991.

[108] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel. *Nature*, 439:462–465, 2006.

[109] M.C. Gonzlez, C. Hidalgo, and A.L. Barabsi. Understanding individual mobility patterns. *Nature*, 453:779–782, 2008.

[110] C. Song, T. Koren, P. Wang, and A.L. Barabsi. Modelling the scaling properties of human mobility. *Nature Physics*, 6:818–823, 2010.

[111] C. Song, Z. Qu, N. Blumm, and A.L. Barabsi. Limits of predictability in human mobility. *Science*, 327:1018–1021, 2010.

[112] K. Zhao, M. Musolesi, P. Hui, W. Rao, and S. Tarkoma. Explaining the power-law distribution of human mobility through transportation modality decomposition. *Scientific reports*, 5:srep09136, 2015.

[113] J. Ortuzar and L. Willumsen. Modelling transport. *(Wilsey,Chichester)*, 1994.

[114] A. Wilson. Land-use/transport interaction models:past and future. *Journal of Transportation Economic Policy*, 32:3–26, 1998.

[115] J. Choukroun. A general framework for the development of gravity-type trip distribution models. *Regional Science and Urban Economics*, 5:177–202, 1975.

[116] M. Lenormand, A. Bassolas, and J. J. Ramasco. Systematic comparison of trip distribution laws and models. *Journal of Transport Geography*, 51:158–169, 2016.

[117] C. Peng, X. Jin, K. Wong, M. Shi, and P. Lio. Collective human mobility pattern from taxi trips in urban area. *PLOS ONE*, 7:e34487, 2012.

[118] P. Wang, T. Hunter, A.M. Bayen, K. Schechtner, and M.C. Gonzlez. Understanding road usage patterns in urban areas. *Scientific reports*, 2:srep011001, 2012.

[119] A. Anas. Discrete choice theory, information theory and the multinomial logit and gravity models. *Transportation Research part B*, 17:13–23, 1983.

[120] Y. Yang, C. Herrera, N. Eagle, and M.C. Gonzlez. Limits of predictability in commuting flows in the absence of data calibration. *Scientific Reports*, 4:srep05662, 2014.

[121] E. Alpaydin. *ntroduction to Machine Learning 3rd edition*. MIT Press, 2014.

[122] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(8):888905, 2000.

[123] Y. Ji and N. Geroliminis. On the spatial partitioning of urban transportation networks. *Transportation Research Part B*, 46(10):16391656, 2012.

[124] M. Saeedmanesh and N. Geroliminis. Clustering of heterogeneous networks with directional flows based on snake similarities. *Transportation Research Part B*, 91: 250–269, 2016.

[125] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, July 2006. ISBN 0471241954.

[126] V. Filkov and S. Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(04):863–880, 2004. doi: 10.1142/S0218213004001867.

[127] C. Lopez, P. Krishnakumari, L. Leclercq, N. Chiabaut, and H. van Lint. Spatio-temporal partitioning of the transportation network using travel time data. *Transportation Research Records*, page 14p., 2017. doi: http://dx.doi.org/10.3141/2623-11.

[128] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[129] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[130] Fan Yang, Tao Li, Qifeng Zhou, and Han Xiao. Cluster ensemble selection with constraints. *Neurocomputing*, 235:59 – 70, 2017. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.01.001.

[131] Mete Ozay. Semi-supervised segmentation fusion of multi-spectral and aerial images. In *2014 22nd International Conference on Pattern Recognition*. IEEE, aug 2014. doi: 10.1109/icpr.2014.659.

[132] Eleni I. Vlahogianni, Matthew G. Karlaftis, and John C. Golias. Short-term traffic forecasting: Where we are and where were going. *Transportation Research Part C: Emerging Technologies*, 43:3 – 19, 2014. ISSN 0968-090X. doi: http://dx.doi.org/10.1016/j.trc.2014.01.005. Special Issue on Short-term Traffic Flow Forecasting.

[133] U. Mori, A. Mendiburu, M. lvarez, and J. A. Lozano. A review of travel time estimation and forecasting for advanced traveller information systems. *Transportmetrica A: Transport Science*, 11(2):119–157, 2015. doi: 10.1080/23249935.2014.932469.

[134] Yibing Wang, Markos Papageorgiou, and Albert Messmer. Renaissance - a unified macroscopic model-based approach to real-time freeway network traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 14(3):190–212, 2006.

[135] B. A. Kumar, L. Vanajakshi, and S. C. Subramanian. Bus travel time prediction using a time-space discretization approach. *Transportation Research Part C: Emerging Technologies*, 79:308–332, 2017.

[136] A. Nantes, D. Ngoduy, A. Bhaskar, M. Miska, and E. Chung. Real-time traffic state estimation in urban corridors from heterogeneous data. *Transportation Research Part C: Emerging Technologies*, 66:99–118, 2016. doi: 10.1016/j.trc.2015.07.005.

[137] Xiaoyan Zhang and John A. Rice. Short-term travel time prediction. *Transportation Research Part C: Emerging Technologies*, 11(3-4):187–210, 2003.

[138] John Rice and Erik van Zwet. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3): 200–207, 2004.

[139] S. Biswas, S. Chakraborty, S. Chandra, and I. Ghosh. Kriging-based approach for estimation of vehicular speed and passenger car units on an urban arterial. *Journal of Transportation Engineering*, 143(3), 2017. doi: 10.1061/JTEPBS.0000031.

[140] Y. Xu, H. Chen, Q.-J. Kong, X. Zhai, and Y. Liu. Urban traffic flow prediction: A spatio-temporal variable selection-based approach. *Journal of Advanced Transportation*, 50(4):489–506, 2016. doi: 10.1002/atr.1356.

[141] H. Bahuleyan and L. D. Vanajakshi. Arterial path-level travel-time estimation using machine-learning techniques. *Journal of Computing in Civil Engineering*, 31(3), 2017.

[142] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):2191–2201, oct 2014. doi: 10.1109/tits.2014.2311123.

[143] J. Wang, I. Tsapakis, and C. Zhong. A space-time delay neural network model for travel time prediction. *Engineering Applications of Artificial Intelligence*, 52:145–160, 2016.

[144] J. W. C. Van Lint. Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):38–47, 2008.

[145] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[146] H. Yang, T. S. Dillon, and Y. P. Chen. Optimized structure of the traffic flow forecasting model with a deep learning approach. *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[147] A. Bhaskar, T. Tsubota, L. M. Kieu, and E. Chung. Urban traffic state estimation: Fusing point and zone based data. *Transportation Research Part C: Emerging Technologies*, 48:120–142, 2014.

[148] Li Li, Xiaonan Su, Yanwei Wang, Yuetong Lin, Zhiheng Li, and Yuebiao Li. Robust causal dependence mining in big data network and its application to traffic flow predictions. *Transportation Research Part C: Emerging Technologies*, 58B:292–307, 2015.

[149] Pierre-Antoine Laharotte, Romain Billot, Nour-Eddin El-Faouzi, and Hesham A. Rakha. Network-wide traffic state prediction using bluetooth data. In *TRB 94th Annual Meeting Compendium of Papers, 15-3022*. Transportation Research Board, dec 2015.

[150] G. Fusco, C. Colombaroni, and N. Isaenko. Short-term speed predictions exploiting big data on large urban road networks. *Transportation Research Part C: Emerging Technologies*, 73:183–201, 2016. doi: 10.1016/j.trc.2016.10.019.

[151] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Catherine C. McGeoch, editor, *Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings*, pages 319–333. Springer, Berlin, Heidelberg, 2008.

[152] Cédric Chevalier and Ilya Safro. Comparison of coarsening schemes for multilevel graph partitioning. In *Lecture Notes in Computer Science*, pages 191–205. Springer Berlin Heidelberg, 2009.

[153] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[154] Mo Chen. Kmeans algorithm retrieved on 2016-12-10 from https://fr.mathworks.com/matlabcentral/fileexchange/24616-kmeans-clustering , 2016.

[155] Yarpiz. DBSCAN Clustering Algorithm retrieved on 2016-12-10 from https://fr.mathworks.com/matlabcentral/fileexchange/52905-dbscan-clustering-algorithm , 2016.

[156] Martin Treiber and Dirk Helbing. Reconstructing the spatio-temporal traffic dynamics from stationary detector data. *Cooper@ tive Tr@ nsport@ tion Dyn@ mics*, 1(3): 3–1, 2002.

[157] Thomas Schreiter, Hans van Lint, Martin Treiber, and Serge Hoogendoorn. Two fast implementations of the adaptive smoothing method used in highway traffic state estimation. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1202–1208. IEEE, 2010.

[158] JWC Van Lint and Serge P Hoogendoorn. A robust and efficient method for fusing heterogeneous data from traffic sensors on freeways. *Computer-Aided Civil and Infrastructure Engineering*, 25(8):596–612, 2010.

[159] Thomas Schreiter, Hans Van Lint, Yufei Yuan, and Serge Hoogendoorn. Propagation wave speed estimation of freeway traffic with image processing tools. Technical report, 2010.

[160] Martin Treiber and Arne Kesting. Validation of traffic flow models with respect to the spatiotemporal evolution of congested traffic patterns. *Transportation research part C: emerging technologies*, 21(1):31–41, 2012.

[161] Boris S Kerner, Hubert Rehborn, Mario Aleksic, and Andreas Haug. Recognition and tracking of spatial–temporal congested traffic patterns on freeways. *Transportation Research Part C: Emerging Technologies*, 12(5):369–400, 2004.

[162] Luciano Oliveira Andrews Sobral, Leizer Schnitman, and Felippe De Souza. Highway traffic congestion classification using holistic properties. In *10th IASTED International Conference on Signal Processing, Pattern Recognition and Applications*, 2013.

[163] Xin Jin, Dipti Srinivasan, and Ruey Long Cheu. Classification of freeway traffic patterns for incident detection using constructive probabilistic neural networks. *IEEE Transactions on Neural networks*, 12(5):1173–1187, 2001.

[164] Amina Riaz and Shoab A Khan. Traffic congestion classification using motion vector statistical features. In *Sixth International Conference on Machine Vision (ICMV 2013)*, volume 9067, page 90671A. International Society for Optics and Photonics, 2013.

[165] Thammasak Thianniwet, Satidchoke Phosaard, and Wasan Pattara-Atikom. Classification of road traffic congestion levels from vehicles moving patterns: a comparison between artificial neural network and decision tree algorithm. In *Electronic engineering and computing technology*, pages 261–271. Springer, 2010.

[166] Hong Nam Nguyen, Panchamy Krishnakumari, Hai L Vu, and Hans Van Lint. Traffic congestion pattern classification using multi-class svm. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1059–1064. IEEE, 2016.

[167] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[168] Luc Vincent. Morphological grayscale reconstruction in image analysis: Applications and e cient algorithms. *IEEE transactions on image processing*, 2(2):176–201, 1993.

[169] Dirk Helbing, Martin Treiber, Arne Kesting, and Martin Schönhof. Theoretical vs. empirical classification and prediction of congested traffic states. *The European Physical Journal B*, 69(4):583–598, 2009.

[170] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[171] D. Chetverikov, D. Svirko, D. Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Proceedings - International Conference on Pattern Recognition*, volume 16, pages 545–548, 2002.

[172] Colin Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(2):285–321, 1991.

[173] National datawarehouse of traffic information. http://www.ndw.nu/en/.

[174] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of machine learning research*, 5(Jan):101–141, 2004.

[175] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[176] John Rice and Erik Van Zwet. A simple and effective method for predicting travel times on freeways. *IEEE Transactions on Intelligent Transportation Systems*, 5(3): 200–207, 2004.

[177] Stephen Clark. Traffic prediction using multivariate nonparametric regression. *Journal of transportation engineering*, 129(2):161–168, 2003.

[178] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *IEEE transactions on intelligent transportation systems*, 5(4):276–281, 2004.

[179] Yanyan Xu, Hui Chen, Qing-Jie Kong, Xi Zhai, and Yuncai Liu. Urban traffic flow prediction: a spatio-temporal variable selection-based approach. *Journal of Advanced Transportation*, 50(4):489–506, 2016.

[180] Yunlong Zhang and Hancheng Ge. Freeway travel time prediction using takagi–sugeno–kang fuzzy neural network. *Computer-Aided Civil and Infrastructure Engineering*, 28(8):594–603, 2013.

[181] JWC Van Lint. Online learning solutions for freeway travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):38–47, 2008.

[182] Arief Koesdwiady, Ridha Soua, and Fakhreddine Karray. Improving traffic flow prediction with weather information in connected cars: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 65(12):9508–9517, 2016.

[183] Ren Wang, Shimao Fan, and Daniel B Work. Efficient multiple model particle filtering for joint traffic state estimation and incident detection. *Transportation Research Part C: Emerging Technologies*, 71:521–537, 2016.

[184] Yufei Yuan, JWC Van Lint, R Eddie Wilson, Femke van Wageningen-Kessels, and Serge P Hoogendoorn. Real-time lagrangian traffic state estimator for freeways. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):59–70, 2012.

[185] Chris PIJ van Hinsbergen, Thomas Schreiter, Frank S Zuurbier, JWC Van Lint, and Henk J Van Zuylen. Localized extended kalman filter for scalable real-time traffic state estimation. *IEEE transactions on intelligent transportation systems*, 13(1):385–394, 2011.

[186] Yibing Wang, Markos Papageorgiou, and Albert Messmer. A real-time freeway network traffic surveillance tool. *IEEE Transactions on control systems technology*, 14 (1):18–32, 2005.

[187] Yuxuan Ji and Nikolas Geroliminis. On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological*, 46(10):1639–1656, 2012.

[188] Mohammadreza Saeedmanesh and Nikolas Geroliminis. Clustering of heterogeneous networks with directional flows based on snake similarities. *Transportation Research Part B: Methodological*, 91:250–269, 2016.

[189] Clélia Lopez, Panchamy Krishnakumari, Ludovic Leclercq, Nicolas Chiabaut, and Hans Van Lint. Spatiotemporal partitioning of transportation network using travel time data. *Transportation Research Record*, 2623(1):98–107, 2017.

[190] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors*, 17(4):818, 2017.

[191] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors*, 17(7):1501, 2017.

[192] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[193] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[194] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[195] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[196] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[197] Brian Everitt and Anders Skrondal. *The Cambridge dictionary of statistics*, volume 44. Cambridge University Press Cambridge, 2002.

[198] Anja Struyf, Mia Hubert, Peter Rousseeuw, et al. Clustering in an object-oriented environment. *Journal of Statistical Software*, 1(4):1–30, 1997.

[199] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2017.

[200] Henk J. Van Zuylen and Luis G. Willumsen. The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological*, 14(3):281–293–, 1980. ISSN 0191-2615.

[201] E. Cascetta. Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator. *Transportation Research Part B*, 18(4-5):289–299, 1984. doi: 10.1016/0191-2615(84)90012-2.

[202] Michael G. H. Bell. The real time estimation of origin-destination flows in the presence of platoon dispersion. *Transportation Research Part B: Methodological*, 25 (2-3):115–125, 1991. ISSN 0191-2615. doi: 10.1016/0191-2615(91)90018-E.

[203] R. Kitamura, C. Chen, R.M. Pendyala, and R. Narayanan. Micro-simulation of daily activity-travel patterns for travel demand forecasting. *Transportation*, 27(1):25–51, 2000. doi: 10.1023/A:1005259324588.

[204] C. Bhat and H. Zhao. The spatial analysis of activity stop generation. *Transportation Research Part B: Methodological*, 36(6):557–575, 2002. doi: 10.1016/S0191-2615(01)00019-4.

[205] A. Scheffer, G. Cantelmo, and F. Viti. Generating macroscopic, purpose-dependent trips through monte carlo sampling techniques. In *Transportation Research Procedia*, volume 27, pages 585–592, 2017.

[206] G. Cantelmo, F. Viti, E. Cipriani, and N. Marialisa. A two-steps dynamic demand estimation approach sequentially adjusting generations and distributions. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, volume 2015-October, pages 1477–1482, 2015.

[207] T.A. Arentze and H.J.P. Timmermans. A need-based model of multi-day, multi-person activity generation. *Transportation Research Part B: Methodological*, 43(2): 251–265, 2009. doi: 10.1016/j.trb.2008.05.007.

[208] M. Cremer and H. Keller. A new class of dynamic methods for the identification of origin-destination flows. *Transportation research part B - methodological*, 21(2): 117–132, 1987. ISSN 0191-2615.

[209] H. Yang, T. Sasaki, Y. Iida, and Y. Asakura. Estimation of origin-destination matrices from link traffic counts on congested networks. *Transportation Research Part B*, 26 (6):417–434, 1992. doi: 10.1016/0191-2615(92)90008-K.

[210] K. Ashok and M. E. Ben-Akiva. Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows. *Transportation Science*, 34 (1):21–36, 2000.

[211] Xuesong Zhou and Hani S. Mahmassani. A structural state space model for real-time traffic origin-destination demand estimation and prediction in a day-to-day learning framework. *Transportation Research Part B: Methodological*, 41(8):823–840, 2007. ISSN 0191-2615. doi: 10.1016/j.trb.2007.02.004.

[212] E. Castillo, A. Rivas, P. Jimenez, and J. M. Menendez. Observability in traffic networks. plate scanning added by counting information. *Transportation*, 39(6):1301–1333, 2012. doi: 10.1007/s11116-012-9390-0.

[213] Ennio Cascetta, Andrea Papola, Vittorio Marzano, Fulvio Simonelli, and Iolanda Vitiello. Quasi-dynamic estimation of od flows from traffic counts: Formulation, statistical validation and performance analysis on real data. *Transportation Research Part B: Methodological*, 55(Supplement C):171–187, 2013.

[214] G. Cantelmo, E. Cipriani, A. Gemma, and M. Nigro. An adaptive bi-level gradient procedure for the estimation of dynamic traffic demand. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1348–1361, 2014.

[215] E. Cipriani, M. Nigro, G. Fusco, and C. Colombaroni. Effectiveness of link and path information on simultaneous adjustment of dynamic o-d demand matrix. *European Transport Research Review*, 6(2):139–148, 2014.

[216] Constantinos Antoniou, Jaume Barcelo, Martijn Breen, Manuel Bullejos, Jordi Casas, Ernesto Cipriani, Biagio Ciuffo, Tamara Djukic, Serge Hoogendoorn, Vittorio Marzano, Lidia Montero, Marialisa Nigro, Josep Perarnau, Vincenzo Punzo, Tomer Toledo, and Hans van Lint. Towards a generic benchmarking platform for origin-destination flows estimation/updating algorithms: Design, demonstration and validation. *Transportation Research Part C-Emerging Technologies*, 66:79–98, 2016. ISSN 0968-090X.

[217] J. T. Lundgren and A. Peterson. A heuristic for the bilevel origin-destination-matrix estimation problem. *Transportation Research Part B: Methodological*, 42(4):339–354, 2008. doi: 10.1016/j.trb.2007.09.005.

[218] Ennio Cascetta and Maria Nadia Postorino. Fixed point approaches to the estimation of o/d matrices using traffic counts on congested networks. *Transportation Science*, 35(2):134–147–, 2001.

[219] Iwao Okutani and Yorgos J. Stephanedes. Dynamic prediction of traffic volume through Kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):1–11, 1984. ISSN 0191-2615. doi: 10.1016/0191-2615(84)90002-X.

[220] Nanne Van Der Zijpp. Dynamic origin-destination matrix estimation from traffic counts and automated vehicle identification data. *Transportation Research Record: Journal of the Transportation Research Board*, 1607(-1):87–94, 1997. doi: 10.3141/1607-13.

[221] T. Djukic, G. Flotterod, H. Van Lint, and S. Hoogendoorn. Efficient real time od matrix estimation based on principal component analysis. In *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 115–121, 2012. doi: 10.1109/ITSC.2012.6338720.

[222] S. Carrese, E. Cipriani, L. Mannini, and M. Nigro. Dynamic demand estimation and prediction for traffic urban networks adopting new data sources. *Transportation Research Part C: Emerging Technologies*, 81:83–98, 2017. ISSN 0968-090X. doi: 10.1016/j.trc.2017.05.013.

[223] M. Bierlaire and Ph L. Toint. Meuse: An origin-destination matrix estimator that exploits structure. *Transportation Research Part B: Methodological*, 29(1):47–60–, 1995. ISSN 0191-2615.

[224] J. Kim, F. Kurauchi, N. Uno, T. Hagihara, and T. Daito. Using electronic toll collection data to understand traffic demand. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 18(2):190–203, 2014. doi: 10.1080/15472450.2013.806858.

[225] Roberto Camus, Giulio E. Cantarella, and Domenico Inaudi. Real-time estimation and prediction of origin–destination matrices per time slice. *International Journal of Forecasting*, 13(1):13–19–, 1997. ISSN 0169-2070.

[226] Jaume Barcelo, Lidin Montero, Laura Marquos, and Carlos Carmona. Travel time forecasting and dynamic origin-destination estimation for freeways based on Bluetooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, 2175(-1):19–27, 2010.

[227] X. Zhou and H. S. Mahmassani. Dynamic origin-destination demand estimation using automatic vehicle identification data. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):105–114, 2006. doi: 10.1109/TITS.2006.869629.

[228] Q. Ge and D. Fukuda. Updating origin-destination matrices with aggregated data of GPS traces. *Transportation Research Part C: Emerging Technologies*, 69:291–312, 2016. ISSN 0968-090X. doi: 10.1016/j.trc.2016.06.002.

[229] Lauren Alexander, Shan Jiang, Mikel Murga, and Marta C. Gonzalez. Origin-destination trips by purpose and time of day inferred from mobile phone data. *Transportation Research Part C: Emerging Technologies*, 58, Part B:240–250, 2015.

[230] T. A. Zin, Kyaing, K. K. Lwin, and Y. Sekimoto. Estimation of originating-destination trips in yangon by using big data source. *Journal of Disaster Research*, 13(1):6–13, 2018.

[231] Jadrzej Gadzinski. Perspectives of the use of smartphones in travel behaviour studies: Findings from a literature review and a pilot study. *Transportation Research Part C: Emerging Technologies*, 88:74–86, 2018. ISSN 0968-090X.

[232] M. Nigro, E. Cipriani, and A. del Giudice. Exploiting floating car data for time-dependent origindestination matrices estimation. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 22(2):159–174, 2018.

[233] Ennio Cascetta and Gerald Marquis. Dynamic estimators of origin-destination matrices using traffic counts. *Transportation Science*, 27(4):363–373, 1993. doi: 10.1287/trsc.27.4.363.

[234] T. Djukic, J.W.C. van Lint, and S.P. Hoogendoorn. Reliability assessment of dynamic od estimation methods based on structural similarity index. In *Transportation Research Board Annual Meeting*, page 13, Washington D.C., 2013. National Academies.

[235] R. Tolouei, S. Psarras, and R. Prince. Origin-destination trip matrix development: Conventional methods versus mobile phone data. In *Transportation Research Procedia*, volume 26, pages 39–52, 2017.

[236] C. Chen, L. Bian, and J. Ma. From traces to trajectories: How well can we guess activity locations from mobile phone traces? *Transportation Research Part C: Emerging Technologies*, 46:326–337, 2014. doi: 10.1016/j.trc.2014.07.001.

[237] P. Widhalm, Y. Yang, M. Ulm, S. Athavale, and M.C. Gonzlez. Discovering urban activity patterns in cell phone data. *Transportation*, 42(4):597–623, 2015. doi: 10.1007/s11116-015-9598-x.

[238] P.S. Castro, D. Zhang, C. Chen, S. Li, and G. Pan. From taxi gps traces to social and community dynamics: A survey. *ACM Computing Surveys*, 46, 2013. doi: 10.1145/2543581.2543584.

[239] X. Yang, Y. Lu, and W. Hao. Origin-destination estimation using probe vehicle trajectory and link counts. *Journal of Advanced Transportation*, 2017, 2017.

[240] L. Cheng, S. Zhu, Z. Chu, and J. Cheng. A bayesian network model for origin-destination matrices estimation using prior and some observed link flows. *Discrete Dynamics in Nature and Society*, 2014, 2014. doi: 10.1155/2014/192470.

[241] Rodric Frederix, Francesco Viti, Ruben Corthout, and Chris M. J. Tampere. New gradient approximation method for dynamic origin-destination matrix estimation on congested networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2263:19–25, 2012.

[242] X. Di and H.X. Liu. Boundedly rational route choice behavior: A review of models and methodologies. *Transportation Research Part B: Methodological*, 85:142–179, 2016. doi: 10.1016/j.trb.2016.01.002.

[243] K. Ashok and M. E. Ben-Akiva. Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows. *Transportation Science*, 36(2):184–198, 2002. ISSN 1526-5447.

[244] C. Antoniou, M. Ben-Akiva, and H. N. Koutsopoulos. Dynamic traffic demand prediction using conventional and emerging data sources. 153(1):97–104, 2006. ISSN 1748-0248.

[245] Z. Lu, W. Rao, Y. J. Wu, L. Guo, and J. Xia. A Kalman filter approach to dynamic od flow estimation for urban road networks using multi-sensor data. *Journal of Advanced Transportation*, 49(2):210–227, 2015. doi: 10.1002/atr.1292.

[246] J. Barcelo, L. Montero, M. Bullejos, O. Serch, and C. Carmona. A kalman filter approach for exploiting bluetooth traffic data when estimating time-dependent od matrices. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 17(2):123–141, 2013.

[247] Vittorio Marzano, Andrea Papola, Fulvio Simonelli, and Markos Papageorgiou. A kalman filter for quasi-dynamic od flow estimation/updating. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–9, 2018.

[248] Clelia Lopez, Ludovic Leclercq, Panchamy Krishnakumari, Nicolas Chiabaut, and Hans van Lint. Revealing the day-to-day regularity of urban congestion patterns with 3d speed maps. *Scientific Reports*, 7(1):14029, 2017.

[249] Anna Altman and Jacek Gondzio. Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. *Optimization Methods and Software*, 11(1-4):275–302, 1999.

[250] Tamara Djukic, J. W. C. van Lint, and S. P. Hoogendoorn. Application of principal component analysis to predict dynamic origin-destination matrices. *Transportation Research Record*, (2283):81–89, 2012. doi: 10.3141/2283-09.

[251] Ian T Jolliffe. *Introduction*. Springer, 2002.

[252] Clelia Lopez, Panchamy Krishnakumari, Ludovic Leclercq, Nicolas Chiabaut, and Hans van Lint. Spatio-temporal partitioning of transportation network using travel time data. *Transportation Research Record: Journal of the Transportation Research Board*, (2623), 2017.

[253] Thomas Martinetz, Klaus Schulten, et al. A" neural-gas" network learns topologies. 1991.

[254] Ennio Cascetta. *Transportation systems engineering: theory and methods*, volume 49. Springer Science & Business Media, 2013.

[255] Thomas P Vogl, JK Mangis, AK Rigler, WT Zink, and DL Alkon. Accelerating the convergence of the back-propagation method. *Biological cybernetics*, 59(4-5): 257–263, 1988.

[256] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[257] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[258] Aimsun. *Aimsun Next 8.2 User's Manual*. Aimsun SL, Barcelona, Spain, 2017.

[259] F. Viti, M. Rinaldi, F. Corman, and C. M. J. Tampe'‘re. Assessing partial observability in network sensor location problems. *Transportation Research Part B: Methodological*, 70:65–89, 2014. doi: 10.1016/j.trb.2014.08.002.

[260] R Eddie Wilson and Jonathan A Ward. Car-following models: fifty years of linear stability analysis–a mathematical perspective. *Transportation Planning and Technology*, 34(1):3–18, 2011.

[261] Nicola Bellomo and Christian Dogbe. On the modeling of traffic and crowds: A survey of models, speculations, and perspectives. *SIAM review*, 53(3):409–463, 2011.

[262] Femke van Wageningen-Kessels, Hans Van Lint, Kees Vuik, and Serge Hoogendoorn. Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics*, 4(4):445–473, 2015.

[263] Qian Ge and Daisuke Fukuda. A macroscopic dynamic network loading model for multiple-reservoir system. *Transportation Research Part B: Methodological*, 126: 502–527, 2019.

[264] Victor L Knoop, Hans Van Lint, and Serge P Hoogendoorn. Traffic dynamics: Its impact on the macroscopic fundamental diagram. *Physica A: Statistical Mechanics and its Applications*, 438:236–250, 2015.

[265] Martin Dörnfelder, Jiong Guo, Christian Komusiewicz, and Mathias Weller. On the parameterized complexity of consensus clustering. *Theoretical Computer Science*, 542:71–82, 2014.

[266] Panchamy Krishnakumari, Tin Nguyen, Léonie Heydenrijk-Ottens, Hai L Vu, and Hans van Lint. Traffic congestion pattern classification using multiclass active shape models. *Transportation Research Record: Journal of the Transportation Research Board*, (2645):94–103, 2017.

[267] Panchamy Krishnakumari, Oded Cats, and Hans van Lint. Heuristic coarsening for generating multiscale transport networks. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[268] Opensource heuristic coarsening code. https://github.com/Panchamy/Heuristic-Coarsening/wiki.

[269] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[270] David G Kendall. A survey of the statistical theory of shape. *Statistical Science*, pages 87–99, 1989.

[271] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[272] Nwb (nationaal wegenbestand). https://www.arcgis.com/home/item.html?id=36486316f8674b3fa15c9ee2b2d8ecd7#data.

[273] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[274] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[275] Vladimir Filkov and Steven Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(04):863–880, 2004.

[276] Martin Trépanier, Catherine Morency, and Bruno Agard. Calculation of transit performance measures using smartcard data. *Journal of Public Transportation*, 12(1):5, 2009.

[277] Jinhua Zhao, Michael Frumin, Nigel Wilson, and Zhan Zhao. Unified estimator for excess journey time under heterogeneous passenger incidence behavior using smartcard data. *Transportation research part C: emerging technologies*, 34:70–88, 2013.

[278] Patricia Hendren, Justin Antos, Yvonne Carney, and Richard Harcum. Transit travel time reliability: shifting the focus from vehicles to customers. *Transportation Research Record*, 2535(1):35–44, 2015.

[279] Luis Moreira-Matias, João Mendes-Moreira, Jorge Freire de Sousa, and João Gama. Improving mass transit operations by using avl-based systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1636–1653, 2015.

[280] Haris N Koutsopoulos, Peyman Noursalehi, Yiwen Zhu, and Nigel HM Wilson. Automated data in transit: Recent developments and applications. In *2017 5th IEEE international conference on models and technologies for intelligent transportation systems (MT-ITS)*, pages 604–609. IEEE, 2017.

[281] Neema Nassir, Alireza Khani, Sang Gu Lee, Hyunsoo Noh, and Mark Hickman. Transit stop-level origin–destination estimation through use of transit schedule and automated data collection system. *Transportation research record*, 2263(1):140–150, 2011.

[282] Jason B Gordon, Harilaos N Koutsopoulos, Nigel HM Wilson, and John P Attanucci. Automated inference of linked transit journeys in london using fare-transaction and vehicle location data. *Transportation Research Record*, 2343(1):17–24, 2013.

[283] Xiaolei Ma, Yao-Jan Wu, Yinhai Wang, Feng Chen, and Jianfeng Liu. Mining smart card data for transit riders travel patterns. *Transportation Research Part C: Emerging Technologies*, 36:1–12, 2013.

[284] Ashish Bhaskar, Edward Chung, et al. Passenger segmentation using smart card data. *IEEE Transactions on intelligent transportation systems*, 16(3):1537–1548, 2014.

[285] Xiaolei Ma, Congcong Liu, Huimin Wen, Yunpeng Wang, and Yao-Jan Wu. Understanding commuting patterns using transit smart card data. *Journal of Transport Geography*, 58:135–145, 2017.

[286] Zhen-Liang Ma, Luis Ferreira, Mahmoud Mesbah, and Ahmad Tavassoli Hojati. Modeling bus travel time reliability with supply and demand data from automatic vehicle location and smart card systems. *Transportation Research Record*, 2533(1): 17–27, 2015.

[287] Zhenliang Ma, Sicong Zhu, Haris N Koutsopoulos, and Luis Ferreira. Quantile regression analysis of transit travel time reliability with automatic vehicle location and farecard data. *Transportation Research Record*, 2652(1):19–29, 2017.

[288] Menno Yap, Oded Cats, and Bart van Arem. Crowding valuation in urban tram and bus transportation based on smart card data. *Transportmetrica A: Transport Science*, pages 1–20, 2018.

[289] Martin Trépanier, Nicolas Tranchant, and Robert Chapleau. Individual trip destination estimation in a transit smart card automated fare collection system. *Journal of Intelligent Transportation Systems*, 11(1):1–14, 2007.

[290] Gabriel E Sánchez-Martínez. Inference of public transportation trip destinations by using fare transaction and vehicle location data: Dynamic programming approach. *Transportation Research Record*, 2652(1):1–7, 2017.

[291] Juanjuan Zhao, Fan Zhang, Lai Tu, Chengzhong Xu, Dayong Shen, Chen Tian, Xiang-Yang Li, and Zhengxi Li. Estimation of passenger route choice pattern using smart card data for complex metro systems. *IEEE Transactions on Intelligent Transportation Systems*, 18(4):790–801, 2016.

[292] Yiwen Zhu, Haris N Koutsopoulos, and Nigel HM Wilson. A probabilistic passenger-to-train assignment model based on automated data. *Transportation Research Part B: Methodological*, 104:522–542, 2017.

[293] P. Krishnakumari, H. van Lint, T. Djukic, and O. Cats. A data driven method for od matrix estimation. *Transportation Research Part C: Emerging Technologies*, 2019. doi: 10.1016/j.trc.2019.05.014.

Summary

Cities are complex, dynamic and ever-evolving. We need to understand how these cities work in order to predict, control or optimize its operations. There has been significant effort from the transportation field in using simulation-based and data-driven approaches to understand these complex dynamics. However, there are several limitations to simulation-based approaches that hamper the network-wide analysis of traffic patterns for large-scale metropolitan cities. Here, data-driven approaches can help. Notwithstanding, the current data-driven approaches also face several challenges. We have identified some open issues in two key areas that need to be solved to build feasible methods for this purpose - networks and network dynamics. To this end, this thesis develops a series of data-driven methods for extracting the mobility patterns of large-scale metropolitan networks and explore some of their applications.

Networks are the cornerstone of any transportation system. We present a heuristic method for automatically generating multiscale transportation networks without compromising its key topological characteristics. It addresses a problem that is becoming increasingly relevant in the age of big data, where reducing the network complexity could easily determine the viability of the research in real-world applications. To support further research, an open-source implementation of the algorithm is made available.

Despite the network complexity reduction, the dimensionality of the traffic variables to represent the traffic state can still be high, depending on the space and time aggregation of the data. Thus, we examine different methods from fields such as graph partitioning, data point clustering and computer vision to extract the essence of network dynamics from the vast amount of spatiotemporal data.

First, we construct 3D speed maps to represent the spatiotemporal relations of traffic data for an urban network and for a particular day. Partitioning techniques combined with a new post-treatment methodology has been used to reduce the dimensionality of these 3D networks to create 3D cluster maps. These cluster maps have been used to group multiple days with similar patterns and synthesis these patterns into representative consensual patterns. This paves the way for a cutting-edge systematic method for travel time predictions in cities, where we can match the current observation to historical consensual 3D patterns to make real-time predictions.

Second, we use concepts of human vision for understanding the complex mobility patterns. This is a new way of looking at traffic patterns. It combines the field of pattern recognition - with a focus on computer vision - with the traffic domain. The inspiration comes from the fact that humans are the most sophisticated pattern recognizer in the world and we use specific visual features to recognize different complex patterns. Our assumption is that

we can use the same features for recognizing traffic patterns. To this end, we use one such visual feature – shapes – to define traffic congestion patterns. With just two archetype base shapes, we are able to distinguish between different traffic patterns and achieve promising results.

Third, we further expand on this experience at using high-level features from computer vision. We encoded traffic information as images and used a deep learning model trained on natural images to extract feature vectors. These feature vectors cluster naturally into meaningful network traffic states where low, medium and high congestion are distinctly recognized. This opens up many avenues of future research as we can disentangle the dominant features from the pre-trained neural network that make the distinction between different traffic states and use this to identify high-level features other than shapes to define a network traffic state.

Leveraging the compact representation of the massive data over the entire network using 3D maps, we have proposed three applications of such mobility patterns. First, we use the 3D supply patterns for revealing an unknown correlation with demand patterns. Furthermore, we use these predicted demand patterns and the 3D supply patterns for a new data-driven OD estimation framework. This framework uses only two behavioral assumptions and do not need an equilibrium assignment or network loading model, unlike traditional OD estimation methods. Moreover, the framework incorporates additional constraints, which can be derived directly from data, to be scalable for large networks.

Second, we extend the shape-based approach to create compact and scalable custom feature vector to reveal regularity between daily network patterns. We compare the performance of the method against partitioning-based approach for network-wide travel time predictions. We also evaluate the scalability of the approach by applying it to the entire Dutch highway network. This type of feature vector formulation allows for efficiently incorporating additional information such as context, weather and incident data.

Finally, we also pave the way to introduce these methods for other modes of transport by proposing a new estimation method to represent the spatio-temporal network dynamics of a public transport network using 3D maps. The method decomposes the individual passenger trajectories into its corresponding track segment delay, initial waiting time and transfer delay. This way of compressing individual passenger trajectories into a 3D map has numerous applications such as delay prediction, disruption detection and asset management.

With the increasing availability of data in the transport domain, the Achilles heel is not data scarcity anymore but rather extracting insights from this massive amount of data. This thesis is a step forward in solving this complex problem by leveraging the increased acceptance of using machine learning as a worthy and effective method for network-wide analysis of traffic patterns.

Samenvatting

Steden zijn complex, dynamisch en altijd in ontwikkeling. We moeten begrijpen hoe deze steden werken om hun activiteiten te voorspellen, te beheersen of te optimaliseren. Vanuit het transportveld zijn aanzienlijke inspanningen geleverd om op simulatie gebaseerde en datagestuurde benaderingen te gebruiken om deze complexe dynamiek te begrijpen. Er zijn echter meerdere limitaties aan deze op simulaite gebaseerde banderingen die de netwerkbrede analyse van verkeerspatronen voor grootstedelijke steden belemmeren. Hier kunnen datagestuurde benaderingen helpen. Ondanks dat de huidige datagestuurde benaderingen ook voor verschillende uitdagingen staan. We hebben enkele open problemen gedentificeerd op twee belangrijke gebieden die moeten worden opgelost om voor dit doel bruikbare methoden te ontwikkelen: netwerken en netwerkdynamiek. Hiertoe ontwikkelt dit proefschrift een reeks datagestuurde methoden om de mobiliteitspatronen van grootschalige grootstedelijke netwerken te extraheren en enkele van hun toepassingen te verkennen.

Netwerken vormen de hoeksteen van elk transportsysteem. We presenteren een heuristische methode voor het automatisch genereren van multischaal transportnetwerken zonder afbreuk te doen aan de belangrijkste topologische kenmerken. Het lost een probleem op dat steeds relevanter wordt in het tijdperk van big data, waarbij het verminderen van de netwerkcomplexiteit gemakkelijk de levensvatbaarheid van het onderzoek in real-world applicaties kan bepalen. Ter ondersteuning van verder onderzoek is een open-source implementatie van het algoritme beschikbaar gesteld.

Ondanks de vermindering van de netwerkcomplexiteit, kan de dimensionaliteit van de verkeersvariabelen om de verkeersstatus weer te geven, nog steeds hoog zijn, afhankelijk van de ruimte- en tijdaggregatie van de gegevens. Daarom onderzoeken we verschillende methoden uit velden zoals grafiekpartitionering, datapuntclustering en computer visie om de essentie van netwerkdynamiek te extraheren uit de enorme hoeveelheid tijdruimtelijke gegevens.

Eerst maken we 3D-snelheidskaarten om de tijdruimtelijke relaties van verkeersgegevens voor een stedelijk netwerk en voor een bepaalde dag weer te geven. Partitioneringstechnieken gecombineerd met een nieuwe methode voor nabehandeling is gebruikt om de dimensionaliteit van deze 3D-netwerken te verminderen om 3D-clusterkaarten te maken. Deze clusterkaarten zijn gebruikt om meerdere dagen met vergelijkbare patronen te groeperen en deze patronen samen te stellen in representatieve consensuele patronen. Dit maakt de weg vrij voor een geavanceerde systematische methode voor reistijdvoorspellingen in steden, waar we de huidige observatie kunnen koppelen aan historische consensuele 3D-patronen om realtime voorspellingen te doen.

Ten tweede gebruiken we concepten van menselijke visie om de complexe mobiliteitspatronen te begrijpen. Dit is een nieuwe manier om naar verkeerspatronen te kijken. Het combineert het veld van patroonherkenning - met een focus op computer visie - met het verkeersdomein. De inspiratie komt van het feit dat mensen de meest geavanceerde patroonherkenners ter wereld zijn en we gebruiken specifieke visuele kenmerken om verschillende complexe patronen te herkennen. Onze veronderstelling is dat we dezelfde kenmerken kunnen gebruiken voor het herkennen van verkeerspatronen. Hiertoe gebruiken we een dergelijke visuele kenmerken - vormen - om verkeersopstoppingspatronen te definiren. Met slechts twee basisvormen kunnen we onderscheid maken tussen verschillende verkeerspatronen en veelbelovende resultaten bereiken.

Ten derde breiden we deze ervaring verder uit bij het gebruik van kenmerken op hoog niveau van computer visie. We codeerden verkeer variabelen als afbeeldingen en gebruikten een diepgaand leermodel dat was getraind op natuurlijke afbeeldingen om kenmerkvectoren te extraheren. Deze kenmerkvectoren clusteren op een gemakkelijke wijze in zinnige netwerkverkeerstoestanden waar lage, gemiddelde en hoge verkeersdichtheid duidelijk worden herkend. Dit opent vele mogelijkheden voor toekomstig onderzoek, omdat we de dominante kenmerken van het vooraf opgeleide neurale netwerk kunnen onderscheiden die het onderscheid maken tussen verschillende verkeersstatussen en dit gebruiken om andere kenmerken op hoog niveau dan vormen te identificeren om een netwerkverkeerstoestand te definiren.

Gebruikmakend van de compacte weergave van de enorme gegevens over het gehele netwerk met behulp van 3D-kaarten, hebben we drie toepassingen van dergelijke mobiliteitspatronen voorgesteld. Eerst gebruiken we de 3D-aanbodpatronen om een onbekende correlatie vraagpatronen te onthullen. Verder gebruiken we deze voorspelde vraagpatronen en de 3D-aanbodpatronen voor een nieuw datagestuurd OD-schattingskader. Dit kader gebruikt slechts twee gedragsaannames en heeft geen evenwichtstoewijzing of netwerkbelastingsmodel nodig, in tegenstelling tot traditionele OD-schattingsmethoden. Bovendien bevat het kader extra beperkingen, die rechtstreeks uit gegevens kunnen worden afgeleid, om schaalbaar te zijn voor grote netwerken.

Ten tweede, breiden we de methode gebaseerd op vormen uit om een compacte en schaalbare kenmerkvector te maken waarmee trends waargenomen kunnen worden tussen de dagelijkse patronen van het netwerk. We vergelijken de prestatie van de methode met methoden gebaseerd op verdelingen voor de reistijd voorspellingen over het gehele netwerk. We evalueren ook de schaalbaarheid van de methode door deze toe te passen op het gehele Nederland wegen netwerk. Dit type kenmerkvector formulering maakt het efficint opnemen van aanvullende informatie mogelijk, zoals context-, weers- en incidentgegevens.

Ten slotte maken we de weg vrij om deze methoden voor andere vervoerswijzen te introduceren door een nieuwe schattingsmethode voor te stellen om de tijdruimtelijk netwerk dynamiek van een openbaar vervoersnetwerk met behulp van 3D-kaarten weer te geven. De methode ontleedt de individuele passagierstrajecten in de overeenkomstige vertraging van het spoorsegment, initile wachttijd en overdrachtsvertraging. Deze manier om individuele passagierstrajecten te comprimeren tot een 3D-kaart heeft tal van toepassingen zoals vertragingsvoorspelling, detectie van verstoringen en activabeheer.

Met de toenemende beschikbaarheid van data in het transportdomein is de achilleshiel geen data schaarste meer, maar het extraheren van inzichten uit deze enorme hoeveelheid gegevens. Dit proefschrift is een stap voorwaarts bij het oplossen van dit complexe probleem door gebruik te maken van de toegenomen acceptatie van het gebruik van machine learning als een waardige en effectieve methode voor netwerkbrede analyse van verkeerspatronen.

About the author

Panchamy Krishnakumari was born on August 26, 1991 in Kollam, Kerala, India, a place known for its breathtaking backwaters and scenery. She conducted her bachelor's study in the same city from 2009 to 2013, majoring in Electronics and Communication Engineering at Amrita School of Engineering. In January 2013, she was selected for an exchange program to spend the final semester at KTH Royal Institute of Technology, Sweden. She completed her bachelor's study in August 2013 with the thesis titled *'Visual-based inflight file transfer'*.

In September 2013, Panchamy started her double degree master's program in Information and Communication Technology with a scholarship from EIT Digital. The first year of study was at KTH Royal Institute of Technology, Sweden and second year at TU Delft, the Netherlands with a major in medical imaging and a minor in Entrepreneurship. During her masters, she worked as a student assistant at the transportation department of both universities, where she pursued her passion for working with data and building interesting visualization. She completed her master's study in August 2015 with the thesis titled *'Supervised Learning for Measuring Hip Joint Distance in Digital X-ray Images'*, which she conducted at a startup in Delft, Clinical Graphics BV.

From September 2015, she worked as a Software Consultant at CGI Nederlands BV with a focus on building traffic monitoring tools, while also working one day a week at Dittlab as a guest researcher. In February 2016, Panchamy started her Ph.D. in the Department of Transport and Planning at TU Delft, funded by the European Union Horizon 2020 project. During the Ph.D., she used her background in data analysis, network science, and computer vision to come up with new ways to analyze traffic data of large-scale metropolitan cities.

Apart from research, Panchamy is passionate about building tangible products -from simple visualizations to full-fledged web portals- based on her research as she believes that even the greatest ideas can fail if it is not delivered in an understandable and user-friendly form. She also enjoys working with students to share her knowledge through thesis and course supervision. After the Ph.D., she wants to continue to share her passion for data with the transport community and further explore how she can contribute to the transportation field.

Journal Articles

1. **P. Krishnakumari**, T. Nguyen, L. Heydenrijk-Ottens, H.L. Vu, and H. van Lint (2017) Traffic congestion pattern classification using multiclass active shape models. *Transportation Research Record: Journal of the Transportation Research Board*,(2645):94-103. https://doi.org/10.3141/2645-11.

2. C. Lopez, **P. Krishnakumari**, L. Leclercq, N. Chiabaut, and H. van Lint (2017) Spatiotemporal Partitioning of Transportation Network Using Travel Time Data. *Transportation Research Record: Journal of the Transportation Research Board*, (2623):98-107. https://doi.org/10.3141/2623-11.

3. C. Lopez, L. Leclercq, **P. Krishnakumari**, N. Chiabaut, H. van Lint (2017) Revealing the day-to-day regularity of urban congestion patterns with 3D speed maps. *Scientific Reports*, 7(1):14029. https://doi.org/10.1038/s41598-017-14237-8.

4. T. Nguyen, **P. Krishnakumari**, S.C. Calvert, H.L. Vu, and H. van Lint (2019) Feature extraction and clustering analysis of highway congestion. *Transportation Research Part C: Emerging Technologies*, 100:238-258. https://doi.org/10.1016/j.trc.2019.01.017.

5. **P. Krishnakumari**, O. Cats, and H. van Lint (2019) Heuristic Coarsening for Generating Multiscale Transport Networks. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2019.2912430.

6. **P. Krishnakumari**, H. van Lint, T. Djukic, and O. Cats (2019) A data driven method for OD matrix estimation. *Transportation Research Part C: Emerging Technologies*. https://doi.org/10.1016/j.trc.2019.05.014.

Peer-reviewed Conference Contributions

1. **P. Krishnakumari** and H. van Lint (2016) Topological Coarsening Schemes for Multiscale Graph Generation. *5th hEART symposium*, Delft, The Netherlands

2. H.N. Nguyen, **P. Krishnakumari**, H.L. Vu, and H. van Lint (2016) Traffic congestion pattern classification using multi-class SVM. In *Proceeding of Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on* (pp. 1059-1064), Rio, Brazil. https://doi.org/10.1109/ITSC.2016.7795687.

3. **P. Krishnakumari**, O.Cats, and H. van Lint (2017) Network Traffic Pattern Classification using Multi-class 3D Active Shape Models. *2017 the Second International Workshop on Pattern Recognition*, Singapore.

4. **P. Krishnakumari**, A. Perotti, V. Pinto, O.Cats, and H. van Lint (2018) Understanding Network Traffic States using Transfer Learning. In *Proceeding of Intelligent Transportation Systems (ITSC), 2018 IEEE 21st International Conference on*, Maui, USA. https://doi.org/10.1109/ITSC.2018.8569450.

5. O. Cats, **P. Krishnakumari**, and K. Tundulyasaree (2019) Metropolitan network robustness: Investigating the role of rapid network development and a polycentric structure. *98th Annual Meeting of the Transportation Research Board*, Washington D.C., USA.

6. T. Liu, **P. Krishnakumari**, and O.Cats (2019) Exploring Demand Patterns of a Ride-Sourcing Service using Spatial and Temporal Clustering. *6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Krakow, Poland. https://doi.org/10.1109/MTITS.2019.8883312

7. **P. Krishnakumari**, H. van Lint, T. Djukic, and O. Cats (2019) A data driven method for OD matrix estimation. In *Proceedings of the 23rd International Symposium on Transportation and Traffic Theory (ISTTT23)*, Lausanne, Switzerland. https://doi.org/10.1016/j.trpro.2019.05.009

8. **P. Krishnakumari**, O.Cats, and H. van Lint (2020) Day-to-day and Seasonal Regularity of Network Passenger Delay for Metro Networks. *99th Annual Meeting of the Transportation Research Board*, Washington D.C., USA.

9. O. Cats, **P. Krishnakumari**, N. Arbez, N. Chiabaut, and H. van Lint (2020) Empirical Evaluation of the Efficiency and Effectiveness of a Ridesourcing Service. *99th Annual Meeting of the Transportation Research Board*, Washington D.C., USA.

10. H. van Lint, T. Nguyen, **P. Krishnakumari**, S.C. Calvert, H. Schuurman, and M. Schreuder (2020) Can We Estimate the Safety Effects of Congestion Warning Systems from Just Carriageway Aggregate Data? *99th Annual Meeting of the Transportation Research Board*, Washington D.C., USA.

TRAIL Thesis Series

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 250 titles see the TRAIL website: www.rsTRAIL.nl.

The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Krishnakumari, P.K., *Multiscale Pattern Recognition of Transport Network Dynamics and its Applications: A birds eye view on transport*, T2020/5, February 2020, TRAIL Thesis Series, the Netherlands

Wolbertus, *Evaluating Electric Vehicle Charging Infrastructure Policies*, T2020/4, February 2020, TRAIL Thesis Series, the Netherlands

Yap, M.D., *Measuring, Predicting and Controlling Disruptions for Urban Public Transport*, T2020/3, February 2020, TRAIL Thesis Series, the Netherlands

Luo, D., *Data-driven Analysis and Modeling of Passenger Flows and Service Networks for Public Transport Systems*, T2020/2, February 2020, TRAIL Thesis Series, the Netherlands

Erp, P.B.C. van, *Relative Flow Data: New opportunities for traffic state estimation*, T2020/1, February 2020, TRAIL Thesis Series, the Netherlands

Zhu, Y., *Passenger-Oriented Timetable Rescheduling in Railway Disruption Management*, T2019/16, December 2019, TRAIL Thesis Series, the Netherlands

Chen, L., *Cooperative Multi-Vessel Systems for Waterborne Transport*, T2019/15, November 2019, TRAIL Thesis Series, the Netherlands

Kerkman, K.E., *Spatial Dependence in Travel Demand Models: Causes, implications, and solutions*, T2019/14, October 2019, TRAIL Thesis Series, the Netherlands

Liang, X., *Planning and Operation of Automated Taxi Systems*, T2019/13, September 2019, TRAIL Thesis Series, the Netherlands

Ton, D., *Unravelling Mode and Route Choice Behaviour of Active Mode Users*, T2019/12, September 2019, TRAIL Thesis Series, the Netherlands

Shu, Y., *Vessel Route Choice Model and Operational Model Based on Optimal Control*, T2019/11, September 2019, TRAIL Thesis Series, the Netherlands

Luan, X., *Traffic Management Optimization of Railway Networks*, T2019/10, July 2019, TRAIL Thesis Series, the Netherlands

Hu, Q., *Container Transport inside the Port Area and to the Hinterland*, T2019/9, July 2019, TRAIL Thesis Series, the Netherlands

Andani, I.G.A., *Toll Roads in Indonesia: transport system, accessibility, spatial and equity impacts*, T2019/8, June 2019, TRAIL Thesis Series, the Netherlands

Ma, W., *Sustainability of Deep Sea Mining Transport Plans*, T2019/7, June 2019, TRAIL Thesis Series, the Netherlands

Alemi, A., *Railway Wheel Defect Identification*, T2019/6, January 2019, TRAIL Thesis Series, the Netherlands

Liao, F., *Consumers, Business Models and Electric Vehicles*, T2019/5, May 2019, TRAIL Thesis Series, the Netherlands

Tamminga, G., *A Novel Design of the Transport Infrastructure for Traffic Simulation Models*, T2019/4, March 2019, TRAIL Thesis Series, the Netherlands

Lin, X., *Controlled Perishable Goods Logistics: Real-time coordination for fresher products*, T2019/3, January 2019, TRAIL Thesis Series, the Netherlands

Dafnomilis, I., *Green Bulk Terminals: A strategic level approach to solid biomass terminal design*, T2019/2, January 2019, TRAIL Thesis Series, the Netherlands

Feng, Fan, *Information Integration and Intelligent Control of Port Logistics System*, T2019/1, January 2019, TRAIL Thesis Series, the Netherlands

Beinum, A.S. van, *Turbulence in Traffic at Motorway Ramps and its Impact on Traffic Operations and Safety*, T2018/12, December 2018, TRAIL Thesis Series, the Netherlands

Bellsol Olba, X., *Assessment of Capacity and Risk: A Framework for Vessel Traffic in Ports*, T2018/11, December 2018, TRAIL Thesis Series, the Netherlands

Knapper, A.S., *The Effects of using Mobile Phones and Navigation Systems during Driving*, T2018/10, December 2018, TRAIL Thesis Series, the Netherlands

Varotto, S.F., *Driver Behaviour during Control Transitions between Adaptive Cruise Control and Manual Driving: empirics and models*, T2018/9, December 2018, TRAIL Thesis Series, the Netherlands

Stelling-Koczak, A., *Cycling Safe and Sound*, T2018/8, November 2018, TRAIL Thesis Series, the Netherlands

Essen, van M.A., *The Potential of Social Routing Advice*, T2018/7, October 2018, TRAIL Thesis Series, the Netherlands

Su, Zhou, *Maintenance Optimization for Railway Infrastructure Networks*, T2018/6, September 2018, TRAIL Thesis Series, the Netherlands

Cai, J., *Residual Ultimate Strength of Seamless Metallic Pipelines with Structural Damage*, T2018/5, September 2018, TRAIL Thesis Series, the Netherlands