Netwerkmodellering voor het ontwerp van algoritmes en toepassingen in intelligente transportsystemen

Network Modelling for Algorithm Design and Applications in Intelligent Transportation Systems

Maarten Houbraken

Promotoren: prof. dr. ir. M. Pickavet, dr. P. Audenaert
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017 - 2018

ISBN 978-94-6355-128-1
NUR 975, 976
Wettelijk depot: D/2018/10.500/46

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

Promotoren:	prof. dr. ir. Mario Pickavet
	dr. Pieter Audenaert
Juryleden:	prof. dr. ir. Didier Colle, Universiteit Gent (secretaris)
	em. prof. dr. ir. Daniël De Zutter, Universiteit Gent (voorzitter)
	prof. dr. Veerle Fack, Universiteit Gent
	dr. ir. Steven Logghe, Be-Mobile
	prof. dr. ir. Chris Tampère, Katholieke Universiteit Leuven

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur

Vakgroep Informatietechnologie
Technologiepark-Zwijnaarde 15, 9052 Gent, België

Tel.: +32-9-331.49.00
Fax.: +32-9-331.48.99

Dit werk kwam tot stand in het kader van een mandaat als Aspirant bij het Fonds voor Wetenschappelijk Onderzoek Vlaanderen

Proefschrift tot het behalen van de graad van Doctor in de ingenieurswetenschappen: computerwetenschappen
Academiejaar 2017-2018

"If I have seen further than others, it is by standing upon the shoulders of giants."

— Isaac Newton (1643 - 1727)

Dankwoord

Eén van de meest voorkomende vragen die ik de voorbije jaren kreeg was: "Zo doctoreren, wat is dat nu eigenlijk?". Een simpele vraag, maar zelfs voor zij die het doen (of gedaan hebben) is er geen eenvoudig antwoord. Op zich is doctoreren het hele proces tot het behalen van een doctorstitel, maar wat dat proces precies is, verschilt bij iedereen. Daar waar iedereen start met het uitvoeren van verdiepend onderzoek rond een specifiek onderwerp, zijn er doorheen het doctoraat veel verschillende en uiteenlopende vertakkingen. Je kan een doctoraat dan misschien het best nog vergelijken met een film. De film start met een hoofdpersonage (of onderwerp) dat je eerst leert kennen, men gaat op (onder-)zoek om een probleem op te lossen en eindigt dan in een grote finale (doctoraatsverdediging). Gaandeweg zijn er spanningsbogen, sleutelmomenten en acteurswijzigingen die bepalen of het een avonturenfilm, feel-good movie of horrorprent wordt. Gelukkig voor mij is het een science-non-fiction-film geworden met een happy ending, waarvan jullie nu de credits zitten te lezen.

Allereerst wil ik mijn promotors en de faculteit ingenieurswetenschappen danken om mij aan dit onderzoek te laten starten. Daarnaast wil ik het BOF en het FWO bedanken om in mij voldoende potentiëel te zien waardoor ik zonder financiële zorgen mij kon focussen op de wetenschap.

Wetenschappers worden soms bekeken als eenzaten in hun ivoren toren. Gelukkig was ik niet eenzaam in de Zuiderpoort en de iGent-toren. Bedankt Abhishek, Dimitri, Frederic, Ludwig, Sachin, Sahel, Sander, Sofie, Sofie, Ward, Willem en Wouter voor de leuke sfeer in onze bureaus. Bram, aan jou bedankt voor de squash- en voetbalwedstrijdjes om onze conditie op peil te houden. Thijs, bedankt om samen met mij in hetzelfde onderzoeksschuitje te zitten. Je technische skills waren altijd een plezier om op terug te vallen, zowel op pc als in de keuken. Een laatste eervolle vermelding gaat nog naar Marlies. Bedankt voor al je organisatie, voor je verbredende blik, voor je enthousiasme en voor je vriendschap.

Naast de faculteit wil ik ook Be-Mobile bedanken om het decor te zijn op het eind van de film. De inhoudelijke mogelijkheden waren een cruciale bijdrage aan dit onderzoek. In het bijzonder wil ik Steven Logghe en Wim Vanbelle bedanken. Steven, bedankt om je inhoudelijke kennis ter beschikking te stellen, om je organisatietalenten te gebruiken om interessante projecten te regelen en om met je toekomstvisie een licht in de duisternis te schijnen. Wim, bedankt voor het ver-

trouwen al die jaren, voor het in goede banen leiden van de echte rocket science en voor het pragmatische tegengewicht bij de overambitieuze ideeën.

Binnen Be-Mobile wil ik verder ook al mijn collega's bedanken voor de gezellige werksfeer. In het bijzonder wil ik Jens bedanken om altijd voor mijn data te willen zorgen, Karolien om met mij te brainstormen, Wim om met mij de krijtlijnen van mijn onderzoek vast te leggen en Bart om met zijn kritische blik mijn papers van nuttige feedback te voorzien.

Tijdens mijn onderzoek heb ik ook mogen genieten van de steun van de Nederlandse wegbeheerder Rijkswaterstaat tijdens onze samenwerkingsprojecten. Aan Marco Schreuder en Wim Breedveld een hartelijke dankjewel!

Tijdens het doctoraat waren er hier en daar wel wat stresserende momenten. Gelukkig had ik een schare trouwe supporters die ten gepaste tijde voor afleiding en ontspanning konden zorgen. Bedankt Els, Glenn, Hanne, Jeroen, Karel-Jan, Marc, Sofie en Toon. Gertjan, bedankt voor alle voetbalmatches en quizzen. Tim, merci om altijd wel te vinden te zijn voor een babbel, film en feestje!

Ook mijn Gentse supportersafdeling wil ik hier even vermelden. Aan Brecht, Daan, Pieter en Tim, merci voor alle ingenieuze avondjes. Voor Véronique hoort hier een speciale vermelding. Al die keren dat je last-minute kon invallen, dat je random pannenkoeken maakte of dat je wou nalezen, maken van jou toch wel één van mijn meest enthousiaste supporters! Bedankt om zo je hartelijke zelve te zijn!

Verder zijn er nog twee formidabele personen waar ik altijd kon op rekenen die hier thuishoren. Cédric, bedankt voor al de runs en de ontspannen babbels. Ittai, bedankt om me te inspireren met je doorzettingsvermogen en flexibiliteit.

Met trots vervoeg ik nu mede-doctors Annelies, David, Dorien, Joeri, Jonathan en Karel. Na al die jaren dezelfde beslommeringen te hebben gehad, kan ik nu ook beginnen meepraten over het leven na het doctoraat. Karel, bedankt om mijn papers te voorzien van een externe maar constructieve blik. Joeri, bedankt voor de leuke momenten samen maar vooral bedankt om er te zijn op de zwaardere. Ook ik mis de pauzes van weleer maar dan niet echt die koffieautomaat!

Doorheen mijn doctoraat (maar eigenlijk al mijn hele leven) is het toch mijn familie die me het meest gesteund heeft. Oma, opa, bedankt voor jullie levenswijsheid maar vooral ook om zo trots te zijn op jullie kleinzoon zonder echt precies te weten wat doctoreren is! Matthias, bedankt om me te tonen hoe vastberadenheid en nieuwe uitdagingen je leven kunnen verrijken. Michael, binnen 2 weken zijn de rollen omgedraaid he! Merci om al die jaren als grote broer het voorbeeld te geven, die nieuwe aanwinst kan zich geen beter rolmodel wensen!

Mama, papa, bedankt om me al die jaren te steunen en altijd klaar te staan om te helpen. Weet dat ik jullie meer waardeer dan dat ik soms laat blijken. Zonder jullie inzet en steun zou dit boek er al helemaal niet geweest zijn.

De laatste lijnen in dit stukje zijn voor iemand heel speciaal. Iemand om bij tot rust te komen, die me steunt in de lastige dagen en me terugbrengt naar de schoonheid in het leven. Stephanie, bedankt om het zonnestraaltje te zijn in de regenachtige dagen! Ik zie je graag.

Table of Contents

Dankwoord **i**

Samenvatting **xv**

Summary **xix**

1 Introduction **1**
1.1 Context . . . 1
1.2 Network modelling . . . 2
1.3 Data collection . . . 3
1.3.1 Inductive loops . . . 3
1.3.1.1 Measurements . . . 3
1.3.1.2 Processing . . . 4
1.3.1.3 Data properties . . . 5
1.3.1.4 System deployment . . . 5
1.3.2 Non-GPS-based vehicle tracking . . . 6
1.3.2.1 Bluetooth . . . 6
1.3.2.2 Licence plate recognition . . . 7
1.3.2.3 Mobile phones . . . 7
1.3.3 Floating Car Data . . . 8
1.3.3.1 Measurements . . . 8
1.3.3.2 Processing . . . 9
1.3.3.3 Data properties . . . 9
1.3.3.4 System deployment . . . 11
1.3.4 Inductive loops versus FCD . . . 11
1.4 Application algorithms . . . 13
1.5 Research challenges . . . 13
1.6 Contributions and outline . . . 14
1.7 Publications . . . 17
1.7.1 Publications in international journals (listed in the Science Citation Index) . . . 17
1.7.2 Publications in international conferences . . . 18
1.7.3 Publications in national conferences . . . 19
1.7.4 Other publications . . . 19
References . . . 20

2 An Evaluation of Section Control based on Floating Car Data 21
2.1 Introduction . . . 22
2.2 FCD capturing . . . 24
2.3 Analysis . . . 25
2.3.1 Data . . . 25
2.3.2 Speed distribution . . . 31
2.3.3 Speed dispersion . . . 33
2.3.4 Variation throughout the day . . . 35
2.4 Conclusions . . . 38
2.5 Future work . . . 38
References . . . 40

3 Modelling Traffic Congestion Evolution through Speed Profile Generation using Floating Car Data 43
3.1 Introduction . . . 44
3.2 FCD generation . . . 46
3.2.1 Converting vehicle positions to experienced travel times . 46
3.2.2 Deriving link travel times . . . 47
3.3 Modelling speed evolution throughout the day . . . 48
3.3.1 Bin model . . . 48
3.3.2 Fourier series . . . 49
3.3.3 Wavelet model . . . 50
3.3.4 Trapezoidal Rush Hour Congestion model . . . 51
3.3.4.1 Model presentation . . . 51
3.3.4.2 Fitting the model . . . 52
3.4 Application: Dutch road network . . . 54
3.4.1 FCD generation . . . 54
3.4.2 Individual profile fitting . . . 56
3.4.3 Full dataset analysis . . . 60
3.4.4 Congestion analysis . . . 62
3.5 Summary . . . 68
References . . . 68

4 Examining the Potential of Floating Car Data for Dynamic Traffic Management 71
4.1 Introduction . . . 72
4.2 Existing system & infrastructure . . . 74
4.3 FCD System . . . 77
4.3.1 FCD data . . . 78
4.3.2 FCD VSL . . . 79
4.4 Evaluation methodology . . . 80
4.4.1 State modelling . . . 81
4.4.2 State-based comparison . . . 82
4.4.3 FCD VSL calibration . . . 84
4.5 Evaluation . . . 85

4.5.1 Real-time potential analysis 86
4.6 Discussion 89
4.7 Conclusion 89
References 90

5 Automated Incident Detection using Real-Time Floating Car Data 95
5.1 Introduction 96
5.2 Existing monitoring & management 98
5.3 FCD extensions 99
5.4 Evaluation methodology 101
5.5 Results 104
5.5.1 Experimental setup 104
5.5.2 FCD AID configuration results 106
5.5.3 Full state comparison 113
5.6 Discussion 114
References 118

6 Real-time Traffic Monitoring by fusing Floating Car Data with Stationary Detector Data 121
6.1 Introduction 122
6.2 Basic data fusion algorithm 123
6.3 Data source normalisation 125
6.4 Experimental application: A58 use case 126
6.5 Conclusion and future work 129
References 129

7 Conclusions & Future Challenges 133
7.1 Conclusions 133
7.1.1 Network analysis 134
7.1.2 Dynamic traffic management 135
7.2 Future challenges 136
7.2.1 Technical challenges 136
7.2.2 Methodological challenges 137
7.2.3 Valorisation challenges 138

A Fault Tolerant Network Design inspired by *Physarum polycephalum* 141
A.1 Introduction 142
A.2 Mathematical model 145
A.3 Extensions 147
A.3.1 Migrating nodes 148
A.3.2 Stimulation of alternative paths 149
A.4 Simulations 151
A.4.1 Methods and parameters 152
A.4.2 Validation 153
A.4.3 Effects of extensions on thresholding 154

A.4.4 Comparison to original model 155
A.4.5 Comparison to living slime mould 159
A.4.6 Application to telecommunication networks 161
A.5 Conclusion 163
References 166

B The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS): Exploiting Symmetry for Faster Subgraph Enumeration 169
B.1 Introduction 170
B.2 Methods 173
B.2.1 ISMA 173
B.2.2 Symmetry in ISMA 174
B.2.3 Symmetry in ISMAGS 176
B.2.3.1 Symmetry detection 176
B.2.3.2 Symmetry breaking 181
B.2.4 Integrating symmetry detection and symmetry breaking with ISMA 183
B.3 Results 186
B.3.1 Algorithms 187
B.3.2 Network data 187
B.3.3 ISMA versus ISMAGS 190
B.3.4 Full comparison 194
B.4 Conclusion 199
References 199

List of Figures

1.1 Overview data flow. 2
1.2 Magnetic loop induction sensor. 4
1.3 Vehicle re-identification along the route. 7
1.4 Schematic grouping of the chapters & challenges in this dissertation. 15

2.1 Travel time allocation. 26
2.2 Highway layout. 26
2.3 Speed histogram. 27
2.4 Distribution of measurements over road segments for the E40 dataset. 29
2.5 Distribution of measurements over road segments for the E17 dataset. 30
2.6 Speed profile for the E40 dataset. 31
2.7 Speed profile for the E17 highway. 34
2.8 Speed dispersion. 36
2.9 Speed variance E40. 37
2.10 Top speed quantile. 39

3.1 Bin model example. 49
3.2 The general speed profile of the TRHC model. 51
3.3 Illustration of step 4 through 6 with the individual operations on the flanks of the peak. 55
3.4 Number of car and truck samples for March 24, 2015. 56
3.5 Illustration of the model fits to the source FCD. 58
3.5 Illustration of the model fits to the source FCD (continued). 59
3.6 Training and validation MSE results for Tuesday. 61
3.7 Delay index for the Netherlands. 66
3.8 Delay index for Utrecht. 67

4.1 Overview of A27 study area. 75
4.2 Spatio-temporal plot of the loop VSL on the A27 highway. 77
4.3 Individual FCD vehicle trajectories on the A27 highway. 78
4.4 FCD vehicle trajectories and loop variable speed limit signs. . . . 79
4.5 FCD VSL result on the A27 highway. 80
4.6 Category classification on the A27 highway. 82
4.7 Latency simulation of FCD. 88

5.1 AID switching. 100
5.2 State classification. 103
5.3 A58 test route. 104
5.4 A27 test route. 105
5.5 Trade-off between false positives and false negatives on the A58. . 107
5.6 Trade-off between false positives and false negatives on the A27. . 112

6.1 Schematic representation of the traffic wave in the EGTF. 124
6.2 A58 highway. 127
6.3 SDD on the A58. 128
6.4 Aggregated FCD. 128
6.5 Fused traffic state. 128

A.1 *Physarum polycephalum* on agar surface. 143
A.2 Typical simulation. 145
A.3 Migration mechanism applied on a single node. 149
A.4 Steps in stimulation process. 151
A.5 Simulation of maze-solving capabilities. 153
A.6 Relationship between edge survival and the applied conductivity threshold. 154
A.7 Results for Belgian network. 156
A.8 Analysis of simulation results. 157
A.8 Analysis of simulation results (continued). 158
A.9 Results for the Iberian peninsula. 159
A.10 Telecommunication networks. 162

B.1 Subgraph examples. 175
B.2 Some permuted instances of the Petersen graph. 176
B.3 Example of subgraph refinement. 178
B.4 Subgraph symmetry analysis of “XX00XX”. 179

List of Tables

1.1 Current OSM road network sizes. 3
1.2 Comparison between inductive loops and FCD. 12

2.1 Points of interest on the E40 highway. 29
2.2 Points of interest on the E17 highway. 30

3.1 Overall RMSE results for the training datasets on each day of the week. 63
3.2 Overall RMSE results for the validation datasets on each day of the week. 64

4.1 Points of interest in A27 study area. 76
4.2 Categorisation of VSL system combinations. 83
4.3 Optimal FCD parameters for conservative loop AID approximation. 85
4.4 Comparison FCD VSL to loop VSL. 87

5.1 Experimental FCD AID configurations deployed on the test routes. 106
5.2 AID FCD scores per configuration on the A58. 108
5.2 AID FCD scores per configuration on the A58 (continued). 109
5.3 AID FCD scores per configuration on the A27. 110
5.3 AID FCD scores per configuration on the A27 (continued). 111
5.4 Full state comparison for the A58 in configuration c_4. 115
5.5 Full state comparison for the A27 in configuration c_4. 116

A.1 Results for Iberian peninsula. 164
A.2 Results for the European telecommunication network. 165

B.1 Network properties. 188
B.2 Comparison between ISMA and ISMAGS on the biological networks. 191
B.2 Comparison between ISMA and ISMAGS on the biological networks (continued). 192
B.3 Comparison between ISMA and ISMAGS on the Slashdot and SNAP networks. 195
B.3 Comparison between ISMA and ISMAGS on the Slashdot and SNAP networks (continued). 196

B.4 Comparison between ISMAGS, VF2 and GK on the biological networks. 197
B.4 Comparison between ISMAGS, VF2 and GK on the SNAP networks. 198

List of Acronyms

A

AID Automated Incident Detection

B

BIVV Belgian Institute for Traffic Safety/Belgisch instituut voor verkeersveiligheid

C

CS Correspondence Score

D

DTM Dynamic Traffic Management

E

EGTF Extended and Generalised Treiber-Helbing Filter

F

FCD Floating Car Data

FN False Negatives

FP False Positives

FS Food Source

G

GDPR General Data Protection Regulation

GG Gabriel Graph

GNSS Global Navigation Satellite System

GPS Global Positioning System

H

HM Hard Misses

I

ICT Information and Communication Technology

ISMA Index-based Subgraph Matching Algorithm

ISMAGS Index-based Subgraph Matching Algorithm with General Symmetries

ITS Intelligent Transportation System

L

LAD Look-ahead Distance

M

MAC Media Access Control

MSE Mean Squared Error

MST Minimal Spanning Tree

O

OBU On-board Unit
OPP Ordered Partition Pair
OSM OpenStreetMap

R

RCS Relative Correspondence Score
RE Roadside Equipment
RMSE Root Mean Squared Error
RNG Relative Neighborhood Graph

S

SDD Stationary Detector Data
SPRF Search Space Reduction Factor

T

THF Treiber-Helbing Filter
TMaaS Traffic Management as a Service
TRHC Trapezoidal Rush Hour Congestion

V

VMS Variable Message Sign
VSL Variable Speed Limit

X

xFCD Extended Floating Car Data

Samenvatting
– Summary in Dutch –

De laatste decennia zijn netwerken steeds meer deel gaan uitmaken van ons dagelijks leven. Ondanks het feit dat ze heel eenvoudig zijn, zijn ze uiterst geschikt voor het abstract modelleren van allerlei systemen, zeker in het veld van communicatietechnologie en transport. De veelzijdigheid van een netwerk laat toe om grote geconnecteerde systemen voor te stellen in een gestructureerd model. Met dit model kunnen de specifieke problemen beter worden geïdentificeerd, geanalyseerd en opgelost.

In dit doctoraat ligt de focus voornamelijk op Intelligente Transportsystemen (ITS), de verzamelnaam voor diverse ICT-systemen en technologietoepassingen die ons verkeer veiliger en efficiënter proberen te maken. In ITS worden netwerken vooral gebruikt om geconnecteerde wegennetwerken voor te stellen. Met deze modellering is het mogelijk om verkeersbegrippen zoals afgelegde voertuigritten, verkeersvolumes en reistijden uit te drukken in netwerkgrootheden zoals respectievelijk paden, linkcapaciteiten en linkkosten. Door deze netwerkmodellering worden de onderliggende problemen vereenvoudigd, krijgen we een duidelijker overzicht en kunnen we een efficiënte oplossing vinden.

Verkeersdata in ITS werd vroeger voornamelijk gegenereerd door lokale meetapparatuur op vaste locaties in of langs de kant van de weg (zoals tellussen in het wegdek). Deze data werd gekoppeld aan de plaatselijke netwerklinks en/of -knopen in het grotere (landelijke) netwerk, gevolgd door verdere dataverwerking die resultaten over het volledige netwerk opleverde. Deze data blijft echter heel plaatsgebonden en grootschalige sensornetwerken zijn zeer prijzig in installatie en onderhoud. De opmars van mobiele en geconnecteerde toestellen heeft gelukkig een nieuwe waardevolle bron aan verkeersdata opgeleverd onder de vorm van Floating Car Data (FCD). Deze data bestaat uit individuele voertuigritten die samen een inherent gedistribueerd beeld geven van het verkeer over een groot gebied. Echter, door het fundamentele verschil met klassieke meetsystemen is het alsnog onduidelijk welke informatie precies kan gehaald worden uit FCD. Kunnen we de data van de individuele voertuigen verwerken tot een algemener verkeersbeeld?

Om na te gaan welke informatie FCD bevat, maakten we in dit doctoraat eerst een evaluatie van de potentiële waarde van FCD voor netwerkanalyse. Hierbij gebruikten we FCD om een analyse te maken van een trajectcontrolesysteem op 2 belangrijke Belgische snelwegen. Met een set van FCD werd een beeld gevormd van de lokale verkeersomstandigheden en kon zowel het effect van het trajectcon-

trolesysteem als het effect van vaste snelheidscamera's op de snelwegen worden gekwantificeerd en vergeleken.

Nadat de trajectcontrolestudie aantoonde dat FCD interessant is voor lokale analyse, werd er gekeken naar de mogelijkheden voor grootschaligere netwerkanalyse. Hierbij lag de focus specifiek op hoe FCD gebruikt kan worden in netwerkroutering en hoe om te gaan met de gigantische hoeveelheid ruwe FCD (individuele voertuigposities). Hoewel deze ruwe vorm interessant is voor sommige verkeerstoepassingen, werken de meeste systemen op een compactere vorm waarbij de relevante verkeerseigenschappen al zijn afgeleid uit de data. Het verwerken van deze data en de voorstelling ervan in een gebruiksvriendelijk formaat is echter niet zo eenvoudig. Hierom werd in dit doctoraat dieper ingegaan op deze verwerking en werden verschillende modellen voor het voorstellen van tijdsafhankelijke snelheidsprofielen op onze wegen onderzocht. Voor de gepresenteerde modellen werd gekeken naar hoe goed zij de verkeersparameters in de data konden vatten en voorspellen. Eén van deze modellen, nl. het TRHC-model (Eng. Trapezoidal Rush Hour Congestion model), werd specifiek ontwikkeld om de verkeersevolutie voor te stellen met intuïtieve verkeersvariabelen zoals de start- en eindtijden van de spitsperiodes. Dit laat toe om netwerkalgoritmes en analysetools rechtstreeks te laten werken op de verkeersvariabelen zonder daarvoor de volledige profielen te moeten verwerken. Als voorbeeld van deze toepassing werd met de gegenereerde profielen een verkeerscongestiestudie gedaan voor Nederland met de focus op het identificeren van structurele pijnpunten in het wegennetwerk.

Na het onderzoek naar de geschiktheid van FCD voor netwerkanalyse, onderzochten we het potentieel van FCD voor dynamische verkeersmanagementtoepassingen. Deze systemen, waaronder onder andere variabele snelheidslimiteringssystemen (zoals variabele borden boven de snelweg), laten toe om vanop afstand in te grijpen in de actuele verkeerssituatie om zo de verkeersveiligheid en doorstroming te verbeteren. Hoewel deze systemen manueel bediend kunnen worden door verkeersoperatoren (bijvoorbeeld op basis van live camerabeelden), kunnen ze ook worden aangestuurd door automatische incidentdetectiealgoritmes (AID-algoritmes). Deze algoritmes gebruiken live verkeersdata om de verkeerssituatie te monitoren, problemen te detecteren en gepaste acties voor te stellen. De gebruikte verkeersdata is veelal afkomstig van plaatselijke sensorapparatuur specifiek geïnstalleerd voor de AID-toepassing. Binnen dit doctoraat werd onderzocht hoe FCD gebruikt kan worden binnen deze AID-toepassingen. Hierbij werden 2 haalbaarheidsstudies gedaan (1 offline, 1 online) waarbij een AID-algoritme ontwikkeld werd om te werken op FCD verzameld door een landelijk FCD platform. De resultaten van het FCD AID-algoritme werden vergeleken met het huidige operationele AID-systeem werkende op data van lokale inductieve lussensoren. Onze resultaten toonden aan dat het huidige FCD AID-algoritme in staat is om de verschillende technische hindernissen zoals lage technologieadoptie en data delay te overbruggen en zo een kostenefficiënte toevoeging en alternatief is voor inductieve lusdata.

Als afsluiter, na het presenteren van het nut van FCD voor netwerkanalyse en live verkeersmanagementtoepassingen, toont deze scriptie nog aan hoe FCD kan gecombineerd worden met andere databronnen in een datafusiealgoritme. Datafusie laat toe om verschillende bronnen en hun specifieke voordelen te combineren in één samengestelde bron. De individuele datapunten in de FCD en de inductieve lusdata werden hiervoor eerst genormaliseerd alvorens ze te combineren. Bij het combineren en extrapoleren van de data werd speciaal rekening gehouden met hoe de verkeerseigenschappen variëren langsheen een gemonitord traject. Dit datafusiealgoritme werd met succes toegepast op een studietraject in Nederland.

Naast de voorgestelde toepassingen van netwerkmodellering voor FCD werd er in dit doctoraat ook naar meer generische netwerkalgoritmes voor andere toepassingsdomeinen gekeken. Deze algoritmes zijn te vinden in de appendix. Het eerste algoritme bestaat uit een netwerkontwerpalgoritme voor het ontwerpen van fouttolerante netwerken. Dit algoritme is gebaseerd op een computationeel model van een slijmzwam. Het tweede voorgestelde algoritme gaat in grote netwerken op zoek naar kleinere netwerkjes met een vooropgestelde structuur. Het algoritme zelf mikt op het optimaliseren van het zoekproces door rekening te houden met de symmetrie vervat in het kleinere netwerk.

Summary

Over the last decades, the basic notion of a network has found its way into our everyday life. Thanks to their abstract modelling power and despite being very simple on their own, networks have been used in all kinds of domains, most importantly communication and transportation. In both domains, networks have proven to be very versatile, allowing large systems of interconnected components to be represented in a structured model. With this structure, the underlying problems can be more easily identified, analysed and solved.

In this dissertation, we primarily focus on the field of Intelligent Transportation Systems (ITS), consisting of all Information and Communication Technology (ICT) systems enabling a safer and more efficient traffic using a wide range of technology. In ITS, networks are widely used to model interconnected roads. Traffic variables like vehicle trips, traffic volumes and travel times are easily translated to network concepts like paths, link capacities and link costs. By using a network representation of the underlying problem, the problem context is simplified, allowing a clear overview and an efficient solution to be found.

Traffic data in ITS is traditionally generated at fixed locations using roadside monitoring equipment. This data is coupled to the network links/nodes at that position and further processing is done to obtain network wide results. However, this data is highly localised and large scale deployment and operation of roadside monitoring networks are costly. With the rise of mobile, networked computing devices, a valuable source of traffic data has become available termed Floating Car Data (FCD). This data consists of individual driver trajectories that, together, provide an inherently distributed view on traffic conditions over a large area. As this fundamentally differs from traditional data, it remains unclear to what extent FCD contains information on the traffic situation. Can we process the data of the individual vehicle probes to a more general view on the traffic situation?

In this dissertation, we first focus on what information FCD actually contains by looking at its potential for network analysis. In our exploratory research, we employed FCD to estimate the effect of a section control speed limit enforcement system on 2 important Belgian highways. For these 2 roads, a sample of FCD was used to obtain a local view on the traffic conditions. Using the FCD, our analysis showed the local effect of the section control system as well as the effect of speed cameras further on the routes. These results nicely demonstrate the potential for FCD as a traffic data source.

Following the results of our local section control study, we take a look at the suitability of FCD for large scale network analysis and how raw FCD can be trans-

formed into routing information. FCD in its raw form (individual vehicle positions) consists of huge amounts of data. While some applications work on those samples, most require a more condensed dataset in which the traffic properties are already distilled from the raw input feed. However, deriving information from the raw samples and aggregating it into a useable format is not so straightforward. We therefore delve deeper in the FCD processing and present several models for time dependent speed profiles. The models are first evaluated for their suitability to capture and predict traffic parameters. One of the presented models, the Trapezoidal Rush Hour Congestion (TRHC) model, aims at capturing the traffic dynamics in intuitive parameters, directly representing traffic properties e.g. start and end times of rush hour periods. This allows for algorithms and analysis tools to work directly on the parameters without the need to fully process the entire speed profile. This is shown in a congestion study for the Netherlands, identifying structural congestion bottlenecks.

After showing the suitability of FCD for network analysis, this dissertation turns to evaluating the potential of FCD for Dynamic Traffic Management (DTM). To improve traffic flow and safety, road operators often install dynamic systems like Variable Speed Limit (VSL) systems that can be remotely switched to manage traffic. These systems can be controlled manually by human operators (e.g. using video surveillance to decide if they should intervene) but can also be coupled to Automated Incident Detection (AID) algorithms. These algorithms use live traffic data to monitor traffic conditions, detect traffic problems and decide on appropriate actions. The live traffic data used traditionally comes from dedicated roadside sensors. To investigate the potential of FCD for AID, 2 feasibility studies (1 offline, 1 online) using real FCD were carried out. The existing installed AID system, working on loop induction data, is used as a reference system for the developed FCD-based AID algorithm. Results show that the current FCD AID algorithm can overcome technical challenges like delay and sample penetration rates and offer a cost-efficient addition and alternative to loop data.

With the usefulness of FCD for network analysis and live traffic management demonstrated, this dissertation concludes by showing how FCD can be combined with other data sources in a data fusion algorithm. Data fusion in general allows for multiple sources to be combined, resulting in the best of both worlds. FCD and loop data samples are first normalised and then combined, taking into account traffic conditions propagating along the route. The resulting algorithm was successfully applied to a use case in the Netherlands.

Aside from network modelling for FCD, this dissertation also resulted in some more generic network algorithms for other application domains. These algorithms are presented in the appendix. The first appendix presents a network design algorithm aimed at incorporating fault tolerance in the network design. The algorithm itself is based on a computational model of a slime mould. The second presented algorithm focuses on querying a large network for smaller subgraphs matching a specified subgraph network structure. The algorithm itself exploits the symmetry in the subgraph to optimise the search process.

"In the beginning there was nothing, which exploded."

— Terry Pratchett (1948 - 2015)

1 Introduction

This chapter sets the stage for the rest of this book by introducing some of the core concepts and terminology used throughout the different chapters. It serves to create the context for the book and to give the reader a broader understanding of the related fields and terminology used.

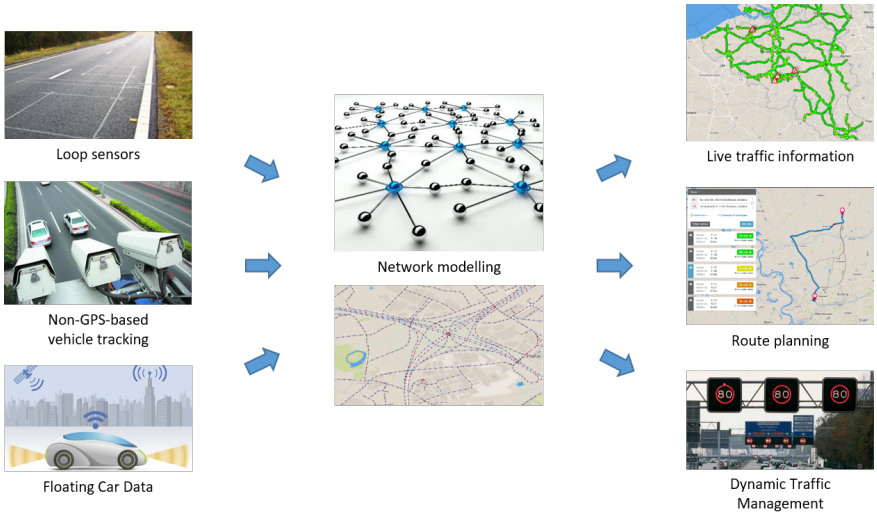
1.1 Context

Over the last decades, the basic notion of a network has found its way into our everyday life. Thanks to their abstract modelling power and despite being very simple on their own, networks have been used in all kinds of domains, most importantly communication and transportation. With various advances in wireless technologies, communication networks have become commonplace, with layers of network protocols on top of each other to hierarchically ensure connectivity. In the field of transportation, networks are the preferred modelling approach to represent different transportation modes e.g. car, train or foot. To get from A to B, countless route planning websites, smartphone apps and navigation units use network modelling in their underlying maps. Networks have become an essential part of these systems and this book aims at further harnessing their power.

Focusing on transportation and communication networks, the major application area of this book is found in their overlap in Intelligent Transportation Systems (ITS). ITS encompass all Information and Communication Technology (ICT)

systems enabling a safer and more efficient traffic using various technologies. By leveraging the power of networks, new applications can be developed to improve traffic conditions. ITS entail a wide range of systems building on top of each other, as shown in Fig. 1.1.

Figure 1.1 Overview data flow.



In the schematic view of Fig. 1.1, the basic modelling approach in this dissertation can be found. Traffic data is first collected and poured into a network model. The data in the model is then worked by a custom algorithm developed for the specific end user application. The following sections elaborate on the different building blocks of this chain.

1.2 Network modelling

In its core, a network consists of a collection of nodes connected to each other by links. Depending on the type of network, these nodes and links take different shapes. The World Wide Web consists of web pages linked by references, social networks consist of people connected through friendships or shared interests and animals consist of cells connected by vascular and nervous systems. Within the context of ITS, networks are used to model roads, with links and nodes being roads and crossroads respectively.

By interpreting the roads as a network, traffic properties can be defined as networks characteristics, allowing an abstracted view on the entire transportation network. Link costs are used to model travel times on roads while link flows are

defined as vehicle counts on the different links. When traffic becomes congested, travel times rise and link flows begin to drop.

To get an idea of current road network sizes, Table 1.1 lists some link and node counts for processed OpenStreetMap (OSM) [1] networks as well as the size of the source input. OSM is an open source data initiative aimed at providing free geographic information. Note that the reported source input sizes are of the unprocessed OSM datasets, also containing information not related to the road networks e.g. points of interest, building layout and land usage. However, the numbers do indicate that modern day applications need to deal with large networks and efficient algorithms and data structures are necessary.

Table 1.1: Current OSM road network sizes.

Network	#Links	#Nodes	Size
Luxembourg	111,973	86,719	22 MB
Belgium	996,953	763,346	308 MB
Spain	4,881,211	3,597,405	688 MB
Europe	68,458,748	53,476,599	20.1 GB

1.3 Data collection

At the base of ITS, we find the (raw) data collection methods, collecting data to be used by rest of the chain. Large differences exist between the different sources of traffic information, mostly determined by the monitoring technology. One of the earliest available and most basic data sources consists of individually reported jam and accident notifications, as provided by emergency services or people present at the event itself. While this provides very detailed information on the traffic event, the manual effort required for this (mostly localised) information created the need for more automated and widespread data collection methods.

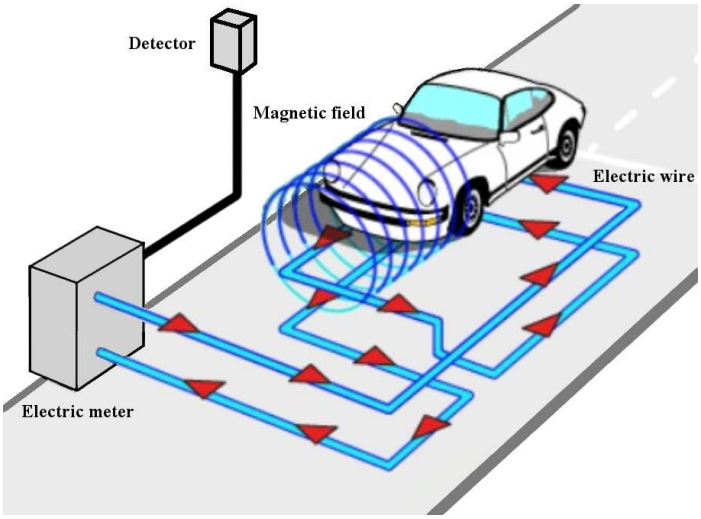
1.3.1 Inductive loops

In the 1960s, inductive loop sensors were introduced into traffic monitoring systems. These sensors monitor traffic by using inductive loop detectors embedded in the road surface as seen in Fig. 1.2.

1.3.1.1 Measurements

On the measurement level, an inductive loop sensor works on the physical principle of magnetic inductance. A conductive electric wire is embedded in the road surface and a small electrical current is sent through the wire to create a magnetic

Figure 1.2 Magnetic loop induction sensor. An electrical current is sent along a conductive wire embedded in the road surface. When the induced magnetic field gets disturbed by a passing vehicle, the electric meter registers a spike in the loop inductance and passes it on to the detector unit keeping track of all detected vehicles. Image from [2].



field. This magnetic field will oscillate at a specific frequency (depending on the wire properties and installation details). An electric meter is tuned to this frequency and monitors it for changes. When a car passes an inductive loop sensor, its metal components (e.g. engine and axles) pass through the magnetic field. This creates an electric current and an opposing magnetic field in the car, which in turn decreases the loop inductance. This change is detected by the electric meter and sent to the loop detector.

1.3.1.2 Processing

The loop detector performs the technical processing of the meter signals and communicates with the larger monitoring network. Depending on the type of loop sensor and the intended application, the loop detector can report each individual vehicle or aggregate the observed measurements to temporal averages. By installing successive inductive loops several metres apart from each other and measuring the time between the individually detected changes caused by a vehicle passing the loops, the loop detector can calculate individual vehicle speeds and vehicle lengths. Typically, the loop detector also reports the estimated average number of vehicles per hour. Each minute, this average is calculated and reported based on the observed individual vehicles during that minute. More advanced detectors classify

the different vehicles passing the loop. As larger vehicles have larger and/or more metal components, their impact on the loop inductance is different (bigger, longer in time and multiple peaks in the inductance measurement), allowing for vehicle classification.

1.3.1.3 Data properties

In terms of data properties, loop detectors can report individual vehicles sightings, average speeds and total vehicle counts. The reported average speeds are time mean speeds $\bar{v}_t$ (see Eq. (1.1)), as they average the m individual instantaneous vehicle speeds v_i at a specific location over a fixed period of time (e.g. 1 minute). However, in traffic flow theory, the space mean speed $\bar{v}_s$ is used to relate the flow rate q (= number of vehicles that pass a certain cross-section per time unit) and density k (= the number of vehicles per kilometre of road) as it is the quotient of the two (see Eq. (1.2)). The space mean speed can intuitively be seen as the average speed over all vehicles present in a fixed stretch of road at a given time. For applications using the loop input data, the difference between the 2 definitions needs to be taken into account. While the required average can be recalculated from the individual speed samples, the averages are also related as given in Eq. (1.3), with σ_s^2 denoting the variance of the averaged samples.

$$\bar{v}_t = \frac{1}{m} \sum_{i=1}^{m} v_i \tag{1.1}$$

$$\bar{v}_s = \frac{q}{k} = (\frac{1}{n} \sum_{i=1}^{n} \frac{1}{v_i})^{-1} \tag{1.2}$$

$$\bar{v}_t = \bar{v}_s + \frac{\sigma_s^2}{\bar{v}_s} \tag{1.3}$$

While loop detector accuracy in the beginning was questionable (with large errors in total vehicle counts and speeds) due to faulty installations, nowadays these problems are avoided through using dual loop detectors, proper installation procedures and calibrations. However, loop data quality tends to become unreliable at very low speeds, as the individual vehicle pulses are harder to distinguish from each other. Furthermore, while the loop detector can report individual vehicle sightings, it is generally infeasible to reconstruct vehicle trajectories as different sightings of the same vehicle at different sensor locations are hard to couple.

1.3.1.4 System deployment

Since its inception, inductive loop technology has been widely adopted and has become one of the most deployed monitoring systems for highways. Inductive loops are typically installed by government road operators. As they are largely dedicated

to monitoring the traffic, their cost is almost entirely attributable to the monitoring application. However, they only provide monitoring for the specific location where they are installed. To monitor an entire network, they are installed at various locations according to chosen deployment strategy. For Flanders, an estimated € 13.5 million was invested in 2011-2012 [3] in installing a base network of dual loop sensors at each highway junction and intersection (276 locations). The following years, the number of installations steadily rose (to 599 locations in 2017), adding an estimated additional € 20 million euros [4–6]. In the Netherlands, loops are even more prevalent, with over 33,000 measurement locations. In contrast to only installing loops at junctions in Flanders, in the Netherlands inductive loops on highways are typically installed every 500-600 m.

The previous numbers pertain to installation costs only and vary severely depending on the available infrastructure. For Flanders, a loop installation is estimated at € 75,000 for a location without pre-existing infrastructure [7]. While the hardware itself is relatively cheap, providing power and networking cable along the route, as well as closing traffic for safe installation is quite costly. Adding to the initial installation cost are the maintenance costs during the lifetime of the sensor. When a loop sensor fails and needs to be repaired, the road might need to be closed again, creating a lot of overhead costs for safety provisions and traffic rerouting. The average loop lifespan is estimated at 7 years, mostly influenced by the road tarmac degradation, which requires the loops to be reinstalled. Depending on when the repairs are executed (part of a planned road construction or an individual loop repair), the costs vary between € 500 and € 7,000. If the loops sensors are tightly coupled to traffic monitoring applications (necessitating quick repairs), the maintenance costs quickly rise and dominate the total system cost.

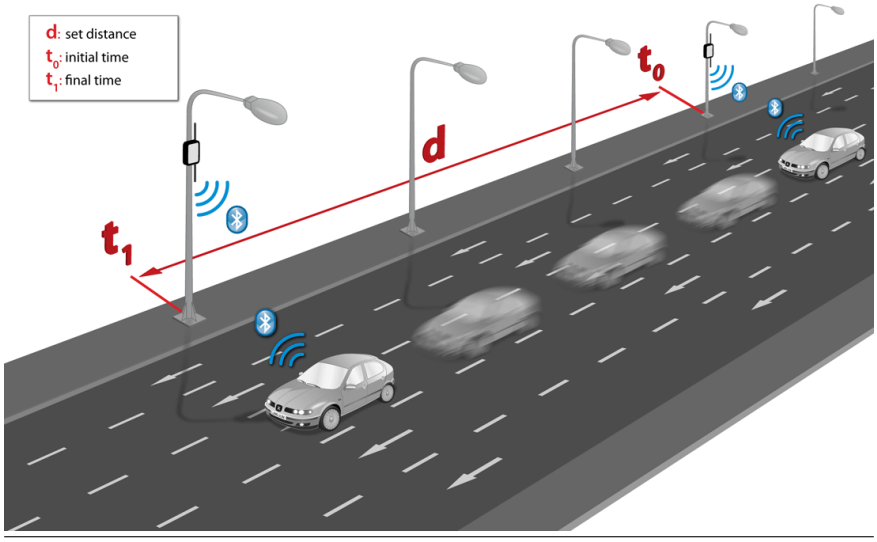
1.3.2 Non-GPS-based vehicle tracking

1.3.2.1 Bluetooth

Over the past decades, other monitoring technologies have emerged building on the success of various new technologies. One such example is wireless communication through Bluetooth. Fig. 1.3 shows how Bluetooth-enabled vehicles can be used to generate traffic information. Bluetooth detectors are placed by the side of the road with a set distance d between them. When a Bluetooth-enabled vehicle passes these detectors, its identifier (broadcasted Bluetooth MAC address) is logged together with a timestamp. Using the identifier, vehicles are re-identified along their route and their average speed can be calculated as $\bar{v} = d/(t_1 - t_0)$. This can be done along the entire vehicle trip.

From a technology point of view, this approach largely resembles loop induction technology but requires less expensive roadside infrastructure. However, ve-

Figure 1.3 Vehicle re-identification along the route. Successive sightings of a vehicle are combined to derive travel times along the vehicle trajectory. The known distance between the sensors is combined with the time difference measured between 2 sightings to obtain a speed estimate of the vehicle. Image from [8].



hicle monitoring is limited to Bluetooth enabled vehicles. This results in a limited view on the total traffic and flow rates can no longer be measured.

1.3.2.2 Licence plate recognition

Closely resembling the Bluetooth-based approach are licence plate recognition systems. Instead of monitoring Bluetooth MAC addresses, these video-based systems analyse imagery of passing vehicles for their licence plates and try to match these between successive detectors. This allows vehicle trip reconstruction for all vehicles but the performance of the system is sensitive to weather conditions (e.g. rain, fog, heavy wind) as they distort the camera imagery. The systems are widely used in speed limit enforcement systems (e.g. section control systems) to detect speed infractions.

1.3.2.3 Mobile phones

Lastly, several advances have been made to use data from mobile phones to estimate traffic properties. When a mobile phone is active, its data signal is handled by geo-tagged cell towers. This provides a very rough estimate of the location of the driver (and could potentially be improved by using multiple cell tower signals and their signal strength). While driving, the mobile switches between neighbouring

cells and causes handover events between cell towers. Based on the sequence of handover events, the vehicle trip can be constructed. The best matching road in an underlying map can be identified and (similar to Bluetooth) the travel times can be derived.

The mobile phone approach, however, suffers from several technical disadvantages. The positioning derived from cell towers is very coarse, making the results difficult to be pinpointed to the exact location of the vehicle. Additionally, trips need to be long and cover enough distance for cell tower handovers to occur. For congested traffic, this is problematic, resulting in a data loss at times when the system is needed the most. These disadvantages have hindered the breakthrough of the technology. It did however pave the way for the current day technology of the next section.

1.3.3 Floating Car Data

With the advent of mobile devices, better positioning hardware and improved network connectivity, a new source of traffic data called Floating Car Data (FCD) became available. FCD in its raw form consists of individual location measurements from probe vehicles transmitted to an external server for further processing.

1.3.3.1 Measurements

To obtain the location measurements, an FCD platform tracks individual probe vehicles along their route. These probes are typically vehicles equipped with an On-board Unit (OBU), smartphone or other mobile device that periodically logs its position (through GPS or another positioning system) for various purposes e.g. track-and-trace delivery systems, navigation assistance or driver behaviour analysis. By monitoring the whereabouts of the vehicles and their trips, a larger aggregated view on the global traffic state can be derived.

Depending on the technology used to generate the data, technical FCD properties vary significantly. The FCD used in this dissertation consisted of timestamped GPS samples generated by mobile on-board devices sampling their location every 1-60 s and transmitting it over the telecommunication network. High-frequency polling schemes, with location sampling every 1-5 s, provide high location accuracy and minimal delay but consume more power from the device. Low-frequency polling systems, with location sampling every 30-60 s, consume less power but necessitate path estimation techniques to determine the exact path followed by the vehicle in between successive samples. Individual FCD probe samples generally consist of a vehicle identifier and a set of coordinates. Additional information in the transmitted data sample (e.g. instantaneous speed, driving direction, vehicle length, weight, engine status) is also possible and is more generally called Extended Floating Car Data (xFCD).

1.3.3.2 Processing

To use the individual FCD measurements, they first need to be processed by the FCD platform. The first step consists of coupling the location information to an internal road network. This process is called *map matching* and aims at correctly mapping the input coordinates while filtering out the different sources of noise from the data. While GPS technology offers average accuracy around 5 m for normal devices, this accuracy worsens when in the neighbourhood of large buildings or bridges, necessitating additional denoising mechanisms. Other example sources of noise are GPS drift, transmission errors or misconfigured OBUs.

Upon receiving a new sample, the set of coordinates is used in a spatial query to determine a set of potential road segments in an underlying network representing the physical road network. From this set of candidates, the best candidate is determined taking into account all available information e.g. the distance of the segment to the sample, the heading of the vehicle and the previously received samples of the vehicle. While a single data sample can be mapped to a location on a map, multiple samples of the same vehicle can be processed together to create a vehicle trip. Combining this trip with the timestamps in the samples, travel times and speeds can be derived for each part of the route.

1.3.3.3 Data properties

As FCD has a fundamentally different measurement approach compared to inductive loop data, its properties are quite different as well. By starting from a vehicle-centric viewpoint, the data is inherently spatially distributed and vehicle samples are trivially grouped based on the vehicle identifier. Also note that again only a fraction of all vehicles is monitored as GPS-based FCD requires vehicles to be equipped with on-board devices connected to the FCD platform. As current FCD platforms have low penetration percentages (in the order of 1-20%), the derived results are only based on a sample of all traffic. This needs to be accounted for in the applications using FCD. Furthermore, as only an unknown fraction of the total traffic is monitored, FCD does not report total traffic flow counts.

One of the most important FCD properties is the *delay*. As traffic management applications want an accurate view on the actual traffic conditions, it is crucial to have as little delay between the FCD measurement time and the time it is available to the application. Several sources of data delay can be identified in the FCD chain. The first source can be attributed to the vehicle polling frequency. Vehicles log their location at fixed time intervals (T_{log}) e.g. every 10 s. However, they only transmit this data back to the FCD server periodically (T_{poll}) e.g. every minute. This introduces *polling/buffering delay*. If data samples are only received every minute, the average age of a data sample is already 30 s. Next, communicating the data from the device to the FCD server introduces a *transmission delay* of

several seconds. Finally, the processing of the samples results in a few seconds of *processing delay* to retrieve vehicle state, perform map matching and calculate travel times.

All these delays need to be minimised to avoid outdated traffic information. Additionally, when looking at a specific location in the network, an extra delay is introduced by the sampling process. Getting an update on the traffic state requires a new vehicle to pass the location. This inter-sample delay depends on the penetration rate (as having more probes increases the likelihood of a vehicle passing the location) as well as the time of day (with more vehicles being on the road during rush hours than at night).

As mentioned, the FCD properties vary significantly depending on the FCD platform source. In the remainder of this chapter and dissertation, an international industrial GPS-based FCD platform will be used and elaborated on. For the Netherlands, the platform currently monitors an estimated average of 6% of all traffic, translating to an average of 160 million samples per day. The bulk of the data samples were obtained from a popular smartphone navigation app providing free live traffic information to individual users. Other sources include professional parcel delivery trucks and various fleet management companies. As vehicle data is collected without specific vehicle information, no exact truck-to-car sample ratio is available and the dataset could be biased towards the users of the harvested sources. In Section 3.4, cars are estimated to generate 60-85% of all samples in the dataset, which corresponds to reference literature. The importance of the car-to-truck ratio and the sample distribution depends on the intended application. They are most relevant when looking at speed distributions (Chapt. 3) but less so when focusing on congestion situations (Chapt. 4 and 5) where traffic is more homogeneous.

A final note needs to be made regarding privacy. As FCD tracks individual drivers and their location, the raw FCD could contain sensitive information that could be coupled back to the individual user. Therefore, several mechanisms exist to protect driver privacy. However, a delicate trade-off needs to be made, as removing information can negatively impact FCD quality. For example, individual vehicle trips are anonymised so different trips of the same driver cannot be readily grouped. Sadly, this eliminates interesting driver behaviour studies. Even more aggressive mechanisms exist to further remove sensitive data from the dataset (e.g. removing parts of the trips) but these again reduce the value of the dataset. FCD providers therefore need to be careful when publishing results or providing data to external parties. The need for proper privacy management is also evident by the recent European General Data Protection Regulation (GDPR) requiring any private company (among which FCD providers) to provide end users with more clarity on the use of their data and give them more control over it.

1.3.3.4 System deployment

Current FCD platforms are typically commercially and/or privately owned with data being harvested from various sources e.g. track-and-trace systems, delivery companies and smartphone navigation applications. Compared to the loop system owned by government, the business model of an FCD platform is very different. With the hardware costs limited to some processing servers and the underlying map data model (varying depending on provider), the main issue becomes obtaining the data from the probes. The availability and quality of probe data heavily depends on the location (e.g. rural areas versus large cities) and the technological development in the area (e.g. developing countries versus technologically developed countries) but also on government policies (e.g. mandatory GPS/GNSS-based tolling in Belgium or fleet tracking in Brazil).

Aside from the cost of the input data, FCD platforms also differ from traditional loop systems in terms of revenue. FCD platforms typically feed various applications (e.g. real-time travel time services, historical analysis products, fleet management systems and telematics) each having their own revenue streams. This allows the cost of the FCD platform itself to be shared across different clients and applications, none of them needing to pay for the entire platform.

A last noteworthy detail is that FCD platforms themselves differ significantly in terms of penetration, technology, coverage, data properties. These technological properties are sensitive information and often not freely available. Additionally, commercial offerings of FCD providers are even more confidential and differ depending on the client, area and company strategy. Given all these factors, a simple cost estimate for FCD is infeasible (see [9] for an early estimate and how different factors like dedicated OBUs versus smartphone applications can affect system cost up to 2 order of magnitude). However, reports from industry (see [10] for a report of 75% operational cost reduction of FCD compared to loop data) do promise significant gains. With more and more vehicles becoming available as FCD probes, the cost of FCD will further decrease, which only strengthens the FCD business model.

1.3.4 Inductive loops versus FCD

To summarise the main differences between the inductive loop data and FCD, Table 1.2 lists the main points of interest mentioned above. While it is clear that both sources fundamentally work differently, they both report on the same traffic conditions. How these sources compare to each other (in terms of data quality and application suitability) will be discussed further on in this book.

Table 1.2: Comparison between inductive loops and FCD.

	Inductive loops	**FCD**
Measurements	Roadside sensors Electromagnetic	Probe vehicles tracking GPS
Processing	Individual sightings = instantaneous speeds	No instantaneous speeds (in general), requires averaging multiple samples of the same vehicle
Data properties	Point-based No vehicle trajectories Full traffic monitoring Traffic volumes and speeds	Spatially distributed Vehicle trajectories Small percentage of total traffic Only vehicle speeds
System properties	Government property Dedicated traffic monitoring	Privately owned Multiple commercial applications

1.4 Application algorithms

Building on the network modelling, various applications can be developed using the traffic information for traffic optimisation. With technology maturing and monitoring becoming increasingly available, new ideas and applications are found every day. We now list a few examples of the most common applications relevant to this dissertation.

The most common use of FCD is calculating travel time on routes. Starting from a road network enriched with live travel times on each road segment, travel times on specific routes can be calculated and communicated to drivers to inform them on the current, possibly congested traffic situation. By comparing the travel times to normal, uncongested network conditions, traffic jams can also be detected. This allows for automated traffic event notifications, informing drivers as soon as possible and allowing them to optimise their route.

Aside from using live travel times for real-time event generation, the network state can also be monitored for longer periods of time to perform network analysis. By logging road network state for long periods of time, large sets of historical data can be built, allowing traffic patterns and structural network problems to be detected. This information can then, for example, be used in planning road works, forecasting traffic congestion or assisting mobility policy makers.

Going one step further than monitoring the current situation is acting on live information. Ideally, when traffic congestion is detected, an automated ITS application should try to alleviate the problem. One such application is found in variable speed limiting systems. Depending on the traffic state, these systems impose speed limits on the passing traffic to alert drivers of upcoming traffic problems. This lets them slow down gradually, thereby avoiding tail-of-queue collisions. They also try to harmonise traffic by smoothing out stop-and-go congestion waves, reducing speed differences between cars and improving safety.

1.5 Research challenges

With the above context, we can now describe the research performed and group the different contributions in overall research challenges. In the context of ITS, the emergence of FCD as a new technology offered interesting network modelling questions. While traditional data sources are mostly localised sensor measurements, the spatially distributed FCD potentially offers new insights in traffic dynamics. With careful modelling and data processing, information is available on a large scale. However, how valuable is this data? As only a fraction of the total traffic volume is monitored, it remains unclear how good traffic properties can be derived and under which circumstances. Focusing on this question, **Challenge 1** consists of **evaluating the potential of FCD for network analysis**. While large

amounts of data are available, it is on a microscopic level, being the single vehicle location. With only a limited set of probe vehicles, the high level traffic properties need to be estimated. A straightforward averaging of all vehicle speeds to obtain an average speed (as done by loop sensors) becomes quite complex. However, if successful, FCD allows for large scale network analysis, yielding valuable information in previously unmonitored locations.

As mentioned above, the purpose of ITS is to improve traffic. Network analysis allows to identify problem areas which can then be tackled by various actions e.g. constructing new roads to increase capacity, tuning traffic light signalling to optimise waiting times or imposing speed limits to reduce traffic speed at hazardous locations. However, ITS can also use FCD as an input data source to determine if they should take action. These systems are termed Dynamic Traffic Management (DTM) systems and regulate traffic with live data. Integrating FCD in these systems presents the next step in the FCD technology evolution. To investigate the suitability for FCD, **Challenge 2** can be stated as **evaluating the potential of FCD for dynamic traffic management**.

A last research challenge is found when taking a step back from the ITS application domain to more general network applications. As mentioned, networks are used in various application areas as they allow to capture specific technical properties in an application or domain in general network properties. With that in mind, the research in this PhD started off with creating and improving more generic network-oriented algorithms suited for various purposes and later evolved towards ITS. As this work was chronologically tackled first, we group it under **Challenge 0**, consisting of **developing generic network-oriented algorithms**. While some part of the work around ITS and FCD can also be generalised for other domains, this challenge is mostly limited to our work in the appendices, as discussed in the next section.

1.6 Contributions and outline

This dissertation is composed of a number of publications that were realised within the scope of this PhD. The selected publications provide an integral and consistent overview of the work performed. The complete list of publications that resulted from this work is presented in Section 1.7. Each main chapter and appendix contains a publication as-is. Within this section, we give an overview of the remainder of this dissertation and explain how the different chapters are linked together. Fig. 1.4 schematically shows how the different chapters are related to the research challenges.

In the context of ITS, FCD presents a valuable novel source of network data for a wide range of applications. Focused on evaluating the potential of this data for network analysis (Challenge 1), we investigated if and how FCD can be used

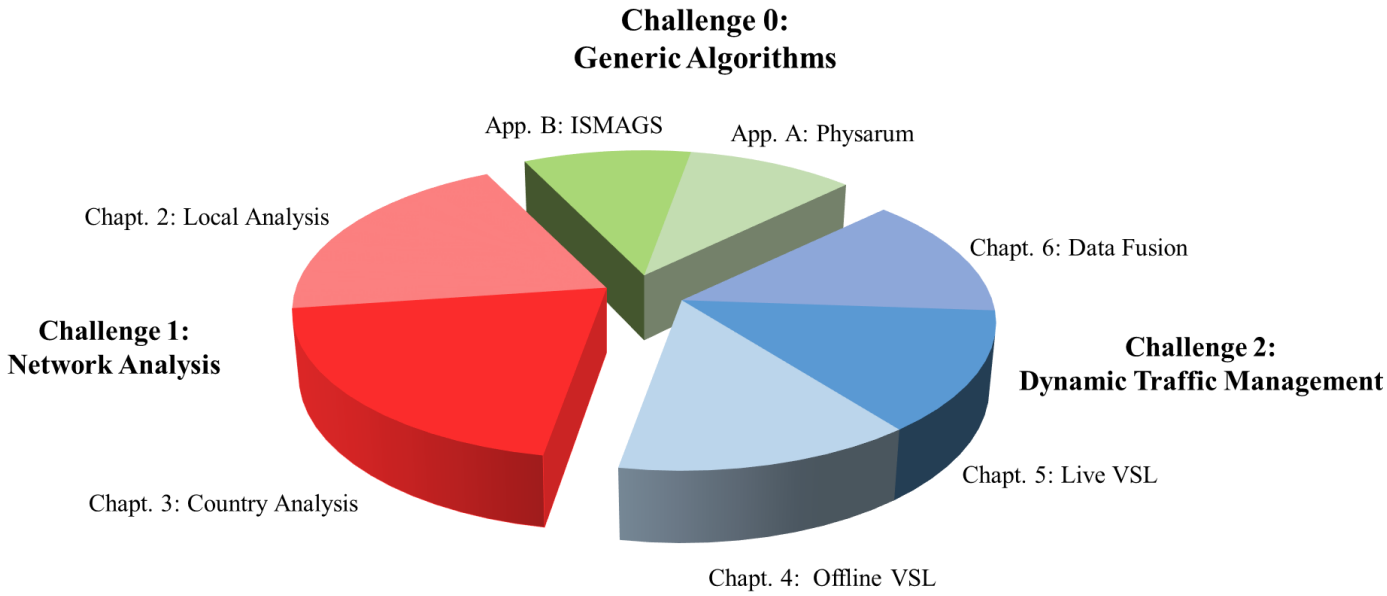


Figure 1.4: Schematic grouping of the chapters & challenges in this dissertation.

to derive high level traffic characteristics under varying traffic situations on motorways. More specifically, by statistically analysing the individual user travel times in our FCD, we evaluated the impact of 2 speed limiting systems (section control system and fixed speed cameras) on driver behaviour. This exploratory research is presented in **Chapter 2**. Results showed that the various traffic situations on the routes (road works, speed cameras, section control systems and on- and off-ramps) could be identified in the FCD (consisting of vehicle trips with low-frequency sampling) without traditional (loop) measurement systems. Furthermore, the presented analysis method of the study demonstrates that the effects of the speed enforcement policies can be quantified using FCD. In contrast to most other studies using either loops or simulated data, our study is based on real FCD, showing its value for network analysis.

Convinced by the available information in the FCD, the suitability for network analysis with FCD was further investigated on a countrywide scale, again tackling Challenge 1. As FCD is generated on an individual user level, aggregation methods are necessary to distill information from the data. **Chapter 3** presents several aggregation methods to obtain workable network data to be used in network algorithms as well as large scale analysis e.g. charting congestion hot spots. We present the Trapezoidal Rush Hour Congestion (TRHC) model which is specifically designed to capture travel time evolution throughout the day in intuitive parameters, easily used in further network applications.

Aside from using FCD for network analysis, it can also be leveraged directly into traffic management algorithms. While Chapter 2 presented an evaluation of installed speed limiting systems, those systems could also be coupled to an FCD feed, replacing and/or supplementing existing data input (Challenge 2). **Chapter 4** presents a feasibility study of an FCD speed limiting algorithm in a dynamic traffic management system in an offline evaluation context. The existing system was adapted to work with our FCD algorithm (working on high-frequency FCD). The main contribution of this study was to provide a scientific analysis of the feasibility of FCD VSL using real FCD, in contrast to simulation data commonly used in literature.

Following our offline study of Chapter 4 and to continue taking on Challenge 2, the FCD-based algorithm was further developed and studied in a live setting to investigate the real-time performance. One of the major points of interest here was to counter the different inherent types of delay in the FCD algorithm. This study is given in **Chapter 5**. The results here show that our FCD VSL system (working on high-frequency sampled FCD vehicle trajectories) has acceptable performance compared to the loop-based VSL and that the experienced technical delays can be overcome. To the best of our knowledge, the presented

A third and final algorithmic FCD application can be found in data fusion. In the previous chapters, the presented work mostly only uses FCD. However, the

available (traditional) data input streams can also be enriched with FCD to obtain a more detailed view on the entire situation. The combination of both data sources is termed data fusion. During the course of the PhD, a data fusion algorithm was developed in cooperation with an industry partner and has since been patented (see publication 3 in 1.7.4). **Chapter 6** presents a short application of this work which tackles Challenge 2 by exploring the suitability of FCD for live fusion DTM.

To combine some of the key results of the work performed during this PhD, **Chapter 7** provides an overall conclusion and looks ahead to identify future challenges and opportunities.

This book is concluded by 2 appendices with 2 more publications obtained during this PhD in the field of network modelling and algorithmic design focusing on Challenge 0. **Appendix A** presents a network design algorithm focusing on fault tolerance provisions. The algorithm itself is based on a computational model of a natural organism *Physarum polycephalum* which was adapted to account for network failure. **Appendix B** contains an algorithm for enumerating small subgraphs contained in a larger network. The algorithm itself exploits symmetry information to more efficiently process the entire search space.

1.7 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during the PhD research.

1.7.1 Publications in international journals (listed in the Science Citation Index[1])

1. **Maarten Houbraken**, Sofie Demeyer, Dimitri Staessens, Pieter Audenaert, Didier Colle, and Mario Pickavet. *Fault tolerant network design inspired by Physarum polycephalum.* Published in Natural Computing, vol. 12, issue 2, pp. 277-289, June 2013.
   DOI: 10.1007/s11047-012-9344-7

2. **Maarten Houbraken**, Sofie Demeyer, Tom Michoel, Pieter Audenaert, Didier Colle, and Mario Pickavet. *The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS): Exploiting Symmetry for Faster Subgraph Enumeration.* Published in PLOS One, vol. 9, issue 5, May 2014.
   DOI: 10.1371/journal.pone.0097896

[1]The publications listed are recognised as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

3. Mario Vanlommel, **Maarten Houbraken**, Pieter Audenaert, Steven Logghe, Mario Pickavet, and Phillipe De Maeyer. *An Evaluation of Section Control based on Floating Car Data.* Published in Transportation Research Part C: Emerging Technologies, vol. 58, pp. 617-627, Sept. 2015.
DOI: 10.1016/j.trc.2014.11.008

4. Thomas Van Parys, Ine Melckenbeeck, **Maarten Houbraken**, Pieter Audenaert, Didier Colle, Mario Pickavet, Piet Demeester, and Yves Van de Peer. *A Cytoscape app for motif enumeration with ISMAGS.* Published in Bioinformatics, vol. 33, issue 3, pp. 461-463, Feb. 2017.
DOI: 10.1093/bioinformatics/btw626

5. **Maarten Houbraken**, Steven Logghe, Marco Schreuder, Pieter Audenaert, Didier Colle and Mario Pickavet. *Automated Incident Detection Using Real-Time Floating Car Data.* Published in the Journal of Advanced Transportation, vol. 2017, Dec. 2017.
DOI: 10.1155/2017/8241545

6. **Maarten Houbraken**, Steven Logghe, Pieter Audenaert, Didier Colle, and Mario Pickavet. *Examining the potential of Floating Car Data for Dynamic Traffic Management.* Published in IET Intelligent Transport Systems, Jan. 2018.
DOI: 10.1049/iet-its.2016.0230

1.7.2 Publications in international conferences

1. Thijs Walcarius, **Maarten Houbraken**, Pieter Audenaert, Mario Pickavet, and Piet Demeester. *Modeling real-world vehicle routing problems with dependencies.* 6th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), Aug. 2013.

2. **Maarten Houbraken**, Pieter Audenaert, Didier Colle, Mario Pickavet, Karolien Scheerlinck, Isaak Yperman, and Steven Logghe. *Real-time traffic monitoring by fusing Floating Car Data with stationary detector data.* Models and Technologies for Intelligent Transportation Systems (MT-ITS), June 2015.

3. **Maarten Houbraken**, Steven Logghe, Pieter Audenaert, Didier Colle, and Mario Pickavet. *Modelling traffic congestion evolution through speed profile generation using Floating Car Data.* 25th ITS World Congress (ITSWC), Sept. 2018.

1.7.3 Publications in national conferences

1. **Maarten Houbraken**, Thijs Walcarius, Sofie Demeyer, Pieter Audenaert, Didier Colle, and Mario Pickavet. *An insertion-based heuristic for the constrained pickup and delivery problem.* Annual Conference of the Belgian Operations Research Society (ORBEL), Jan. 2013.

2. Sofie Demeyer, **Maarten Houbraken**, Thijs Walcarius, Pieter Audenaert, Didier Colle, and Mario Pickavet. *Using contraction hierarchies to find dissimilar paths in transportation networks.* Annual Conference of the Belgian Operations Research Society (ORBEL), Jan. 2013.

3. **Maarten Houbraken**, Pieter Audenaert, and Mario Pickavet. *ISMAGS: Speeding up subgraph enumeration using dynamic node ordering and symmetry analysis.* 4th MultRP N2N Bioinformatics Symposium, May 2014.

1.7.4 Other publications

1. **Maarten Houbraken**, Pieter Audenaert, and Mario Pickavet. *Design of fault tolerant networks: the slime mould approach.* 12th UGent-FEA PhD Symposium, Dec. 2011. 3rd laureate junior researcher track.

2. **Maarten Houbraken**, Pieter Audenaert, and Mario Pickavet. *Tackling traffic congestion: modelling, analysis & prediction.* 14th FEA PhD Symposium, Dec. 2013. 1st laureate senior researcher track.

3. Steven Logghe, Wim Vanbelle, Isaak Yperman, Karolien Scheerlinck, **Maarten Houbraken**, Wim Vandenberghe, Pieter Callewaert, Bert De Rijck, Jens De Valck, Pieter Pauwels, and Mario Vanlommel. *A traffic data fusion system and the related method for providing a traffic state for a network of roads.* European patent EP3035314B1, Nov. 2017.

References

[1] OpenStreetMap contributors. *OpenStreetMap*, 2018. Available from: https://www.openstreetmap.org.

[2] T. Harris. *How Stuff Works*, 2001. Available from: www.howstuffworks.com/car-driving-safety/safety-regulatory-devices/red-light-camera.htm.

[3] H. Crevits. *Dertien miljoen om het verkeer te meten*, 2009. Available from: http://www.hildecrevits.be/nl/dertien-miljoen-om-het-verkeer-te-meten.

[4] *Eerste Verkeersindicatorenrapport met precieze gegevens over aantal en type voertuigen op de autowegen in Vlaanderen*. Technical Report april, Ministerie van Mobiliteit en Openbare werken, 2011.

[5] *Projectfiche Dynamisch Verkeersmanagement*. Technical Report December 2015, Agentschap Wegen en Verkeer, 2015.

[6] *Projectfiche Dynamisch Verkeersmanagement*. Technical report, Agentschap Wegen en Verkeer, 2016.

[7] *Schriftelijke vraag Autosnelwegen - Filedetectie*. Technical report, Ministerie van Mobiliteit en Openbare werken, 2010.

[8] Libelium. *Smartphone Detection*, 2018. Available from: http://www.libelium.com/products/meshlium/smartphone-detection/.

[9] E. Vanhauwaert, W. Vandenberghe, P. Demeester, I. Moerman, and S. Verbrugge. *Feasibility of expanding traffic monitoring systems with floating car data technology*. IET Intelligent Transport Systems, 6(4):347–354, 2012.

[10] T. Blanken. *"Onze verwachtingen zijn meer dan waargemaakt"*. NW magazine, (1), 2017.

"If we knew what it was we were doing, it would not be called research, would it?"

— Albert Einstein (1879 - 1955)

2

An Evaluation of Section Control based on Floating Car Data

This chapter presents the first steps in discovering the potential of FCD for network analysis. FCD offers spatially distributed data samples containing data on the traffic situation. By statistical aggregation and analysis, the underlying high level traffic properties can be derived. This chapter demonstrates this approach on 2 Belgian highways. The traffic is characterised and the effect of the installed speed limit enforcement systems is investigated. Compared to other research using inductive loop data or small and/or simulated FCD, this study uses real FCD collected by an operational FCD system. This reduces the modelling assumptions and more clearly shows how the underlying phenomena can be studied by FCD.

This work was done jointly with Mario Vanlommel. He provided the data and the initial map matching methodology. The remainder of the study (methodology refinement, implementation, data analysis and result interpretation) was done by the main author of this book.

⋆ ⋆ ⋆

Mario Vanlommel, Maarten Houbraken, Pieter Audenaert, Steven Logghe, Mario Pickavet, Philippe De Maeyer

Published in Transportation Research Part C: Emerging Technologies, September 2015.

Abstract Floating Car Data (FCD) consists of positional information from moving vehicles and is ideally suited to monitor traffic conditions in large road networks. Given a large mobile fleet equipped with the FCD system, FCD allows to determine actual travel times on road segments/links and determine traffic state. Based on this link-specific information, we analyse the effectiveness of section control, also called average speed control, on 2 Belgian highways. Section control systems measure the travel time of vehicles between different positions to determine the speed limit compliance. Results show that the section control systems reduce speed limit violations over the entire length of the system, contrary to the local effect of fixed traffic cameras. Section control systems also reduce speed differences between vehicles in the same section as the speed dispersion is much smaller compared to similar, uncontrolled parts of the highway.

2.1 Introduction

The monitoring of traffic conditions has evolved during the past decades. Traffic and accident reports from people on the scene were two of the earliest sources of traffic information. Following the digital wave, the monitoring was complemented with information from roadside sensors like inductive loop counters and traffic cameras. These systems provide accurate information about the current traffic situation but have the disadvantage to be limited to their specific location and the need to install and maintain them. To completely cover the entire road network, every road would have to be equipped with these sensors, which is infeasible due to the size of the network and the maintenance costs of the sensors. Furthermore, the installation of cameras to detect traffic conditions can influence the objective measurement of speed, because drivers can falsely interpret these as speeding cameras and adapt their driving speed accordingly. Loop detectors have the disadvantage that they only measure speed at specific locations and cannot measure speed over longer road stretches or in a network wide area.

However, the penetration of mobile (navigation) devices has opened the door for vehicle-based monitoring systems. These systems exploit the positional information of the device to produce Floating Car Data (FCD) and are currently rolled out using numerous technologies in various countries. The aim is mainly to provide commercial traffic information and navigation services, but they can also be used for other commercial applications e.g. personalised insurance fees. The scientific description of FCD is mainly based on proof of concepts on test sites. The main problem for FCD systems is the available number of vehicles [1, 2]. While vehicle-based monitoring provides information on a larger area than fixed sensors, a larger number of vehicles is needed to accurately monitor traffic conditions. FCD systems have previously been tested [3] but the small scale of the setup and the poor penetration rate limits the significance of the results. Large scale experiments

are harder to set up due to the technical difficulties in hardware diversity and communication systems. However, if these challenges are overcome, FCD can be used to predict traffic conditions [4] and to study traffic dynamics. As FCD consists of measurements of the constantly changing travel conditions on roads and is collected by different drivers with different driving styles, it is inherently stochastic and can be used to create enhanced stochastic routing algorithms [5].

On Belgian highways, a lower speed limit of 70 km/h and an upper speed limit of 120 km/h are posted for cars while trucks have a maximum speed limit of 90 km/h. In order to improve traffic safety, the Flemish government installed a section control camera system on the E40 highway that became fully operational at the end of March 2013. This system measures the average speed of drivers over a length of 7.4 km of highway and is complemented on the E40 by 2 fixed speed cameras posted further along the highway. A similar section control system on the E17 spanning 1.9 km had already been operational since June 2012 in one direction and was expanded to monitor both directions in March 2013. The monitored section on the E17 is limited to 90 km/h for all vehicles. Note that all discussed speed cameras in this study are clearly marked and visible to all drivers, in contrast to hidden/mobile patrol cameras which are not discussed here.

During the first week of operation, the section control system on the E40 registered 6,632 speeding infractions in the direction of Brussels and 7,477 in the direction of Ostend. The normal traffic load varies on an average weekday between 50,000 and 70,000 vehicles (in each direction) [6]. The system on the E17 had 4,525 infractions (in the extra direction) in the same period. The activation of both systems was publicly announced and clearly indicated by road signs. As stated in a report of the OECD/ECMT [7], speeding is very common on the European highways. For Belgian highways, the BIVV conducted a study in 2011 [8], showing 45% of cars exceeding the speed limit with the 85th percentile at 131 km/h.

To evaluate the impact of the speed control measures on the overall traffic safety, three properties are of high interest: the average speed, the maximum speed and the speed dispersion/variation. This focus is supported by the detailed literature review of the effectiveness of average speed control systems presented in [9]. Least intuitive is the effect of speed dispersion, the difference in speed between cars driving along the same piece of highway at the same time. The relations of the three properties to crash rates is also discussed in [10] by giving an overview of several studies. They note that several factors influence the crash rate but their review does show an important correlation of the above three properties with high crash rates. While most people recognise the dangers of speeding, [11] reports that some offenders do not see speeding as a serious violation. In their research, they categorise offenders in several subclasses with one of the classes consisting of people that actively slow down for the camera to avoid getting caught and speed up afterwards.

In this paper, we evaluate the effectiveness of the section control camera system using Floating Car Data (FCD). The FCD delivers speed measurements for the entire road network. We first discuss our data capturing method and our extensions to avoid reported inherent FCD problems. We then show the results of the data capturing on two long highway stretches including multiple speed control systems, section-based and location-based, and discuss the effect of the speed control measures on the passing traffic.

2.2 FCD capturing

For this research, we used a fully deployed and operational FCD system. In this section we explain the underlying methodology of this FCD system and how it is used to capture the individual vehicle speeds on the complete road network by processing vehicle positions.

The FCD data is generated on a detailed digital representation of the road map. This map is composed of unidirectional road segments s, each defined by their segment identifier SID_s, coordinates pos_s and length l_s of at most 50 metres. With this granularity, it is possible to assume uniform traffic on a single road segment (also see [12]). Each segment is attributed with a free-flow travel speed. This speed is indicative of how fast it can be traversed and is set according to the applicable speed limit and other road properties. For example, on the Belgian highway, these speeds are set to 120 km/h as this is the local speed limit. Differences with the posted speed limits can occur due to road properties, for example sharp turns in roads which are traversed slower than allowed by the speed limit.

Raw vehicle positions are used as the input to the system. A vehicle is represented by an identifier *ID*, a location p (GPS coordinates) and a timestamp t of when the position was recorded. Updated positions of the driving vehicles are received every minute, leading to a set of data for vehicle *ID* of $\{p_1, t_1\}$, $\{p_2, t_2\}$, ... A set of preprocessing and filtering rules is applied to prevent mismatches and invalid positions. For example, positions must come from actual driving vehicles, within the area of consideration, valid timestamps and within the required update frequency.

Based on the successive points of an individual vehicle, the driving path of the vehicle is reconstructed on the digital map. This is done by selecting a set of candidate segments in the map in the neighbourhood of the raw position data as described in a map matching algorithm [13].

Within the next steps, the potential driving paths between all combinations of these candidate segments are calculated. The routes are calculated based on the underlying graph representation assuming the vehicle is driving the free-flow speed and using a shortest path algorithm like Dijkstra [14]. The shortest path between all these candidate segments leads to the selection of the chosen candidate

segments for the vehicle positions and the driving path of the vehicle. This driving path consists of an ordered list of segments where the vehicle has been driving.

Upon receiving a new measurement, the new position is linked to the already established path of the vehicle. If the position is near a road, all possible consecutive routes extending the existing path are calculated starting from the last previously known position of the vehicle. If no possible route is found within realistic speed restrictions, the new position is not added to the previous path, but is considered as the start of a new path. This avoids inserting incorrect information into an already verified path.

Based on the map matched driving path and the timestamps of the vehicle positions, the actual travel time on each segment of the driving path is calculated. Fig. 2.1 shows how the free-flow travel time along the path $T_{path,free\text{-}flow}$ is known from the digital map. The timestamps of the vehicle positions lead to an actual travel time along the path: $T_{path,actual} = t_2 - t_1$. We now allocate an actual vehicle travel time to each segment of the path. To do so, the free-flow travel time of the individual segment is scaled according to the difference in actual and free-flow travel time over the complete path.

$$T_{segment,actual} = \frac{T_{path,actual} \cdot T_{segment,free\text{-}flow}}{T_{path,free\text{-}flow}} \tag{2.1}$$

These principles are shown in Fig. 2.1. Based on the travel time of each vehicle on a segment, the individual vehicle speeds can be calculated using the segment lengths.

The key strengths of this FCD approach lie in the composition of the detailed map with very short segments and in the map matching of successive GPS positions of an individual vehicle. By receiving vehicle positions every 60 seconds, the map matching process can be assured to deliver good matches, especially on motorways.

2.3 Analysis

2.3.1 Data

The FCD technology is installed in an operational fleet of more than 200,000 vehicles in Belgium, having a registered total fleet of approximately 7 million vehicles. To evaluate the effect of the speed control measures, the outcome of the map matching method on two relevant highways is extracted. The two highways and their surrounding area are depicted in Fig. 2.2 below.

The datasets S_{E40} and S_{E17} were collected in the period of April 15 to May 8, 2013. In Fig. 2.3, the black dotted line shows the histogram of the dataset S_{E40} consisting of all measurements m during the 17 weekdays on the E40, one of

Figure 2.1 Travel time allocation. The travel time of a vehicle on the individual road segments is calculated by first assuming it drives the free-flow speed of the segments. The resulting travel time allocation is scaled to match the elapsed time between the measurements.

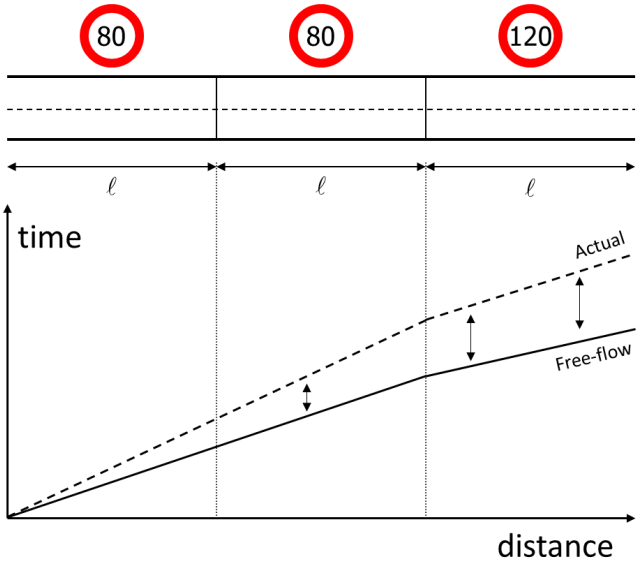


Figure 2.2 Highway layout. The E40 highway goes from Brussels (bottom right) to Ostend (top left). The E17 highway runs from Antwerp (top right) to Kortrijk (bottom left). The exits are indicated on the map by their names as they appear on the signs along the highway. The minimap shows the location of the highways in Belgium together with the major cities they connect.

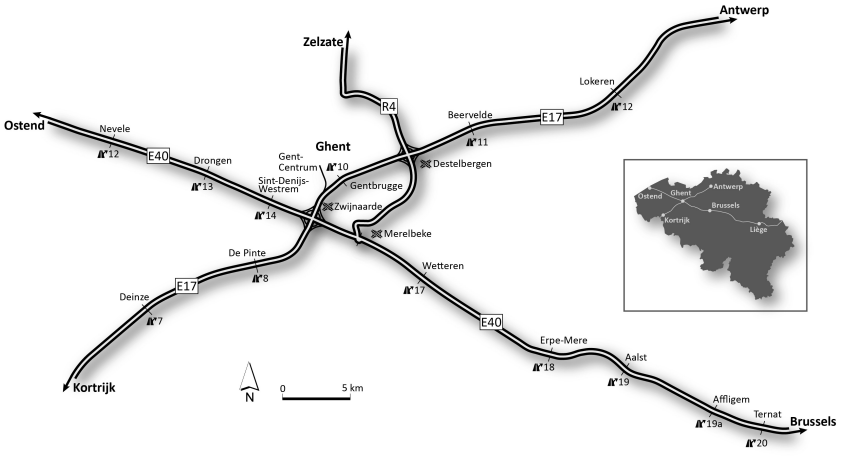
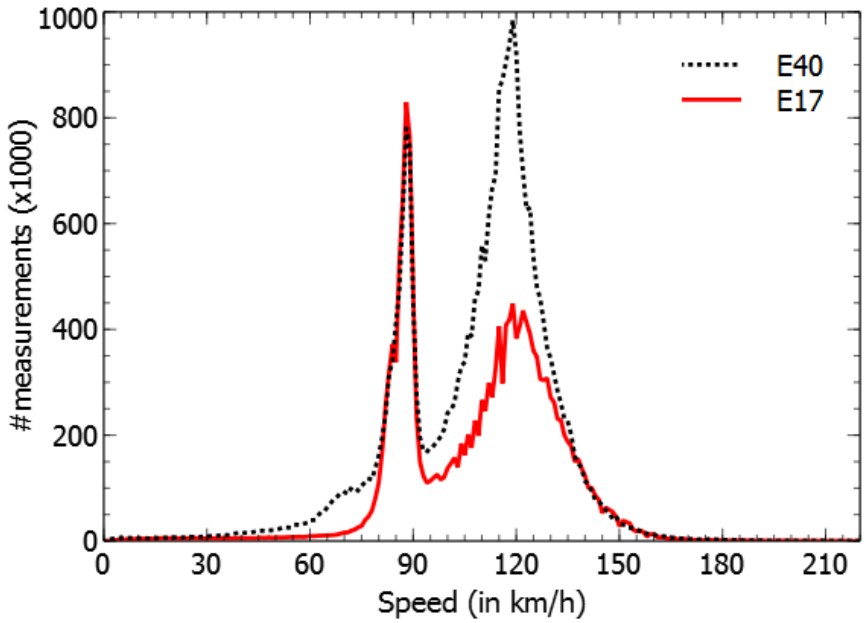


Figure 2.3 Speed histogram. The dotted black line denotes the number of measurements for the 17 weekdays in the E40 dataset S_{E40} for the corresponding vehicle speed. The solid red line shows the same for the S_{E17} dataset.



the major highways in Belgium. The measurements were taken on a 50 km stretch passing Ghent from Ternat to Nevele. The 2 spikes correspond to the general speed limits imposed on Belgian highways: 120 km/h for cars and 90 km/h for trucks. The solid red line shows the same for the second dataset S_{E17} with measurements collected on the E17 from Lokeren to Deinze, also passing Ghent. Each measurement m consists of a timestamp d_m, a measurement segment identifier *MSID* and a travel time t_m. The speed of the vehicle is calculated by the ratio of the length of the corresponding segment l_s over the travel time.

For the first dataset S_{E40}, a total of 2.93 million measurements were captured for the 50 km passing Ghent which is divided in 1,064 segments of approximately 50 m. The second dataset S_{E17} contains 1.86 million measurements for 35 km divided in 784 segments. S_{E17} has fewer measurements as it pertains to a shorter stretch of highway. Note that by using FCD, it is possible to analyse large stretches of highway, while more traditional roadside monitoring sensors (and their associated studies) are mostly limited to the monitored points themselves [9]. FCD obtains a full speed profile for a wide area, thereby providing ample points of reference for the study.

While nominally the number of measurements around 90 km/h is the same for both datasets, there are relatively more measurements for the E17. For S_{E40}, 22.9% of the measurements were below 90 km/h while S_{E17} had 27.3%. This can be attributed to the varying speed limit and to a higher percentage of trucks on the E17. The section control on the E17 monitors a speed limit of 90 km/h for all vehicles whereas the E40 has a constant 120 km/h speed limit. The measurements from cars in the section control on the E17 will therefore be around 90 km/h and shift the histogram. However, this effect is limited to the segments of the section control and not enough to cause the large spike. The E17 is heavily used by industrial traffic as it connects France to the harbour of Antwerp.

To obtain a rough estimate of the fraction of trucks in our datasets, the number of measurements in the range of 77 to 91 km/h can be compared to the total number of measurements. However, part of the measurements in the range will be generated by cars (congestion, highway exits, ...). We estimate this part by linearly interpolating the values inside the range. To eliminate the influence of the varying speed limit (on the E17) and the road works (on the E40), only segments outside these regions are taken into account. This calculation results in fractions of 12% and 21% of trucks for the S_{E40} and S_{E17} respectively. While this is a very rough estimate, it corresponds to the average of 8-15% on the E40 and 21-30% on the E17 reported in [6].

At the high end of the speed distribution, the S_{E40} had 71.4% of measurements below the speed limit and 1.55% above 150 km/h. This is similar to the reported 2.0% of cars above 150 km/h in the BIVV study [8], taking into account the S_{E40} dataset does not differentiate between cars and trucks.

Fig. 2.4 shows the number of measurements for each segment on the E40. More formally, with $M_s = \{m | \mathit{MSID} = \mathit{SID}_s\}$ denoting the set of measurements pertaining to segment s, the figure shows $|M_s|$ with each segment plotted at its relative distance from the start of the dataset (Ternat for S_{E40}).$|M_s|$ varies around 35,000 with 2 notable extremes and small decreases between highway off- and on-ramps as only measurements from people continuing on the highway are taken into account. Between the Wetteren and Merelbeke exits, there are more measurements as it is heavily used by people living in East Flanders to get to Ghent and Antwerp. The number of measurements drops in Merelbeke due to a ring road connecting the E40 with the E17 to Antwerp (see Table 2.1 for highway description). At km 35, the E40 reaches the Zwijnaarde exit which is the preferred exit for people with destination Ghent. The number of measurements reaches a minimum between the off- and on-ramps for Ghent.

Fig. 2.5 depicts $|M_s|$ for S_{E17}. Like before, the minima are attributable to the ring road connecting the E40 and E17 in Destelbergen (km 15) and the highway exit for Ghent (between km 21 and 23) where only data from the people passing the city is taken into account. The points of interest on the E17 are listed in Table 2.2.

Table 2.1: Points of interest on the E40 highway.

Exits	**Distance from start (in km)**
20 Ternat	0
19a Affligem	3.9
19 Aalst	11.3
18 Erpe-Mere	16.5
17 Wetteren	26.4
Merelbeke	31.6
Zwijnaarde	35.0
14 Sint-Denijs-Westrem	38.1
13 Drongen	42.9
12 Nevele	49.8
Traffic cameras	
Merelbeke	30.6
Zwijnaarde	36.1
Section Control	
Start	17.1
End	24.4

Figure 2.4 Distribution of measurements over road segments for the E40 dataset.

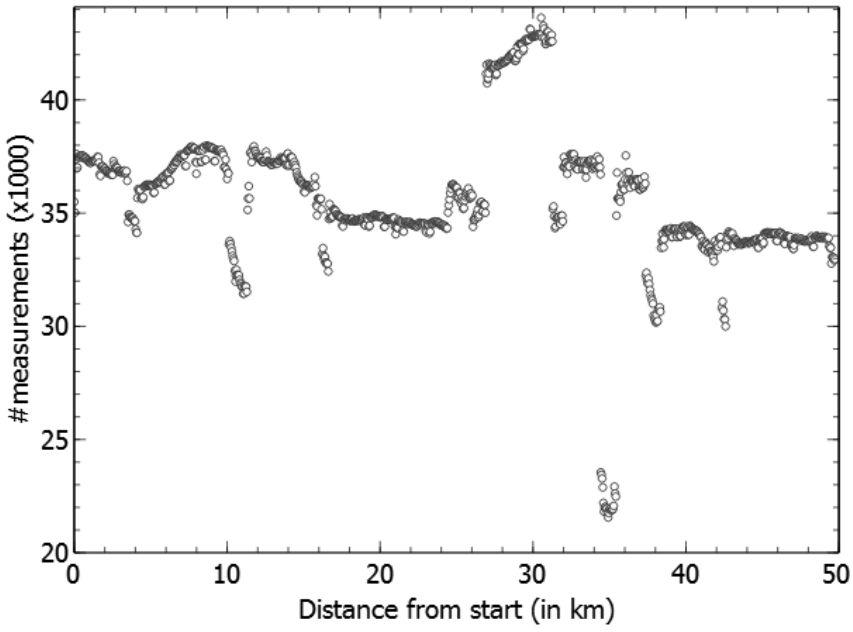


Table 2.2: Points of interest on the E17 highway.

Exits	Distance from start (in km)
12 Lokeren	0
11 Beervelde	10.4
Destelbergen	15.0
10 Gentbrugge	20.1
9 Gent-Centrum	21.9
Zwijnaarde	23.2
8 De Pinte	28.6
7 Deinze	36.8
Traffic camera	
De Pinte	27.5
Section Control	
Start	18.0
End	19.9

Figure 2.5 Distribution of measurements over road segments for the E17 dataset.

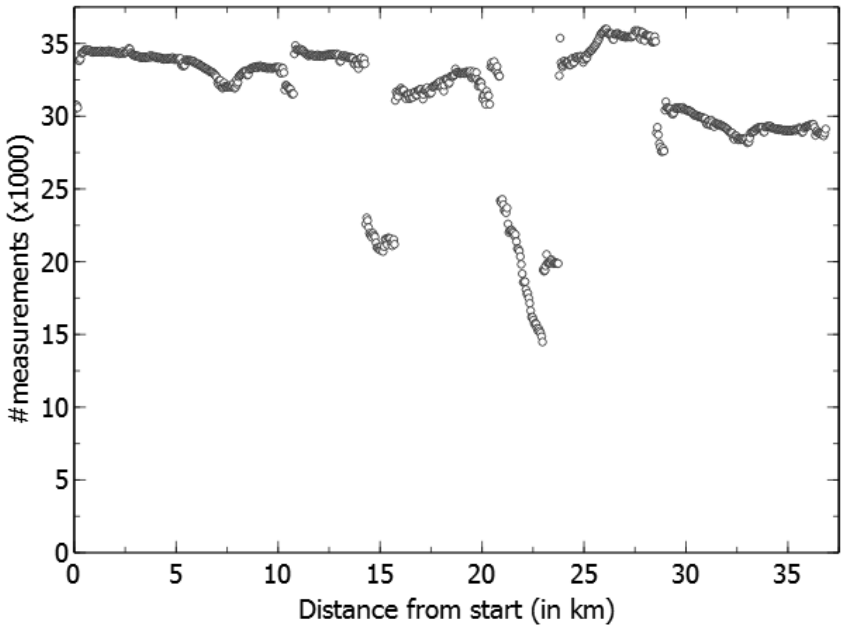
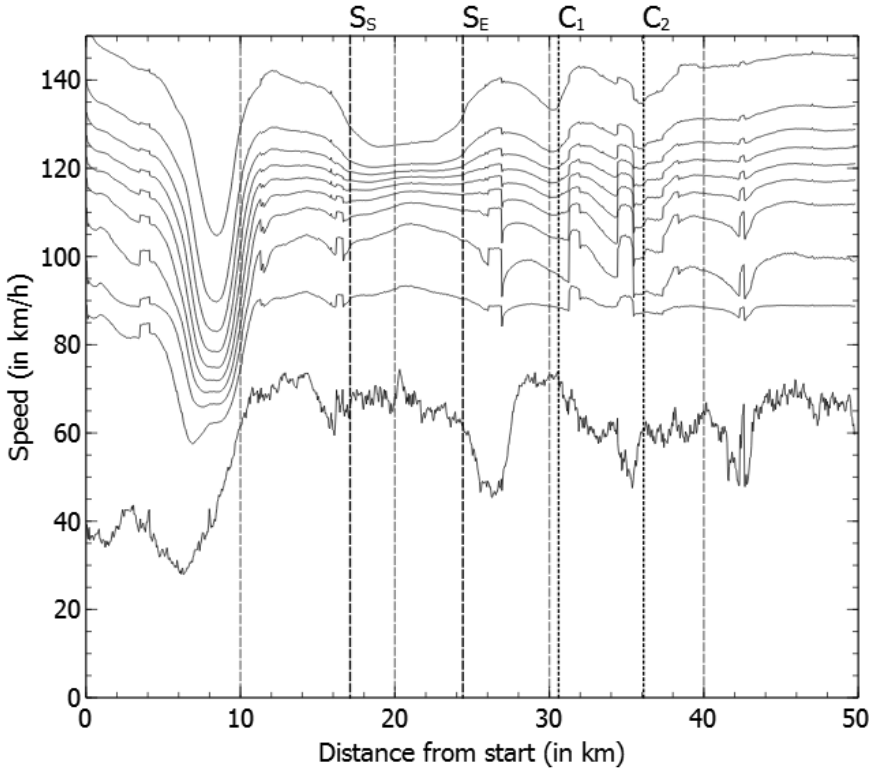


Figure 2.6 Speed profile for the E40 dataset. Each line denotes the average of 10% of the measurements. The top line denotes the average speed of $Q_{10,s}$, the 10% fastest measurements, the 2nd line the average of the next 10%, etc.



2.3.2 Speed distribution

To provide a detailed view on the traffic situation (see [9] for a discussion on the methodological limitations in existing literature), Fig. 2.6 shows the speed distribution for the traffic on the E40. The horizontal axis shows the position of the segment on the E40, starting in Ternat. The measurements per segment are sorted by speed and divided in 10 equally sized groups. More formally, the measurements in each M_s are sorted by decreasing travel time, resulting in $[m_1, m_2, ..., m_{|M_s|}]$, and then divided in 10 groups $Q_{i,s} = \{m_j | (i-1)\frac{|M_s|}{10} < j \leq i\frac{|M_s|}{10}\}, i = 1..10$. Each line in the figure shows the average of one $Q_{i,s}$ with the bottom line denoting the average speed of $Q_{1,s}$ (slowest drivers), the next line the average speed of the next 10%, etc.

From left to right, different parts in the highway can be identified. The large drop in speed in all $Q_{i,s}$ between km 6 and km 10 (between Affligem and Aalst) was caused by large road works during the time of data capturing. The maximum speed limit was lowered to 70 km/h and the number of lanes was reduced. When following the middle lines, highway exits can be identified as sharp discontinuities as people entering and exiting the highway disrupt the normal speed pattern. People getting off and on the highway drive slower, while people in the part between the off- and on-ramp continue at their normal speed.

The bottom line, the average of $Q_{1,s}$, mainly stays below 70 km/h, the minimum speed limit on Belgian highways. Therefore, we can assume that it mainly consists of measurements taken during congestion periods. The minima of this line thus point to the most heavily congested parts of the highway which are the road works and the highway exits at Wetteren (km 25), Zwijnaarde (km 35) and Drongen (km 43).

When looking at the top line, denoting the average of the 10% fastest measurements $Q_{10,s}$, the speed control measures on the highway can be detected. The line shows a significant drop around km 17, indicating the start of the section control on the highway (S_S). After the end of the section control (S_E at km 24.4), the average returns to its previous level with another 2 drops around km 30 and km 36, the locations of 2 fixed speed cameras C_1 and C_2. These fixed cameras are located close to highway exits, which complicates the analysis of the measurements as the effects of the exit and the camera are mixed. However, the results for exits without cameras (e.g. at km 43) indicate that the effect of an exit is small in the top $Q_{i,s}$ while being clearer in the middle $Q_{i,s}$ (shown as discontinuities). In the top $Q_{i,s}$, the average speed is more heavily influenced by exits with cameras than by exits without cameras.

The drop in speed is smaller for the fixed cameras than for the section control. As our system only receives GPS locations every minute, sudden changes in driving behaviour are smoothed by averaging over the time interval. As explained in Section 2.2, the system maps the 1-minute trip to the underlying roads and allocates the time per segment by scaling the time budget of an optimal trip. If a person driving more than the speed limit slows down to avoid being fined by the fixed speed camera and then speeds up again, he no longer follows the free-flow speed distribution. For the fixed speed cameras, the system assumes the constant driving speed of the highway for the entire 1-minute interval and allocates the time budget accordingly. This smoothes the effect of the camera in our measurements as the system will assume that the driver was driving a fixed speed in the entire interval. However, the camera effect is still clearly visible in the top $Q_{i,s}$. For the section controlled part of the highway, the smoothing effect is also present at the start and end of the section, resulting in smooth transitions rather than sudden shifts. However, as the section on the E40 is long enough to collect several GPS

measurements while traversing, the data in the middle of the section is unaffected and shows a steady plateau around 125 km/h for $Q_{10,s}$.

For the lowest $Q_{i,s}$, a slight increase in speed can be found. The 3rd line from the bottom, denoting the average of the measurements in $Q_{3,s}$, rises to its peak value inside the section control, showing a higher speed compared to similar parts of the highway without section control (e.g. around km 45). This increase indicates smooth traffic inside the section control with less congestion and 'slow' drivers speeding up, increasing the throughput of the highway. The effect is also present in $Q_{2,s}$ but weaker, as it still contains a lot of measurements from trucks, which are limited to 90 km/h.

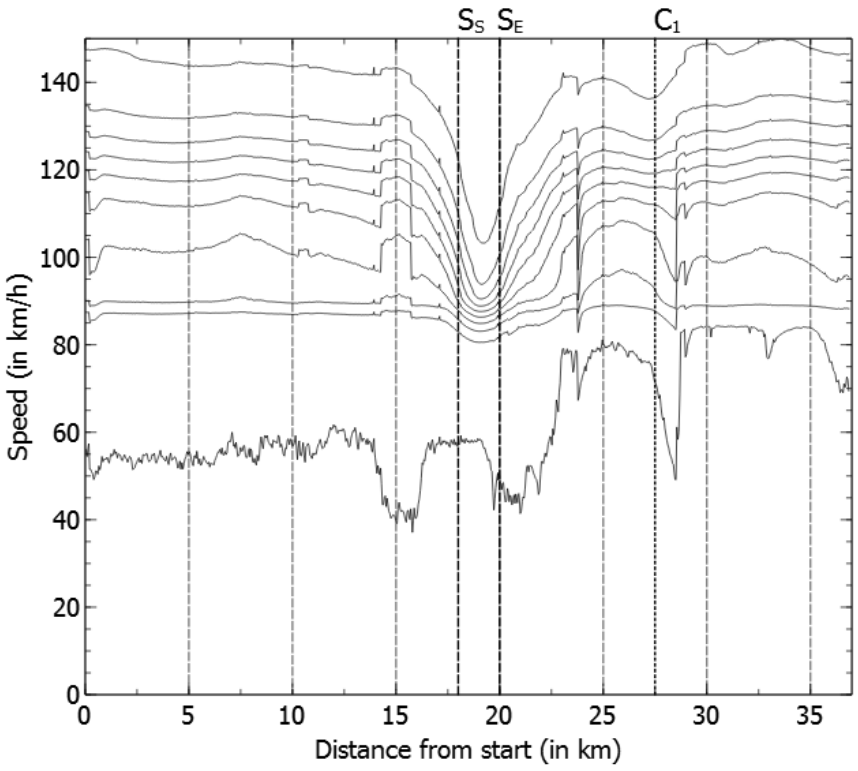
For the second dataset (E17), a similar analysis is made in Fig. 2.7. The bottom line again represents the congested part of the measurements with congested areas for the exits around Ghent. The top line also shows the fixed speed camera C_1 at km 27.5 and the section control, now located between S_S at km 18 and S_E at km 20 with a maximum speed limit of 90 km/h. This causes a significant drop for all $Q_{i,s}$ averages. The smoothing effect (described above) again creates smooth transitions. However, as the section control is shorter, no clear plateau is visible.

2.3.3 Speed dispersion

With respect to safety, it is preferable to have low speed dispersion [10]. The effect of the section control system is most readily visible when comparing the section control part to other (uncontrolled) parts of the highway. More specifically for the E40 highway, the section control (km 17.1-24.4) has similar road properties as the selected reference part of highway further along (around km 45). Both parts are uninterrupted straight roads with long range of sight, no exits or gas stations and no structural design problems. As shown in Fig. 2.6, the speed distribution is much narrower in the section controlled part of the highway than around km 45. To further examine the effect of the section control, Fig. 2.8 plots the difference between the 25th and the 85th percentiles for the E40 (left) and the E17 (right). By taking the difference between the 25th and the 85th percentile, possible influences of accidental traffic jams (during the period of study) are removed from the study. The E40 graph shows a clear decrease in spread inside the section control with 60% of the traffic in a range of 15 km/h while under normal circumstances the range is doubled to 30 km/h. The 25th percentile was taken to avoid the influence of congestion and trucks. As visible in Fig. 2.6, the average of the 10-20% quantile is mostly fixed at 90 km/h, consisting for the bulk part of measurements from trucks.

A similar analysis for the section control on the E17 can be made. However, comparing both sections is complicated as their properties differ significantly. The E17 section has a lower speed limit, causing the speed difference to be inherently

Figure 2.7 Speed profile for the E17 highway. Each line denotes the average of 10% of the measurements. The top line denotes the average speed of the 10% fastest measurements, the 2nd line the average of the next 10%, etc.



smaller. The speed difference clearly drops inside the section control but, as it is much shorter, no clear plateau is reached.

The top line of Fig. 2.9 shows the variance of all measurements inside a segment and more notably, a decrease in variance inside the section control on the E40. However, our dataset contains both truck and car data which have separate speed limits and distributions. To remove the influence of trucks, the bottom line shows the variance of the measurements with a minimum speed of 95 km/h. Note that cutting off at 95 km/h also removes all measurements taken during heavy congestion periods so the resulting line is only relevant to non-congested car traffic. This results in a more appropriate disaggregate view on the speeding behaviour as spikes, caused by local congestion, are removed. The variance decrease is again clearly visible, showing the effect of the section control on the traffic without the disrupting truck influence.

2.3.4 Variation throughout the day

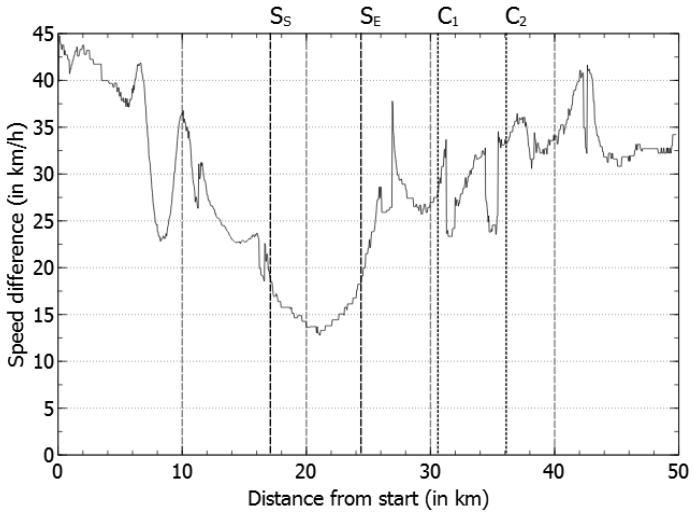
The variance presented in Fig. 2.9 is calculated for each segment throughout the day. However, congestion traffic is strongly influenced by the time of day. To present a more disaggregated view of the data, Fig. 2.10 shows the average speed of the top 10% as a function of the location and the time of day.

As mentioned above, the average speed of the top 10% is closely related to the speed control measures taken on the highway. Inside the section control, the average speed remains nearly constant throughout the day. As the speeds are higher on other but similar parts of the highway, the constant speed can be attributed to the section control which strongly enforces an upper limit on the speed.

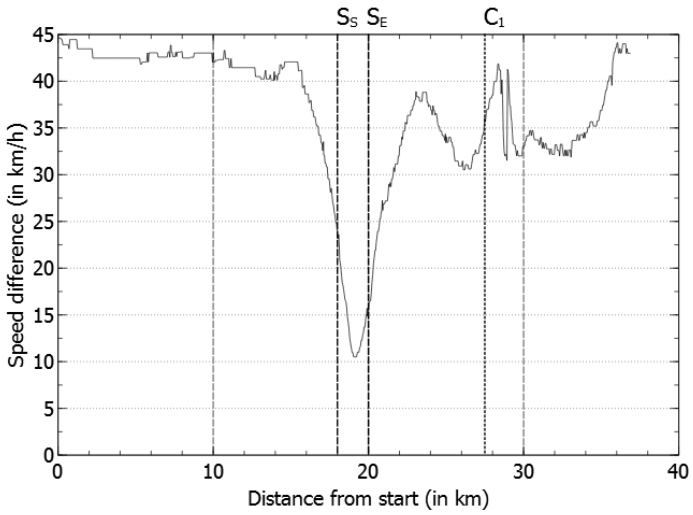
Looking along the location axis, the effect of the fixed speed cameras can again be seen despite of the smoothing effect. Two small dips indicate cars slowing down upon seeing the cameras at those locations but speeding up again afterwards. This behaviour is however not desirable as sudden changes, e.g. cars abruptly braking, can be very dangerous. This is supported by the correspondent's feedback and driver classification in [11] in which some drivers comment on this dangerous behaviour introduced by the cameras. While a slowdown and speedup phase are also present near the start and end of the section control, the average speed control system of 2 camera locations is much more effective in reducing the speed between them than the 2 fixed cameras.

Looking at the maximum speed variation throughout the day, a small increase in maximum speed can be found during the night. While this could be attributed to people possibly driving faster at night, the increase is also due to the lower traffic intensity at night. During the day, the normal (free-flow) traffic volume causes vehicles to interact more frequently compared to during the night. This reduces the possibility of speeding during the day. Furthermore, the occurrence of traffic

Figure 2.8 Speed dispersion. The lines denote the difference in speed between the 25th and 85th percentile of the measurements in the corresponding road segment. Fig. a shows the difference for the E40 highway while Fig. b pertains to the E17 highway.

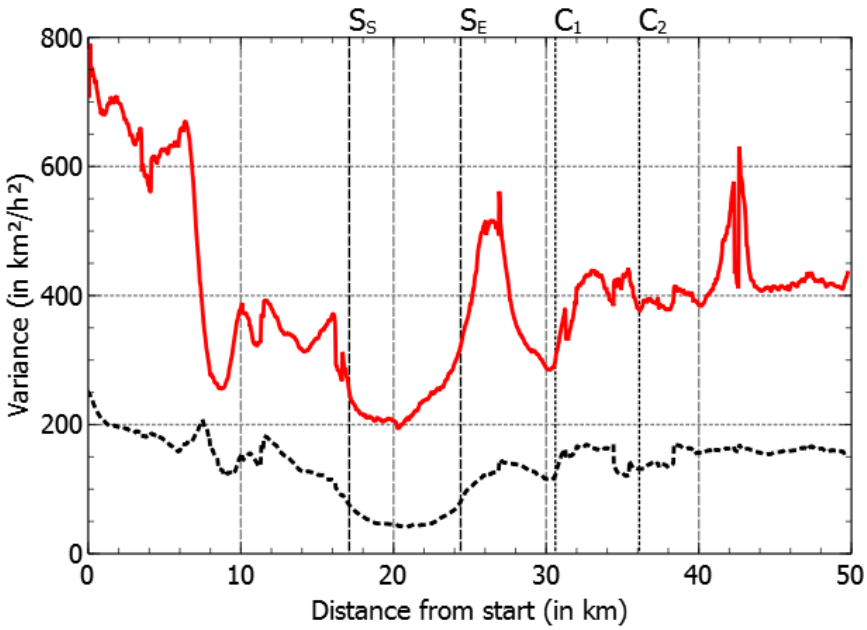


(a)



(b)

Figure 2.9 Speed variance E40. The red line shows the variance of the speed measurements per road segment on the E40. The black dotted line also shows the variance of the measurements in a segment but only for measurements with a minimum speed of 95 km/h. The variance drop around km 20 is located inside the section control.



congestion during the day pushes down the maximum speeds as measurements taken during congestion lower the overall speed and speed distribution.

Note that the dataset used in the evaluation was obtained by monitoring individual vehicles, both trucks and cars. While the measurements are not denoted by the type of vehicle, the percentage of trucks can be roughly estimated by using the number of measurements with typical truck speeds as discussed in Section 2.3.1. The estimated average percentage of trucks during the night (35%) is higher than the estimated percentage during the day (10%) due to fewer active cars and a higher relative truck density on the highway. While this lowers the average speed of the measurements in the timeslot (when aggregating all measurements), the top 10% (used in the analysis for this reason) remains largely unaffected as no truck measurements are in that subset.

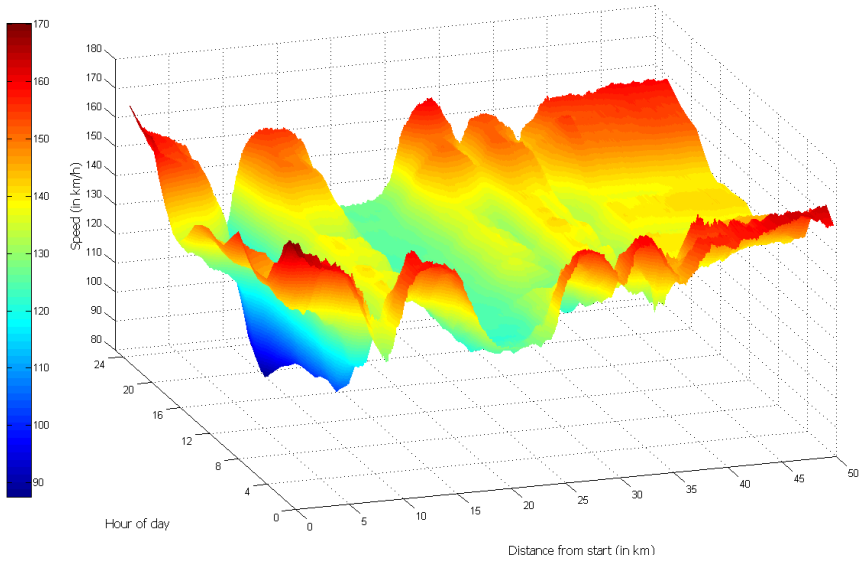
2.4 Conclusions

In this paper, we evaluated the effect of section control camera systems on drivers' behaviour using Floating Car Data (FCD). FCD was collected on 2 important Belgian highways using an operational FCD system of over 200,000 vehicles and converted to time measurements using an improved map matching algorithm. Results show that the section control system reduces the traffic speed over its entire length, effectively enforcing the maximum speed limit. A small speedup was found for vehicles driving under the speed limit. Traffic speed also became more homogeneous, with the speed difference between the 25th and 85th percentile being reduced to 15 km/h inside the monitored section compared to 30 km/h on similar parts of highway. While fixed traffic cameras also reduce speeds, their effect is very localised as traffic speeds up again after having passed the camera positions, resulting in varying speed profiles. With respect to safety, section control is more effective at creating steady, harmonious traffic within the imposed speed limits.

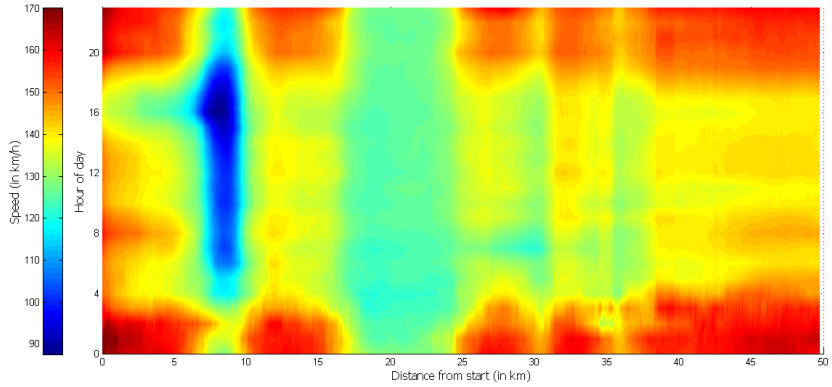
2.5 Future work

As FCD provides detailed information on traffic conditions, it can be used to analyse various traffic situations and scenarios. In this paper, the speed controlled section was analysed on its effectiveness on the driving behaviour. However, this is only one example of speed limit enforcement. Depending on the situation, different policies are adopted to enforce driver compliance and improve safety. During the road works present in the E40 dataset, the 120 km/h speed limit was lowered to 70 km/h without active enforcement. While the effect of this measure is already visible in Fig. 2.6, a more detailed analysis and comparison with other road works will yield a better view on the effect on drivers' behaviour. Moreover, by mon-

Figure 2.10 Top speed quantile. Fig. a shows the evolution of average speed of top 10% fastest measurements throughout the day for the E40 dataset. The section control can be seen around km 20 as the light-blue trench while the sharp dip around km 9 is the result of road works. A top view of this plot is provided in Fig. b.



(a)



(b)

itoring similar situations on an international scale, a comparison could be made between driving behaviour in different countries.

Aside from analysing traffic situations, FCD also provides an interesting benchmark to compare and study other traffic detecting methods like the roadside sensors. The overall traffic view provides a full traffic snapshot that can be matched to the individual sensor readings, allowing the testing and even enrichment of their data.

Acknowledgements

Mario Vanlommel is supported by a PhD Baekeland mandate by the Flemish agency for Innovation by Science and Technology (IWT). Maarten Houbraken is supported by a PhD fellowship grant by the Research Foundation-Flanders (FWO-Vlaanderen). The authors would also like to thank Be-Mobile for their cooperation and supplying the FCD data and iMinds for their support.

References

[1] R. L. Cheu, C. Xie, and D. Lee. *Probe vehicle population and sample size for arterial speed estimation*. Computer-Aided Civil and Infrastructure Engineering, 17(1):53–60, 2002.

[2] J. Hong, X. Zhang, and Z. Wei. *Spatial and temporal analysis of probe vehicle-based sampling for real-time traffic information system*. In IEEE Intelligent Vehicles Symposium, pages 1234–1239, 2007.

[3] H. Bar-Gera. *Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: A case study from Israel*. Transportation Research Part C: Emerging Technologies, 15(6):380–391, dec 2007.

[4] C. De Fabritiis, R. Ragona, and G. Valenti. *Traffic estimation and prediction based on real time floating car data*. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, (NOVEMBER 2008):197–203, 2008.

[5] S. Demeyer, P. Audenaert, M. Pickavet, and P. Demeester. *Dynamic and stochastic routing for multimodal transportation systems*. IET Intelligent Transport Systems, 2013.

[6] Verkeerscentrum Vlaanderen. *Verkeersindicatoren hoofdwegennet Vlaanderen 2012*. Technical report, 2012.

[7] European Conference of Ministers of Transport. *Speed management*. Technical report, 2006.

[8] Belgian Institute for Traffic Safety. *Nationale gedragsmeting Snelheid op autosnelwegen - 2011*. Technical report, 2011.

[9] D. W. Soole, B. C. Watson, and J. J. Fleiter. *Effects of average speed enforcement on speed compliance and crashes: A review of the literature*. Accident Analysis and Prevention, 54(1):46–56, 2013.

[10] L. Aarts and I. van Schagen. *Driving speed and the risk of road crashes: a review.* Accident Analysis and Prevention, 38(2):215–224, mar 2006.

[11] K. M. Blincoe, A. P. Jones, V. Sauerzapf, and R. Haynes. *Speeding drivers' attitudes and perceptions of speed cameras in rural England.* Accident Analysis and Prevention, 38(2):371–378, mar 2006.

[12] B. Hellinga, P. Izadpanah, H. Takada, and L. Fu. *Decomposing travel times measured by probe-based traffic monitoring systems to individual road segments*. Transportation Research Part C: Emerging Technologies, 16(6):768–782, dec 2008.

[13] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. *Current map-matching algorithms for transport applications: State-of-the art and future research directions*. Transportation Research Part C: Emerging Technologies, 15(5):312–328, oct 2007.

[14] E. Dijkstra. *A note on two problems in connexion with graphs*. Numerische mathematik, 1(1):269–271, 1959.

"Curing congestion by adding more lanes is like curing obesity by buying bigger pants."

— Anonymous

3

Modelling Traffic Congestion Evolution through Speed Profile Generation using Floating Car Data

The previous chapter showed FCD being used for local road analysis. However, given the inherent spatially distributed nature of FCD, there is nothing holding it back to tackle larger areas. This chapter further explores the power of FCD for large scale network analysis by first converting it to intuitive speed profiles and then using it in a countrywide congestion analysis. FCD was collected during 4 months in the Netherlands and then processed using a congestion index to detect structural congestion hotspots, demonstrating the suitability of FCD.

⋆ ⋆ ⋆

Maarten Houbraken, Steven Logghe, Pieter Audenaert, Didier Colle, Mario Pickavet

Submitted to 25th ITS World Congress, Copenhagen, Denmark, 17-21 September 2018.

Abstract With more and more location-aware and connected devices, individual probe vehicle tracking has become a valuable source of information for traffic monitoring and analysis. Floating Car Data (FCD), generated by the probe vehicles, provides data samples for the actual travel times and speeds on wide area networks for long periods of time. This information can be used to detect and tackle congestion hotspots in the network. The focus of this paper is the conversion of historic FCD to intuitive time dependent speed profiles for individual road links, suited for routing and congestion analysis. Among other speed profile models, we present the Trapezoidal Rush Hour Congestion (TRHC) model aimed at capturing speed evolution in traffic-oriented model parameters. The models are evaluated using 4 months of data collected by a live countrywide FCD system in the Netherlands and best performance is reached by the TRHC model focusing on rush hour dynamics. The TRHC model parameters can be directly used for network analysis, which is illustrated by analysing the Dutch road network for structural problem areas, showing the suitability of FCD for wide area network analysis.

3.1 Introduction

In the past decade, Floating Car Data (FCD) has emerged as a valuable source of traffic data. FCD consists of data samples generated by vehicles transmitting their position (and other relevant data) to an external server for further processing. By carefully analysing this data, both real-time and historic traffic analysis can be done. This paper aims at presenting how FCD can be used to model speed evolution throughout the day and determine congestion areas on a countrywide network.

Over the past years, FCD has changed significantly. In the 90's, emerging data networks restricted early FCD platforms to low data usage (see [1]). With better communication networks and the ubiquity of GPS, FCD has grown to high data volumes with samples being transmitted every 1-60 seconds. High-frequency polling schemes allow for the most up-to-date information and simple route matching compared to low-frequency polling systems requiring more complex route estimation algorithms (see [2–4]). However, high-frequency polling does have a higher energy consumption and data throughput (see [5]), requiring more resources of the on-board device as well as more central processing power.

Looking at the applications of FCD systems, most research is focused on traffic state categorisation and travel time (distribution) estimation. Traffic state categorisation, classifying live traffic in congested or free-flow states, is most useful in traffic management applications where congestion detection often is more important than speed modelling. Congestion patterns are studied in [6] using 2 months of 1-minute aggregated FCD on a 4.5 km study corridor and show that FCD can detect structural congestion bottlenecks. Regarding real-time traffic management, [7, 8]

present a variable speed limit system warning for end-of-queue collisions working solely on raw, high-frequency FCD, showing how FCD presents a valid alternative to inductive loop data. In [9], a traffic state estimation algorithm is presented using different congestion phases to categorise samples before using them in state estimation. For more traffic state estimation methods, we refer to [10] providing an extensive survey on highway traffic state estimation methods, as this paper is not focused on state classification.

With regards to travel time distributions, [11] derives travel time distributions on 27 selected routes from an FCD platform of 1,500 taxis in Stockholm. While route-based models more accurately capture the experienced travel times of the drivers, link-based modelling is used in this paper as for large networks, the number of routes to be monitored is intractable. Travel time distributions are also focused on in [12] by stochastically modelling intersection turning times.

This paper focuses on deriving simple and intuitive speed profiles for modelling speed evolution throughout the day from large-scale countrywide FCD systems. Most research dealing with FCD speed profiles model the speed evolution by calculating a speed estimate for a number of timeslots during the day. For example, [13] uses 24x7 uniform timeslots to process their FCD. FCD samples are averaged per bin to obtain an average travel time for that timeslot. The resulting speed profile is then used in a clustering algorithm to further group segments by their general profile evolution. Similarly, [14] uses truck FCD to estimate truck speed profiles by aggregating in bins. After clustering the segments based on their found speed profiles to obtain a reference, they derive anomalous traffic events from the speed profiles. While averaging the values in each bin is simple, intuitive and produces acceptable results, it does require a lot of data to be stored per link (1 value for every timeslot and every link), which [13, 14] reduce by clustering, losing some accuracy.

In this paper, the aim is to reduce the size of the data required to store the speed evolution while still maintaining intuitive modelling and accuracy. This is important as it allows easier incorporation in existing applications and a wider area of applicability.

The resulting speed profiles can be used in various further applications. With static routing algorithms easily handling countrywide and even continental route queries [15], speed profiles for large networks are needed for deploying the time-dependent variant [16]. Aside from routing, another important application is general road network analysis to support policy makers and road network operators. FCD, with its spatially distributed coverage, allows for much broader network coverage compared to loop data. Two examples of specific road analysis can be found in [17] (focusing on section control speed enforcement using travel time distributions from FCD) and [18] (focuses on a 20 mile highway to derive congestion space-time areas from 1 year of loop detector data). However, analysis on a

countrywide network scale is still limited as field data is difficult to obtain [19]. In [20], the European Commission reports on the congestion in Europe, starting from averages speed profiles generated from commercial FCD from 2008-2009, again illustrating the need for speed profiles.

The remainder of this paper is outlined as follows. First, the floating car data sample generation from raw positioning data is discussed in Section 3.2. Given the data characteristics, Section 3.3 presents four different models used to convert the historic data samples to speed functions for each link in the study area. Section 3.4 further shows the application of the models in the Dutch case study under evaluation. Section 3.5 concludes the paper.

3.2 FCD generation

Different FCD generation and processing schemes exist to convert raw positioning data to travel times and/or speeds on underlying road segments. As this influences the properties of the data, this section discusses the applied approach of the FCD system in this paper.

3.2.1 Converting vehicle positions to experienced travel times

The FCD is generated on a detailed digital representation of the road map. This map is composed of unidirectional road segments of at most 50 metres. With this granularity, it is possible to assume uniform traffic on a single road segment [21].

The raw vehicle positions are used as the input to the system. A set of pre-processing filtering rules is first applied to omit invalid data. Based on the successive points of an individual vehicle, the driving path of the vehicle is then reconstructed on the digital map. The path reconstruction first maps all individual raw positions to the map by selecting a set of candidate segments close to the input coordinates, similarly as described in [11]. The input positions are then connected to the graph by projecting them to the underlying segments.

Next, the potential driving paths between all combinations of these candidate segments are calculated. The paths are calculated based on the underlying graph representation using a shortest path algorithm like Dijkstra [22] assuming the vehicle is driving the free-flow speed. The shortest path from the start position to the mapped successive positions leads to the selection of the chosen candidate segments for the vehicle positions. The set of raw input coordinates is thus converted to a driven route of the vehicle. Note that if no possible route to an intermediate input position is found within realistic speed restrictions, the route of the vehicle is terminated and a new route is started for the remaining points.

Based on the map matched driving path and the timestamps of the vehicle positions, the actual travel times on each segment of the driving path are calculated.

Using the free-flow speeds from the road map, first the free-flow travel time between 2 successive points is calculated, along with the free-flow travel times on each individual segment on the route. Due to traffic congestion, this time can differ from the measured time (= difference between timestamps of received points). A scaling factor between the free-flow estimate and the measured time can then be calculated (as the ratio between them) and applied to the free-flow travel times on the individual segments. This results in the experienced travel times for each individual segment. This calculation thus converts 2 successive timestamped positions to multiple experienced FCD travel times, one for each segment along the route of the vehicle.

3.2.2 Deriving link travel times

Once the individual vehicle trips are processed to samples for each segment, aggregation is applied to extract the actual travel time on the links. This aggregation is necessary as FCD systems only track a small subset of all traffic and outliers in the data could potentially skew the resulting data. For each segment, the FCD samples are averaged for each 5 minutes of the day. This temporal aggregation window is a trade-off between granularity of the resulting state and data availability. A smaller time window reduces statistical averaging (thereby reducing outlier smoothing) while for larger windows, evolving traffic conditions risk being mixed.

The final step in the data pre-processing consists of combining the aggregated travel time of the segments to longer links. While segments are defined as (at most) 50 m long, links are typically much longer and run between 2 intersections, highway ramps or other splitting points on the road. More formally, a link is a variable length unidirectional stretch of road on which traffic can only enter and exit through the start and end of the link.

While spatially aggregating segment travel times to links risks merging different traffic conditions along the link, it does not affect the overall experienced travel time on the links. The essential benefit of the segmented approach is to group traffic measurements to areas of uniform traffic. This importance is most clearly shown in periods of congestion onset. While parts of a link might be congested, other parts (upstream) might still be unaffected. In that case, the generated FCD for those road parts differs in both value (e.g. speed or intensity) as number of probes (as more distance on the link is covered by the free-flow vehicles). To avoid unfair averaging of the different data, the segments are used to group the data, allowing more balanced sample aggregation.

The segment aggregation consists of summing the segment travel times of the segments on the link. This requires data on all the underlying segments. However, an individual vehicle might not completely cover a long or congested link between 2 successive positions. Therefore, the data for the individual vehicle might not yet

be completely available. This is countered by the temporal aggregation window (taking into account data from the last few minutes) along with the data from other vehicles on the link. Note that this approach of summing travel times of segments for a specific time results in instantaneous travel times on links, in contrast to experienced travel times which represent the time it would take to traverse the link when entering it at that specific instant.

3.3 Modelling speed evolution throughout the day

The FCD generation from above results in link travel times for every time of day for which a vehicle was tracked on the link and with a granularity determined by the temporal aggregation window. Applying this approach for extended periods of time allows to study typical travel time evolution throughout the day. Travel time evolution throughout the day is very important in, for example, accurate route planning engines which need to predict travel times. Another interesting application is road network analysis (see Section 3.4.4), for which it allows to identify structural hotspots and problem areas in the transportation network.

Note that the focus here is on modelling individual link profiles. When optimising for entire networks, more cross-dataset compression can be used to reduce the number of variables per link e.g. clustering neighbouring/connected roads having similar characteristics. Data compression techniques could be used to reduce the memory size needed for storing both profiles. Furthermore, links without much variation in travel time could be modelled separately and more compactly e.g. by assuming a fixed travel time or speed, thereby omitting other remaining parameters. Even more intricate models can be used to model all links at once e.g. by defining a set of base profiles and expressing all individual link profiles as a combination of the base profiles. However, the intent here is to model individual links from FCD with intuitive and meaningful parameters to be easily used in further applications.

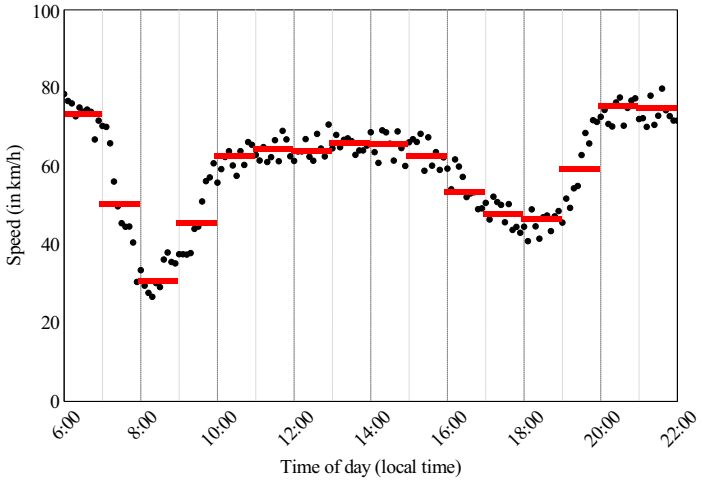
This section presents 4 different approaches for modelling and extracting typical link profiles from FCD. As FCD is inherently sampled data, the models need to account for noise and missing data while still extracting the general trend of the traffic evolution. Furthermore, the models were designed to offer intuitive parameters to facilitate use in further applications.

3.3.1 Bin model

The Bin model consists of dividing the day in n equal sized periods (= n bins) and storing one speed value for each period. The value for each period is calculated as the average of all available data samples in the interval. This minimises the Mean Squared Error (MSE) for the samples in each individual bin. This ap-

proach is both extensible, as n can be varied, and intuitive, as the exported values directly represent speed values. This allows easy profile querying and comprehension. However, the biggest disadvantage is that it does not capture the rush hour dynamics as it ignores the start/end of the rush hour. This results in a step approximation during the onset and dissipation of the rush hours, which are often the most important parts of the speed profile. Another disadvantage of the model is that the profile is discontinuous, as the resulting speed profile is only piecewise continuous (see Fig. 3.1). The sharp changes in speed around the timeslot edges need to be accounted for (e.g. by interpolating) in the applications building on this data to avoid erratic behaviour (e.g. large sudden differences in travel time estimates, unstable time windows in delivery planning applications, ...).

Figure 3.1 Bin model example. The individual speed samples (black dots) are averaged to provide one speed estimate for every hour (shown by the red lines).



3.3.2 Fourier series

The second model presented here is a Fourier series approximation of the speed profile. In this model, the measured speed values for a day are fitted to a general Fourier series of the form in Eq. (3.1).

$$speed(t) = a_0 + \sum_{i=1..k} (a_i \cdot cos(i \cdot \frac{2\pi t}{t_e - t_m}) + b_i \cdot sin(i \cdot \frac{2\pi t}{t_e - t_m})) \qquad (3.1)$$

This function thus consists of $n = 2 \cdot k + 1$ variables, with a_i and b_i representing the k Fourier coefficients defining the speeds between t_m and t_e, denoting the

morning start time and evening end time respectively. The variables are calculated by computing a least-square fit of the input speed samples.

While the Fourier series approximation is more complex than the previous model, it is well-studied and widely available in software libraries. The resulting variables are now less intuitive as they only represent coefficients for sinusoidal base functions, with the exception of the a_0 variable which represents the base speed for the day. To extract the speed for a specific time, all coefficients now need to be used as well. However, the model is extensible, as the number of coefficients k can be increased easily.

3.3.3 Wavelet model

The third model used in this paper is based on wavelet transformation. As wavelets transformations have been extensively studied before, only a short introduction is given here. More detailed explanations and technical descriptions can be found in literature [23].

The Fourier model above aims at capturing the information in the input data by using sinusoidal base functions of specific frequencies. To obtain the coefficients for these base functions, the input data first undergoes Fourier transformation. This results in data values representing a frequency spectrum of the input signal, of which a number of coefficients is retained for the model. The main idea behind wavelet modelling is to capture the information in the input data in the time domain as well as in the frequency domain, in contrast to the Fourier model. This allows to better capture variation in specific parts of the input, as local variations in the time domain are not completely smeared across all coefficients in the frequency domain.

To obtain the model values, the input speed values are first averaged per minute and then sequentially passed through a set of filter banks. A filter bank consists of a low-pass and a high-pass filter and captures the input information in the frequency domain corresponding to the type of wavelet used. The type of wavelet base functions used in this paper are the Haar-wavelets as these provided the best results. The result of one pass through a filter bank is a series of low-frequency values and a series of high-frequency values. Both series are downsampled and the series of low-frequency values is passed to the next filter bank. By iteratively passing the low-frequency values, the end result of the wavelet transformation is a hierarchically structured series of coefficients. The coefficients on the highest level then correspond to the lowest frequencies and span the widest time range after reconstruction. The input signal can be recovered through the inverse filter bank operation on the values.

The Wavelet model used here thus computes the wavelet coefficients and retains n values as the model parameters. These n values, supplemented with zeros,

can then be used to reconstruct the speed profile. As n can be varied easily, the model allows for extensibility similar to the Fourier model. In the model here, the coefficients on the highest levels are retained. An alternative choice would be to retain the largest n values, as these indicate high energy and variation for that specific time-frequency position. However, with a limited number of retained coefficients, the larger wavelets (coupled to the coefficients of the highest level) gave more gain over the entire profile than the smaller wavelets (which are more focused on modelling the function locally).

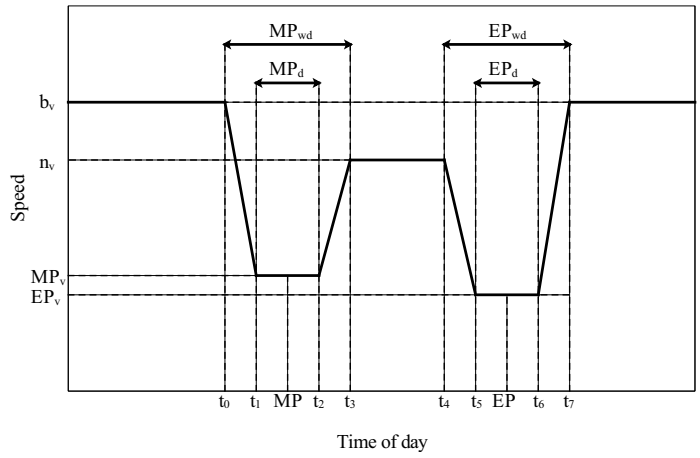
3.3.4 Trapezoidal Rush Hour Congestion model

3.3.4.1 Model presentation

This last model is more elaborate than the previously models as it was specifically designed to capture rush hour dynamics with intuitive parameters. As it explicitly considers rush hour periods, it is more tuned to the field of transportation than the previous models which are more general data fitting models.

The proposed general speed profile is shown in Fig. 3.2 and consists of 2 rush hour periods (morning and evening) combined with baseline speeds during the night and noon period. As the rush hours are modelled as trapezoidal functions, the model is named the Trapezoidal Rush Hour Congestion model (TRHC).

Figure 3.2 The general speed profile of the TRHC model.



The base speed b_v indicates the speed when no traffic is present on the road. Similarly, the noon speed n_v indicates the speed when a normal amount of traffic is present. The rush hour peaks are defined by the peak time (resp. morning peak *MP* and evening peak *EP*), the intensity of the peak (resp. morning peak speed MP_v and evening peak speed EP_v), the duration of the peak period (resp. morning

peak duration MP_d and evening peak duration EP_d) and the duration of the entire peak period, including the onset and dissipation of the traffic congestion (resp. morning peak wide duration MP_{wd} and evening peak wide duration EP_{wd}). The t_i represent the times on which the profile switches to a different phase and can easily be derived from the function parameters.

The biggest advantage of this model is that all variables directly correspond to intuitive traffic concepts. This enables traffic analysis to be performed directly on the variables of the model (morning rush hour speed, start of rush hour,...), in contrast to the previous models in which those values would need to be extracted by re-evaluating the entire function.

The proposed profile consists of 10 variables. While the previous models are more extensible, this model is less flexible. It allows for further simplification by for example assuming block peaks instead of trapezoidal functions, effectively removing 2 out of 10 parameters, yielding only 8 variables to be fitted. However, this is not desirable as the speed profile then becomes discontinuous around the slopes. It also reduces the modelling power of the proposed profile. A second simplification consists of dropping the noon speed and taking the base speed outside of the rush hour peaks. This would yield a TRHC model with 9 variables. A third simplification consists of only having one peak, thereby reducing the model to 2 levels with continuous slope, requiring only 5 variables. Combining these simplifications yield an additional 2 TRHC models: only one rush hour period with discontinuous slopes (4 variables) and a dual rush hour model with no noon value and discontinuous slopes (7 variables).

3.3.4.2 Fitting the model

As the TRHC model, in contrast to the previous models, was developed specifically for capturing the rush hour periods in traffic in this paper, the fitting procedure needs to be presented here. In essence, the different variables are initially set and then iteratively optimised through a set of basic operations (6 steps) that aim to minimise the MSE between the estimate and the input data.

Initialisation Fitting starts with initialising the individual parameters to a set of standard values. The initial values for *MP* and *EP* are set to 8:00 and 17:30 local time respectively. The MP_d and EP_d are initialised to 2 hours while MP_{wd} and EP_{wd} are set to 4 hours. The other parameters b_v, n_v, MP_v and EP_v are set to 0 as these will be updated in what follows before being used elsewhere.

The fitting then starts to iteratively estimate and refine the parameters. Note that only the procedure for the morning peak will be mentioned, the evening peak is analogous.

Step 1: Estimating noon and base speed First, the noon speed n_v is calculated by averaging the speeds in the current noon period. The noon period runs from the end of the morning peak (t_3) to the start of the evening peak (t_4). Similarly, the base speed b_v is calculated based on the data before the morning peak and after the evening peak.

Step 2: Estimating morning and evening peak times In the second step, the peak time values *MP* and *EP* are discretely optimised. This is done by calculating a moving average over the samples in the peak period (between t_0 and t_3). The moving average is used to detect the real dips in the speed samples, as normal averaging could be offset by outliers while the overall peak value is of interest. The time with the lowest average speed is taken to be the peak time. The calculation is done using Eq. (3.2), with $S_{t_0-t_3}$ representing the set of data samples between t_0 and t_3 and v_i the speed of an individual sample.

$$\mathit{MP} = \underset{t_0<s<t_3}{\arg\min} \frac{\sum\limits_{i \in S_{t_0-t_3}} v_i \cdot w(|t_i - s|)}{\sum\limits_{i \in S_{t_0-t_3}} w(|t_i - s|)} \tag{3.2}$$

A moving average with a time window of 1 hour is used to avoid outlier and noise influence. This translates to a $w(t)$ weighting function with only values for $t \leq 30$ min. The complete weight function was defined as in Eq. (3.3).

$$w(t) = \begin{cases} 1 & t \leq 10 \text{ min} \\ 0.5 & 10 \text{ min} < t \leq 20 \text{ min} \\ 0.3 & 20 \text{ min} < t \leq 30 \text{ min} \end{cases} \tag{3.3}$$

Step 3: Estimating morning and evening peak speeds The third step readjusts the speeds during the peak periods by averaging the data sample speeds during the peak period (between t_1 and t_2 resp. t_5 and t_6). This ensures that the MP_v and EP_v correspond to the new *MP* and *EP* times from step 2 and minimise MSE in the new peak interval.

Step 4: Optimising rush hour start and end The following 3 steps optimise the transition between normal traffic and rush hour by minimising the MSE. Note that by averaging in steps 1 and 3, the MSE for the corresponding sections are also optimised. A visualisation of the optimisation is given in Fig. 3.3.

The first explicit MSE minimisation is done on the left flank of the peak by shifting the left side of the peak in time. This entails varying t_0 and t_1 discretely (steps of 5 min) between 6:00 and *MP* simultaneously and minimising the MSE. To avoid explicitly calculating MSE values for every possible start time t_0, the MSE can be calculated incrementally in order (e.g. from earliest to latest). This

allows for calculating the MSE in a single pass over the measurement values. Note that by varying t_0 and t_1 simultaneously, the slope of the peak remains unaltered. This shifting of the slope is repeated for the right side of the peak. With both sides shifted, the *MP*, MP_d and MP_{wd} are recalculated (to ensure *MP* is again in the middle of the function).

Step 5: Optimising rush hour duration The next MSE optimisation step is analogous and consists of optimising only MP_d in terms of MSE. This is done by varying t_1 and t_2 around *MP*. To optimise calculation times, the MSE can again be calculated incrementally. This operation can be seen as finding the optimum function somewhere between a rectangle (with the new t_1 near t_0) and a triangle function (with the new t_1 near *MP*). The optimisation is done at both sides of the peak simultaneously, ensuring *MP* remains unaltered.

Step 6: Optimising rush hour wide duration The last optimisation step is similar to the previous but optimises MP_{wd}. The same method as in step 5 is followed but t_0 and t_3 are varied while keeping the rest fixed. Steps 1-6 are repeated for each link under study as long as the MSE improves. The result is a fully defined general speed profile.

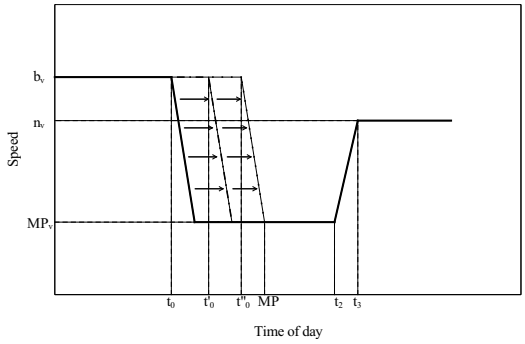
3.4 Application: Dutch road network

3.4.1 FCD generation

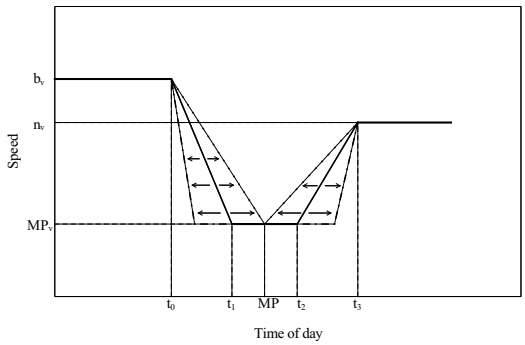
To evaluate the models presented above, link profiles were generated from raw positioning data collected by a live FCD system in the Netherlands from January to April 2015. Holidays were removed from the data, as the traffic characteristics on holidays are typically different from normal days. For the entire period, the system covered an estimated 3% of all active vehicles (percentage derived from comparing FCD sample counts with installed loop detectors), resulting in 8 billion raw GPS positions. The monitored fleet of probe vehicles consists of both cars and trucks/delivery vans.

An estimated split is presented in Fig. 3.4 with the bottom red circle markers denoting the number of truck samples for that specific time and the top blue star markers denoting the number of car samples. The ratio of car position samples to the total number of samples varies between 60% (during the night) and 85% (evening rush hour). According to government statistics [24], cars take up 78% of the total number of kilometres driven while trucks and delivery vans only take 17%, resulting in a split of 82% car versus 18% truck/van samples to be expected. Taking into account that trucks and delivery vehicles are more likely to be active throughout the day, in contrast to typical peaked commuting behaviour of cars,

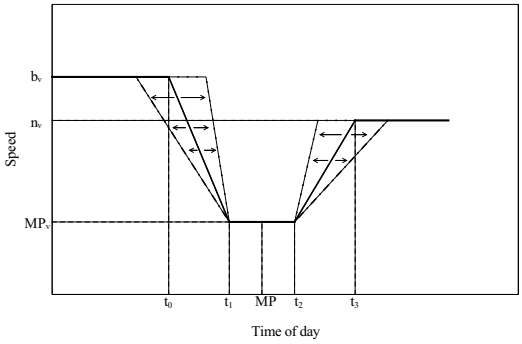
Figure 3.3 Illustration of step 4 through 6 with the individual operations on the flanks of the peak.



(a) Step 4: shifting the left flank

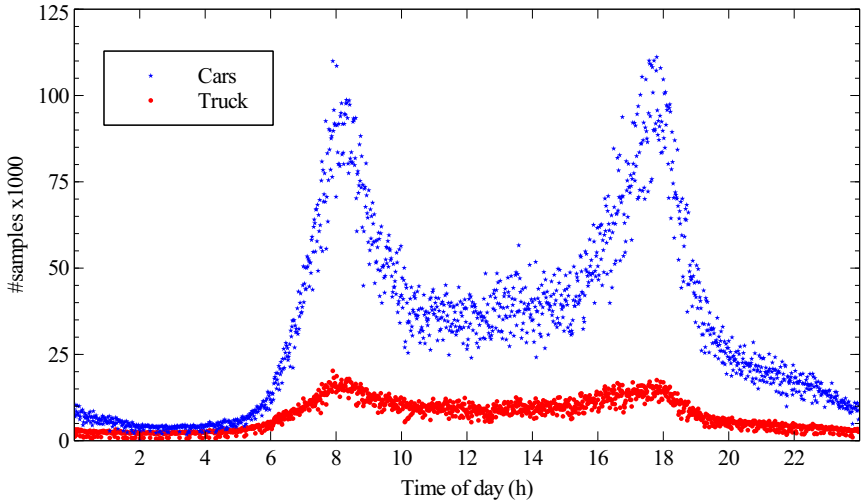


(b) Step 5: optimising MP_d



(c) Step 6: optimising MP_{wd}

Figure 3.4 Number of car and truck samples for March 24, 2015.



the dataset has an acceptable representation of the live traffic. However, in what follows only the data from 6:00 to 22:00 is used as at night, traffic load is lower, yielding less data and a higher risk of outlier noise influence.

After FCD generation, the resulting dataset is further filtered according to the road class classification of the underlying digital map. Dataset C_1 consists of 25,352 links spanning 5,122 km, classified as roads of international importance, typically highways serving high traffic volumes, continuing across country borders. Analogously, the second dataset C_2 consists of 36,443 links spanning 6,080 km, classified as roads of national importance, typically smaller highways connecting different cities within provinces. Each dataset was then partitioned in a validation set and a training set, with the first week of each month in the validation set while all remaining data was grouped in the training set. A final partitioning of the data was done by grouping data according to the day of the week.

3.4.2 Individual profile fitting

The individual models were implemented and evaluated for all datasets. Both the Bin model and TRHC model were implemented in Java while the Fourier series and Wavelet model were implemented in Matlab. The resulting functions are generally similar with the differences being discussed in the next paragraphs. Note that the data shown consists of the training dataset and that identical samples (coming from different days) are plotted on the same position.

Fig. 3.5 shows example fits for all models for various links with 10 variables per link for the Bin, Wavelet and TRHC models but 11 parameters for the Fourier model. The number of parameters is used here to compare the results of the different model fits as it is clear and unambiguous.

For the Bin model, the main disadvantage is that it does not take into account the start or end of the rush hours due to its fixed time phase changes. As shown in Fig. 3.5a, this results in successive steps around the onset and dissipation of the congestion. Another disadvantage can be seen in Fig. 3.5b between 13:00 and 15:00 and Fig. 3.5c between 11:00 and 12:00 where the dataset contained some speed samples in the range 10-40 km/h. This results in the value for that specific timeslot to be lower (as the low values lower the average of all samples in that slot). This is less problematic in Fig. 3.5a. While there the speed values are more widely spread around the mean (during the noon period), the extreme outliers of Fig. 3.5b cause fluctuations between successive timeslots.

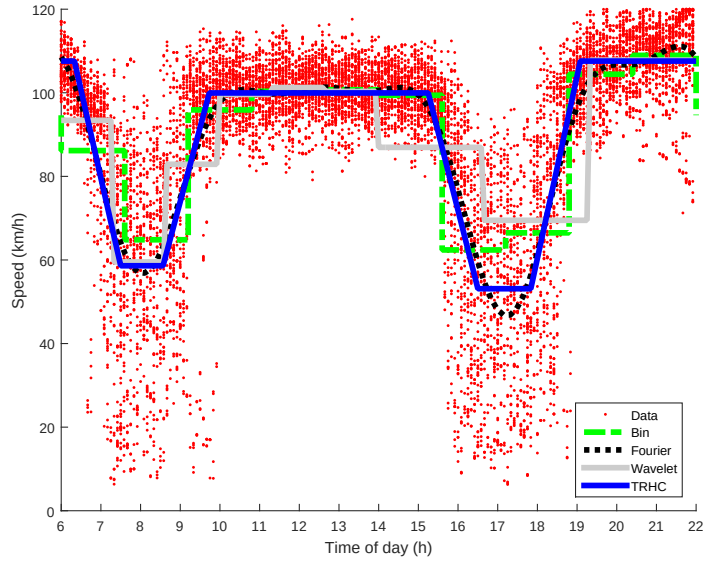
The results for the Fourier series are in line with the other models. However, for sharp changes in the profile, the Fourier fit does show some artefacts as seen in the fluctuations during the noon period in Fig. 3.5c and 3.5d. More base functions would be needed to capture this dynamic but this would increase the number of variables used per link.

The Wavelet model is hindered by the limited number of coefficients. The speed profile estimates are made by passing the retained coefficients through the reverse filter banks of the wavelet transformation. For the reverse filter banks of the Haar-wavelet, a single coefficient is converted to a scaled version of the step function. As only a few coefficients are passed through the filter bank, the individual contributions are still visible in the total response as shown in Fig. 3.5a. Another noticeable problem is that the speed estimates during rush hours are higher compared to the other models. Similar to the Bin model, this is caused by averaging the values in the peak with those on the flanks and before/after the peak. As depicted in the evening peak of Fig. 3.5a and the morning peak of Fig. 3.5d, the additional initial averaging of the input values reinforces this effect.

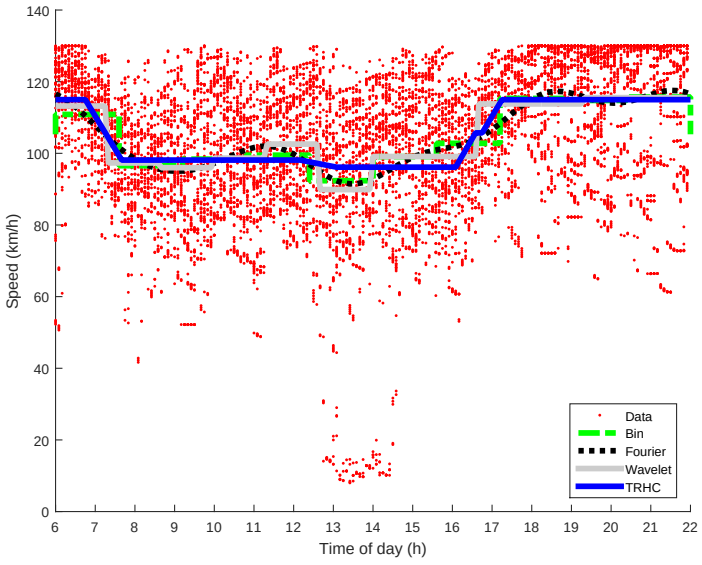
As the TRHC model was developed to capture the rush hours dynamics, it follows the rush hour periods quite well. Note that, as Fig. 3.5b shows, the peak speeds are not necessarily lower than the noon speeds. The fitting also allows the peak speeds to be higher. This can be the case for roads connected to upstream congested links for which the rush hour congestion bottleneck lowers the load on the link, allowing for smoother traffic.

However, for some instances, the peak speeds can be overestimated, as Fig. 3.5a shows a lower evening peak speed for the Fourier model. While this effect could be mitigated by allowing different slopes for the onset and dissipation of the evening peak, this would require additional variables and more complex fitting.

Figure 3.5 Illustration of the model fits to the source FCD.

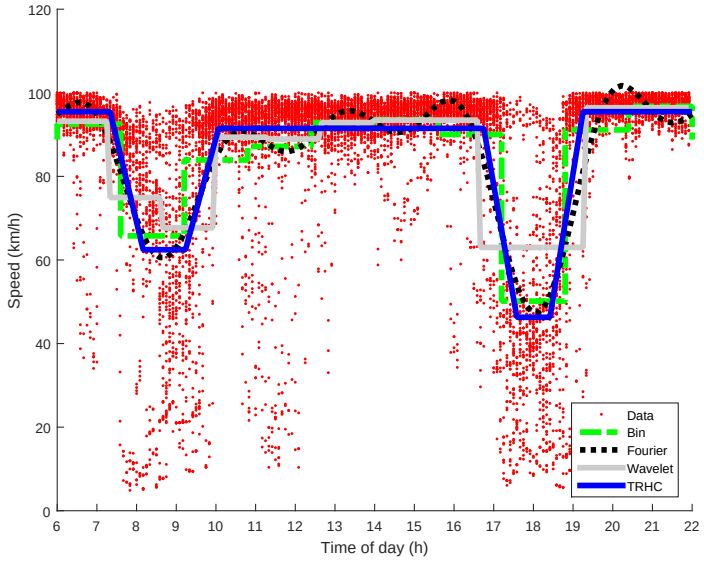


(a)

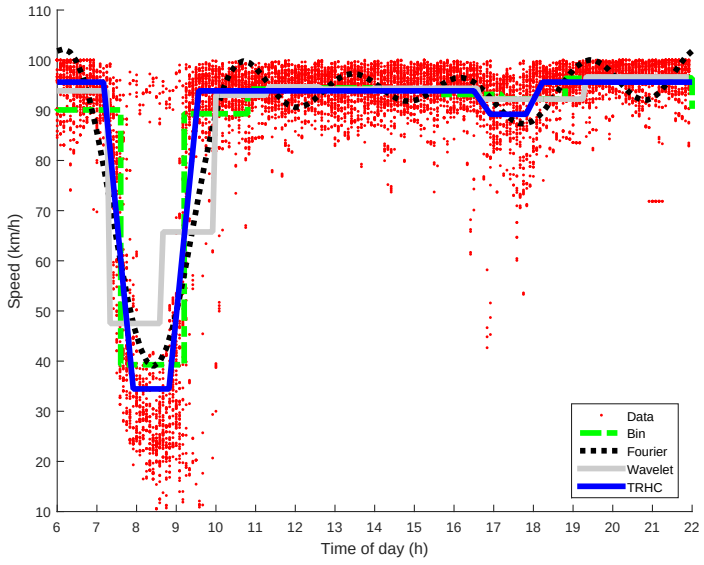


(b)

Figure 3.5 Illustration of the model fits to the source FCD (continued).



(c)



(d)

3.4.3 Full dataset analysis

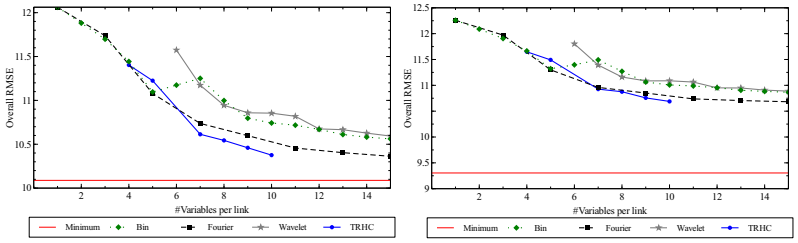
To evaluate the applicability of the presented models, they were applied to all 14 training datasets mentioned above (1 for each road classification and day of the week combination). The resulting fits were then evaluated by calculating the Root Mean Squared Error (RMSE) between the data samples and the speed profile value. The individual RMSE values were weighted by the number of samples for the link and then averaged over all links to yield the overall RMSE. This was done for both the corresponding training set and again for the corresponding validation set. The average error for the training set is an indication of the modelling power, showing how well each of the models is capable of capturing the information in the input data. The average error for the validation set can be used as a measure for the predictive power of the models, showing how well the model succeeds in predicting the values for the days in the validation set. For each dataset, the errors are calculated along with the minimal error attainable. This lower bound is the average error of the RMSE-optimal speed profile for which the value at time t is given as the mean of all values in the dataset. This results in the lowest possible error for any profile for that dataset.

For the Bin and Fourier models, different fits could readily be made by incrementing the number of variables per link. The Wavelet model was also easily extensible and the first 6 to 15 coefficients of the wavelet transform were retained. For the TRHC model, which is not so readily extensible in terms of number of fitted variables, different fits were generated by making additional assumptions on the general model as discussed in the model presentation (see end of Section 3.3.4.1).

Fig. 3.6 shows the results for the Tuesday datasets. As could be expected, allowing more variables to capture the speed evolution generally lowers the overall RMSE. However, this does not hold for the Bin model as indicated by the spike at 7 variables in Fig. 3.6a and 3.6b. This is caused by the borders of the model not properly aligning with the congestion times, mixing congestion values with free-flow values. The TRHC model also has a spike at 9 variables in Fig. 3.6c and 3.6d. This is the model without a noon value, showing that differentiating noon from base/free-flow speeds is an important part of the model.

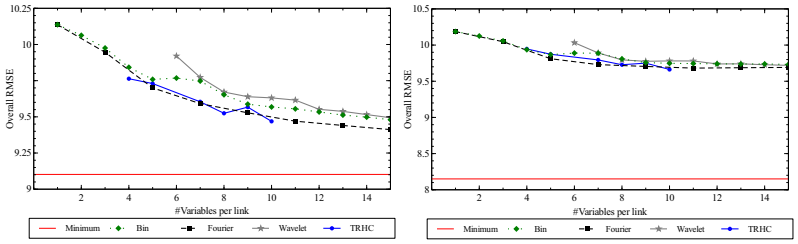
As shown in the figures, the Wavelet model does not perform well. This is again partly attributable to the initial averaging of the input speed values. Before wavelet transformation is applied, the individual speed values are averaged per minute. This removes part of the available information in the input data, which is not done by the other models. By working on the individual samples, the other models can optimise the MSE more successfully. Furthermore, while the time-space wavelet functions allow to capture details in specific parts of the time range, the small number of coefficients does not permit to focus on the individual regions. As only the highest level coefficients are used, the time-space grouping

Figure 3.6 Training and validation MSE results for Tuesday.



(a) RMSE on C_1 training dataset

(b) RMSE on C_1 validation dataset



(c) RMSE on C_2 training dataset

(d) RMSE on C_2 validation dataset

of the hierarchical wavelet coefficients cannot fully be exploited. As the Fourier model shows, it is more beneficial to fully contain the information in the frequency domain.

The minimal overall RMSE is quite high. This indicates, as could be suspected from Fig. 3.5, that the variation of the speed values in the dataset is quite high. This is partly caused by merging the individual days with varying traffic conditions (the rush hour on a specific Monday could have started a bit earlier than a week before or an accident somewhere near the link disrupted typical flows) and partly by the inherent FCD sampling and averaging nature with varying car and truck contribution. The minimum RMSE for the validation sets are smaller as they contains less values per minute, which tends to reduce the MSE for each minute.

As shown in Fig. 3.6a, the TRHC model has the best performance at 10 variables, fully using the presented model of Fig. 3.2. While the model was created with intuitive traffic parameters, it still manages to capture more information than the other models for the same number of variables.

The extensibility of the Bin and Fourier models allows to generate models with an arbitrary number of variables. While this tends to improve the fit for the training dataset, as indicated by the downward trend of the Bin and Fourier models in Fig. 3.6a and 3.6c, the result on the validation dataset may worsen, as shown for the Fourier model in Fig. 3.6d, due to overfitting to the training data.

While Fig. 3.6 shows the models extensible performance for the Tuesday datasets, Tables 3.1 and 3.2 show the results for all datasets. For each day and road class, the tables quantify the overall RMSE for each model trained on the training set. As discussed above, the models were first analysed for their modelling power by evaluating to the training datasets and again for their predictive power by evaluating against the validation datasets. The Bin, Wavelet and TRHC models were allowed 10 variables while for the Fourier model, the reported numbers are the averages between the 9 and 11 variable version of the model. For each dataset, first the minimum overall RMSE is given. The reported values for the individual models are the relative additional errors of the model. The value of 6.48% for the Bin model on the Tuesday training dataset thus indicates that it had an overall RMSE of (1+6.48%) · 10.0883 = 10.7419 at 10 variables as depicted in Fig. 3.6a.

Looking at the variation in the reported minimum overall RMSE values over all datasets, it roughly follows the weekly traffic load pattern. Higher minimum values indicate more variation in the FCD (grouped by day), resulting in lower values during the weekend and slightly lower values for Wednesdays and Fridays. While the difference is most pronounced for the C_1 roads, it is noticeable for the lower classified C_2 roads as well.

Comparing the different models for their modelling power through Table 3.1, the TRHC model performs slightly better, averaging a relative 3.00% additional RMSE value, compared to the Fourier, Bin and Wavelet models with 3.83%, 5.41% and 5.80% respectively. As it was designed to capture rush hour dynamics, it performs best during weekdays, going further below its 3.00% average. As rush hour impact lowers, the TRHC model performs slightly worse. The Wavelet model then performs quite well with a best performance of 2.74% for C_1 on Saturday.

Overall, the TRHC model scores best at capturing the rush hour dynamics, followed by the Fourier model. For weekends, without much rush hour traffic, the Bin and Wavelet model have the advantage.

3.4.4 Congestion analysis

To further illustrate the applicability of speed profiles, this section presents a short evaluation of the Dutch road network, using the estimated profiles from Section 3.4.3. Using the inherent distributed nature of FCD, a clear overview of the entire network can be obtained, in contrast to the localised loop information. Problem areas can be detected and optimised, improving safety and traffic flow whilst reducing economic losses and pollution. As discussed in Section 3.3.4, the TRHC model was designed to capture rush hour dynamics in intuitive parameters. As this allows for link and network analysis to be done directly on the model parameters and the results of Tables 3.1 and 3.2 indicate it captures speed evolution best, the analysis is done using the TRHC model.

Table 3.1: Overall RMSE results for the training datasets on each day of the week.

		Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Average
C_1	Minimum	9.6040	10.0883	8.9073	10.0457	9.1329	8.0061	5.9346	8.8170
	Bin	+ 9.29%	+ 6.48%	+ 5.62%	+ 5.85%	+ 4.09%	+ 2.83%	+ 3.69%	+ 5.41%
	Fourier	+ 4.16%	+ 4.35%	+ 3.98%	+ 3.89%	+ 3.20%	+ 3.56%	+ 3.66%	+ 3.83%
	Wavelet	+ 6.44%	+ 7.59%	+ 7.13%	+ 7.51%	+ 5.43%	+ 2.74%	+ 3.77%	+ 5.80%
	TRHC	+ 2.80%	+ 2.85%	+ 2.97%	+ 2.69%	+ 2.96%	+ 3.27%	+ 3.44%	+ 3.00%
C_2	Minimum	9.0187	9.1021	8.8340	9.0671	8.8809	8.8365	8.0900	8.8327
	Bin	+ 6.90%	+ 5.12%	+ 4.72%	+ 4.62%	+ 4.58%	+ 4.31%	+ 4.93%	+ 5.03%
	Fourier	+ 4.32%	+ 4.36%	+ 4.12%	+ 3.94%	+ 4.18%	+ 4.26%	+ 4.75%	+ 4.28%
	Wavelet	+ 5.45%	+ 5.81%	+ 5.24%	+ 5.28%	+ 5.01%	+ 4.49%	+ 5.19%	+ 5.21%
	TRHC	+ 4.04%	+ 4.03%	+ 4.00%	+ 3.69%	+ 4.21%	+ 4.41%	+ 5.03%	+ 4.20%

Table 3.2: Overall RMSE results for the validation datasets on each day of the week.

		Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	Average
C_1	Minimum	9.9884	9.3043	8.3100	9.7192	8.1656	5.9602	5.2153	8.0947
	Bin	+ 12.09%	+ 18.31%	+ 18.07%	+ 19.37%	+ 12.96%	+ 18.29%	+ 17.05%	+ 16.59%
	Fourier	+ 11.93%	+ 16.01%	+ 16.51%	+ 18.42%	+ 13.23%	+ 19.69%	+ 17.16%	+ 16.14%
	Wavelet	+ 13.11%	+ 19.17%	+ 19.00%	+ 19.87%	+ 13.28%	+ 18.08%	+ 17.00%	+ 17.07%
	TRHC	+ 11.46%	+ 14.88%	+ 15.68%	+ 18.23%	+ 13.44%	+ 19.15%	+ 17.02%	+ 15.69%
C_2	Minimum	8.5126	8.1501	7.8409	8.1674	8.3909	7.3834	6.9261	7.9102
	Bin	+ 17.07%	+ 19.60%	+ 19.29%	+ 21.65%	+ 12.69%	+ 22.16%	+ 24.13%	+ 19.51%
	Fourier	+ 16.75%	+ 18.93%	+ 18.90%	+ 21.35%	+ 12.62%	+ 22.39%	+ 24.49%	+ 19.35%
	Wavelet	+ 17.20%	+ 20.01%	+ 19.63%	+ 21.82%	+ 12.69%	+ 22.09%	+ 24.02%	+ 19.64%
	TRHC	+ 16.65%	+ 18.56%	+ 18.54%	+ 21.14%	+ 12.62%	+ 22.25%	+ 24.27%	+ 19.15%

To quantify network congestion, the average delay index d is used. This metric is calculated as given by Eq. (3.4) and represents the average relative increase in delay per km road throughout the day. It is the ratio of the average delay throughout the day to the travel time in free-flow conditions. While it is defined relative to the travel time in free-flow (to avoid bias towards roads with lower speed limits), the average delay could also be used. A delay index of 50% thus corresponds to a road on which the average delay is half the travel time in free-flow. The average delay focuses on the sensitivity of the road to normal usage and how the traffic on the road influences the attainable speeds. While the delay index receives high contributions from high congestion peaks (with high instantaneous delays), it also takes into account the duration of the peak and the delay accrued during the noon period.

$$d = \frac{\int_{t_m}^{t_e} \frac{1}{v(t)} - \frac{1}{b_v} dt}{(t_e - t_m) \cdot \frac{1}{b_v}} \tag{3.4}$$

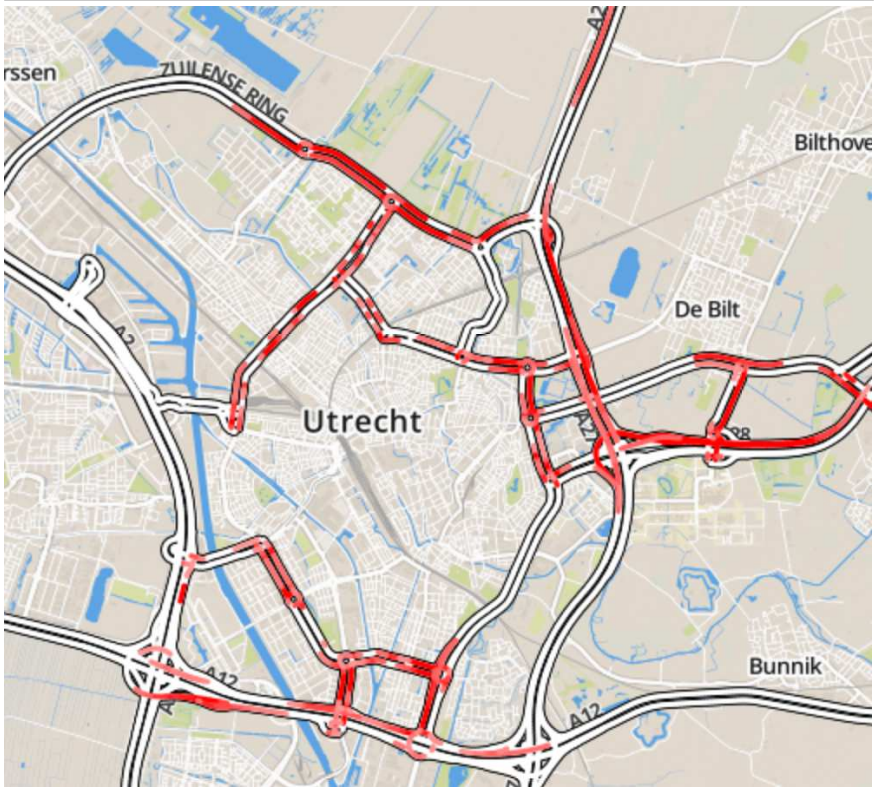
With t_m and t_e denoting the start resp. end time of day in the analysis, the average delay is calculated by averaging the instantaneous delays on the links. For each specific time, the delay on a link (per unit length) is calculated as the difference between the travel time using the instantaneous speed $v(t)$ and the free-flow speed b_v. As the $v(t)$ function is piecewise continuous, the integration can be solved analytically, again enabling analysis directly on the model parameters.

Note that the analysis only uses the estimated profiles of the weekdays as no rush hours are expected in weekends. The estimated profiles for each weekday are used in the metrics above and then averaged to obtain the final estimate. Also note again that the analysis presented here is an illustration of the model application. More elaborate evaluation with additional metrics or spatial/temporal aggregation can be performed.

The delay index, calculated as described above, is plotted in Fig. 3.7. As similarly reported in [24], severe congestion is found in the west of the country (South Holland, near Rotterdam, Gouda and The Hague). Other indicated areas mostly include bottlenecks around exits and ring roads, indicating that these roads are prone to delay throughout the day. Fig. 3.8 shows this in more detail for the city of Utrecht. Furthermore noteworthy are the scattered small stretches of problem roads detected. These are mostly attributable to local road properties like intersections and bridges, which allow for low travel times under low traffic load but relatively high delay under normal to high traffic load conditions.

Figure 3.7 Delay index for the Netherlands. Dark red roads correspond to roads with a delay index of 33.3% or higher. Image © OpenStreetMap contributors.

Figure 3.8 Delay index for Utrecht. Image © OpenStreetMap contributors.



3.5 Summary

This paper focused on modelling speed evolution throughout the day on individual links using historic floating car data (FCD). Starting from raw positioning data, the complete procedure of map matching, data processing and aggregation of a large FCD dataset was shown with respect to the FCD source properties (e.g. low frequency sampling, limited penetration and positioning accuracy). The data was then used to model the speed evolution using 4 different models. Aside from a typical Bin model (consisting of speed estimates for a number of fixed timeslots), the Fourier model (a Fourier series regression) and a Wavelet model (capturing the speed profile in wavelet coefficients), the Trapezoidal Rush Hour Congestion (TRHC) model is presented in this paper. This model consists of a base speed with individual morning and evening rush hour peaks. It is specifically aimed at modelling rush hour dynamics through intuitive model parameters to allow easy integration and use in further applications.

The models were evaluated on their modelling and prediction power on 4 months of live data collected by a commercial FCD system in the Netherlands, subdivided in training and validation data per day of week. While the internal variance of the speeds on a link is quite big, the models capture the variation in the training set quite well with only 3.00% to 5.80% additional root mean squared error.

Of the 4 models, the Trapezoidal Rush Hour Congestion (TRHC) model captured best the rush hour dynamics through its custom provisions. The Fourier model scored slightly worse, mostly due to needing additional model parameters to account for heavy congestion peaks. The Bin model is less suited for rush hours but scored better for low congestion weekends. The Wavelet model performed badly, due to the low number of retained coefficients.

Additionally, a basic congestion analysis of the Dutch road network through the individual model parameters was shown. Congested roads were determined based on the average relative delay per kilometre. The detected problem areas corresponded with previous studies which further illustrates the validity of the TRHC model.

References

[1] M. Westerman, R. Litjens, and J.-P. Linnartz. *Integration Of Probe Vehicle And Induction Loop Data: Estimation Of Travel Times And Automatic Incident Detection*. Technical report, Institute of Transportation Studies, University of California, Berkeley, 1996.

[2] T. Miwa, D. Kiuchi, T. Yamamoto, and T. Morikawa. *Development of map matching algorithm for low frequency probe data*. Transportation Research Part C: Emerging Technologies, 22:132–145, 2012.

[3] B. Y. Chen, H. Yuan, Q. Li, W. H. Lam, S.-L. Shaw, and K. Yan. *Map-matching algorithm for large-scale low-frequency floating car data*. International Journal of Geographical Information Science, 28(1):22–38, 2014.

[4] X. Liu, K. Liu, M. Li, and F. Lu. *A ST-CRF Map-Matching Method for Low-Frequency Floating Car Data*. IEEE Transactions on Intelligent Transportation Systems, 18(5):1241–1254, may 2017.

[5] S. Ancona, R. Stanica, and M. Fiore. *Performance boundaries of massive Floating Car Data offloading*. In 2014 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pages 89–96. IEEE, apr 2014.

[6] O. Altintasi, H. Tuydes-Yaman, and K. Tuncay. *Detection of urban traffic patterns from Floating Car Data (FCD)*. Transportation Research Procedia, 22:382–391, 2017.

[7] M. Houbraken, S. Logghe, M. Schreuder, P. Audenaert, D. Colle, and M. Pickavet. *Automated Incident Detection Using Real-Time Floating Car Data*. Journal of Advanced Transportation, 2017:1–13, 2017.

[8] M. Houbraken, S. Logghe, P. Audenaert, D. Colle, and M. Pickavet. *Examining the potential of Floating Car Data for Dynamic Traffic Management*. IET Intelligent Transport Systems, pages 1–13, jan 2018.

[9] F. Rempe, P. Franeck, U. Fastenrath, and K. Bogenberger. *A phase-based smoothing method for accurate traffic speed estimation with floating car data*. Transportation Research Part C: Emerging Technologies, 85(October):644–663, 2017.

[10] T. Seo, A. M. Bayen, T. Kusakabe, and Y. Asakura. *Traffic state estimation on highway: A comprehensive survey*. Annual Reviews in Control, 43:128–151, 2017.

[11] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. *Non-parametric estimation of route travel time distributions from low-frequency floating car data*. Transportation Research Part C: Emerging Technologies, 58:343–362, sep 2015.

[12] C. Shi, B. Chen, and Q. Li. *Estimation of Travel Time Distributions in Urban Road Networks Using Low-Frequency Floating Car Data*. ISPRS International Journal of Geo-Information, 6(8):253, 2017.

[13] J. F. Ehmke, S. Meisel, and D. C. Mattfeld. *Floating car based travel times for city logistics*. Transportation Research Part C: Emerging Technologies, 21(1):338–352, 2012.

[14] a. Pascale, F. Deflorio, M. Nicoli, B. Dalla Chiara, and M. Pedroli. *Motorway speed pattern identification from floating vehicle data for freight applications*. Transportation Research Part C: Emerging Technologies, 51(305):104–119, 2015.

[15] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. *Contraction hierarchies: Faster and simpler hierarchical routing in road networks*. Experimental Algorithms, 2:319–333, 2008.

[16] G. V. Batz, R. Geisberger, S. Neubauer, and P. Sanders. *Time-dependent contraction hierarchies and approximation*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6049 LNCS:166–177, 2010.

[17] M. Vanlommel, M. Houbraken, P. Audenaert, S. Logghe, M. Pickavet, and P. De Maeyer. *An evaluation of section control based on floating car data*. Transportation Research Part C: Emerging Technologies, 58:617–627, 2015.

[18] M. Yildirimoglu and N. Geroliminis. *Experienced travel time prediction for congested freeways*. Transportation Research Part B: Methodological, 53:45–63, 2013.

[19] M. Gastaldi and R. Rossi. *A methodology for calibrating road link travel time functions using data from driving simulator experiments*. Procedia - Social and Behavioral Sciences, 20(July 2015):656–665, 2011.

[20] P. Christidis and J. N. Ibáñez Rivas. *Measuring road congestion*. Technical report, 2012.

[21] B. Hellinga, P. Izadpanah, H. Takada, and L. Fu. *Decomposing travel times measured by probe-based traffic monitoring systems to individual road segments*. Transportation Research Part C: Emerging Technologies, 16(6):768–782, dec 2008.

[22] E. Dijkstra. *A note on two problems in connexion with graphs*. Numerische mathematik, 1(1):269–271, 1959.

[23] I. Daubechies. *The wavelet transform, time-frequency localization and signal analysis*. Information Theory, IEEE Transactions on, 36(5):961–1005, 1990.

[24] Centraal Bureau voor de Statistiek. *Transport en mobiliteit 2015*. Technical report, 2015.

"The science of today is the technology of tomorrow."

— Edward Teller (1908 - 2003)

4

Examining the Potential of Floating Car Data for Dynamic Traffic Management

With the previous chapters establishing the value of FCD for network analysis, we now turn to traffic management. While network analysis offers insights in the network conditions, the insights themselves need to be used in the field to actually improve traffic. In this chapter, we look at if and how FCD could be used directly in dynamic traffic management systems.

⋆ ⋆ ⋆

Maarten Houbraken, Steven Logghe, Pieter Audenaert, Didier Colle, Mario Pickavet

Submitted to IET Intelligent Transport Systems, Augustus 2016, accepted January 2018.

Abstract Traditional traffic monitoring systems are mostly based on roadside equipment measuring traffic conditions throughout the day. With more and more GPS enabled connected devices, Floating Car Data (FCD) has become an interesting source of traffic information, requiring only a fraction of the roadside equipment

infrastructure investment. While FCD is commonly used to derive historic travel times on individual roads and to evaluate other traffic data and algorithms, it could also be used in traffic management systems directly. However, as live systems only capture a small percentage of all traffic, its use in live operating systems needs to be examined. In this paper, we investigate the potential of FCD to be used as input data for live automated traffic management systems. The FCD in this study is collected by a live, countrywide FCD system in the Netherlands covering 6-8% of all vehicles. The (anonymised) data is first compared to available roadside measurements to show the current quality of FCD. It is then used in a dynamic speed management system and compared to the installed system on the studied highway. Results indicate the FCD setup can approximate the installed system, showing the feasibility of a live system.

4.1 Introduction

Digital connected devices have become widespread in the past decade. As people are travelling, these 'smart' devices can send and receive traffic information on the road. While this allows users to optimise their journey by avoiding traffic jams, it also improves the quality of the traffic information, as the data supplied by the users is fed back into the system and used to more accurately estimate traffic state. This source of data, termed Floating Car Data (FCD), presents an interesting opportunity in the field of traffic monitoring and analysis.

Traffic data generation has evolved significantly. Earliest field trials involving probe vehicles were reported in the 90's [1] with vehicles aggregating and compactly transmitting data over low bandwidth connections and for predefined positions as no universal positioning (such as GPS) was available. These technological limitations guided development in traffic monitoring to focus on Roadside Equipment (RE), such as inductive loops which provide a detailed view on the traffic conditions at a specific location. A large body of research has since grown, focusing on using loop data for various applications such as travel time estimation [2, 3], travel time prediction [2] and vehicle re-identification [4]. While loop data system technology has matured, it remains prone to malfunctions (e.g. [5] reports $> 25\%$ broken loop sensors in a Chinese city), requiring costly maintenance. Together with the localised nature of loop information, a need still exists for cost-effective, alternative systems covering larger areas.

Cellular communication networks can also be used to monitor travellers, using mobile phone call handover events to generate user position data [6, 7] but the complexity of user triangulation among cell towers hindered the breakthrough of the system.

With the introduction of GPS technology with better positioning information outperforming cellular triangulation, a wide range of trial and production FCD sys-

tems were developed. One of the key characteristics among the different technical implementations is the sample frequency. As vehicles move throughout the network, they record and transmit their position periodically to the system for further processing and aggregation. The frequency of data transfer varies, depending on the available bandwidth of the underlying communication system and the energy consumption (in-car versus stand-alone battery-powered devices) ([8] provides a discussion on FCD transfer cost). High-frequency sampling systems with frequencies of one sample every 1-10 seconds allow very accurate vehicle tracking as the difference in successive positions is quite small. In low-frequency systems, with sampling only every 30-60 seconds (or more), tracking is more complex. The exact vehicle path needs to be reconstructed/estimated as the vehicle can travel several kilometres between successive positions. This requires more complex map matching algorithms [9–12].

While FCD is becoming more widespread, it still remains uncertain what penetration rate is necessary for the desired system performance. Depending on the intended application and data sample frequency, the required/advised fraction of all traffic varies. For travel time estimation, [13] uses a fleet of 1,500 taxis in Stockholm to derive travel times for a 1.5 hour timeslot on a specific route with very low-frequency sampling (data reports only every 2 minutes). The same Stockholm system is also used by [14] to estimate route travel time distributions on 27 selected routes. Neural networks are applied in [15] to link travel time estimation with FCD although most results are based on simulated data. For a cellular-based FCD system, [6] reports results for travel times covering 1-3% of all traffic on average while [16] states success with 2.4% of all private cars in Rome. Similarly, [17] suggests 2.4 and 10% for highways and arterial roads respectively and [18] hints at 2-3% for cell phone-based systems.

For incident detection, most installed systems use pure infrastructure-based data collection (e.g. California algorithm [19] and McMaster algorithm [20]). Using vehicle-to-roadside communication, [21] monitors cars passing fixed measurement locations to estimate vehicle headways and lane switches to detect incidents. Looking at speed changes, the UCB algorithm [22] detects incidents from an FCD dataset with an estimated 0.1% coverage. In [23], the UCB algorithm is compared to a CUSUM algorithm which uses increased travel time in moving time windows. Using a modified CUSUM algorithm, [24] monitors flooding events in Beijing. Starting from 40,000 taxis providing data every minute, they detect the specific FCD characteristics when roads are flooded and report these events. Relying solely on FCD for incident detection, [25] advises 1% while [26] reports 1.5%. In [27], the operational Dutch Automated Incident Detection (AID) system (also used in this paper) is studied on 1.6 km British motorway with interpolated loop data. The Dutch AID is a queue tail warning system coupled to an overhead Variable Message Sign (VMS) system with Variable Speed Limit (VSL) being

displayed. The study found 1% FCD penetration rate required for similar performance compared to loop-based systems. While reliability requirements differ among the applications and various technical specifications e.g. frequency, latency and data aggregation impact the required penetration rate significantly, overall, the reported percentages are in the range of 1 to 10%, indicating the order of magnitude required.

One of the main directions in current research is how to combine FCD with RE to achieve *data fusion*. Both FCD and loop data are used simultaneously in [28] to estimate the fundamental diagram. Similarly, [29] fuses camera data with FCD while [30] fuses loop data with FCD to estimate traffic conditions.

This paper focuses on the potential of FCD for dynamic traffic management, more specifically variable speed limits to, among other purposes, homogenise traffic speeds or prevent end-of-queue collisions (see [31]). Most current day traffic management systems rely on RE because these provide a more complete and detailed view on the traffic conditions (as they monitor all traffic). However, for large scale network monitoring, FCD is more interesting as it does not require additional measuring infrastructure, compared to RE needing to be installed on all roads of interest. FCD does require sufficient probe vehicles, with the penetration rates studies mentioned above. However, most of these studies are based on simulated FCD and/or small FCD systems, thereby limiting the significance of the results. For example, [27] studies the application of FCD but uses interpolated loop data as FCD and uses a theoretical evaluation benchmark to determine FCD suitability. In this paper, we bridge this gap by using real FCD collected by an operational countrywide FCD platform and comparing it to the live captured output of the installed system. The rest of this paper is structured as follows. Before the main focus of this study is presented, Section 4.2 first details the existing loop installation and traffic management algorithm on Dutch highways. Section 4.3 is dedicated to examining the potential of FCD for dynamic traffic management by proposing an FCD-based speed advice algorithm. Section 4.4 proposes the comparison methodology used to compare the loop to the FCD VSL algorithm. Section 4.5 shows the global results of the algorithm.

4.2 Existing system & infrastructure

As in this study, we focus on the FCD potential, we first present the currently operational Automated Incident Detection (AID) system in our study area, shown in Fig. 4.1 and annotated in Table 4.1. The Dutch highway loop AID system contains a queue tail warning system to improve traffic conditions and road safety. This system focuses on detecting congestion and other incidents and warns upstream traffic by using dynamic overhead signs. Once congestion is detected, drivers are alerted to reduce their speed and avoid rear-end collisions.

Figure 4.1 Overview of A27 study area. The monitored route (in red) runs from Vianen (top right) southbound to Gorinchem (bottom left). The points of interest of Table 4.1 are marked by the blue markers.

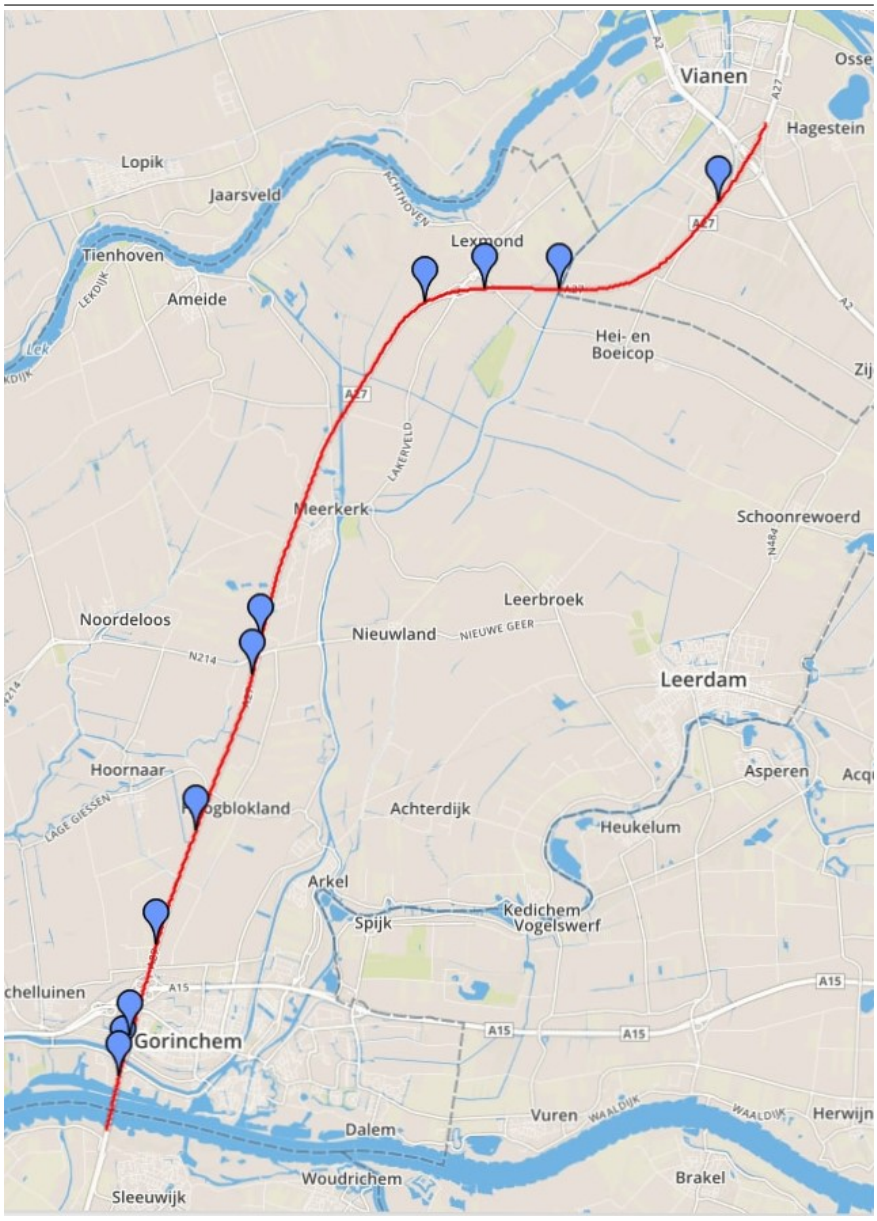


Table 4.1: Points of interest in A27 study area.

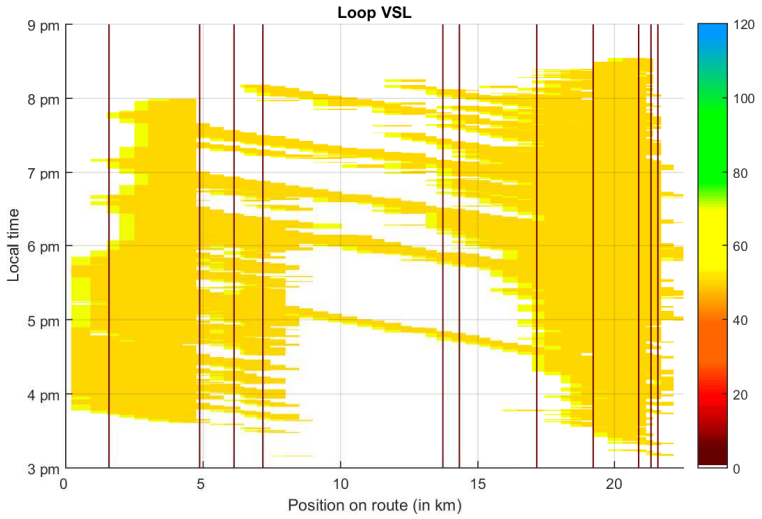
Point of interest	**Distance from route start (in km)**
A2 Everdingen On-ramp	1.53
Bridge Merwedekanaal + lane reduction	4.84
Lexmond Exit	6.09
Lexmond On-ramp	7.17
Noordeloos Exit	13.71
Noordeloos On-ramp	14.32
Scheiwijk Gas station	17.15
A15 Gorichem Exit	19.22
A15 Gorichem On-ramp	20.84
Avelingen Exit	21.31
Avelingen On-ramp	21.55

The system consists of overhead signs installed above the highway, with each dynamic road sign coupled to a dual inductive loop. Each sign keeps a running average v_{cur} of the individual vehicle speeds registered by the loop sensor. When a car passes the sensor with a speed v_s lower than v_{cur}, the current average is changed to $(1 - \alpha_{dec}) \cdot v_{cur} + \alpha_{dec} \cdot v_s$. If $v_s > v_{cur}$, the update is done with a different weighting factor α_{acc} to allow different switching behaviour when traffic is speeding up or slowing down.

Parameters α_{acc} and α_{dec} allow to tune the sensitivity of the average to faster and lower samples respectively, with $0 \leq \alpha_{acc}, \alpha_{dec} \leq 1$. The weighted speeds v_{cur} are monitored to detect congestion events and other incidents. When an individual v_{cur} drops below a predefined lower threshold v_{ON} e.g. 35 km/h, the controller switches on the overhead sign. When traffic improves and v_{cur} becomes larger than the upper threshold v_{OFF} e.g. 50 km/h, the sign is switched off and will show *BLK* (= blank). These thresholds can be set differently (with $v_{on} \leq v_{off}$), creating hysteresis to avoid excessive switching of the road sign. The values for v_{ON} and v_{OFF} were set by the Dutch road operators according to their desired sign switching.

While each controller determines its sign based on its own v_{cur}, the individual controllers are also linked to allow overall consistent signage. Controllers at the same location but monitoring different lanes communicate to avoid inconsistent speed advices e.g. the leftmost lane getting a lower speed advice than the rightmost lane. While the signs on the individual lanes can differ, this only occurs when road operators input manual overrides from the regional traffic management central. Controllers also take into account the first downstream detector (typically within 500 m) to better capture backward propagating congestion waves. When a speed sign is activated by its controller, the first upstream controller (within 500 m) also activates its road sign, allowing drivers to be alerted before the congestion

Figure 4.2 Spatio-temporal plot of the loop VSL on the A27 highway on Thursday 14 January 2016, from 3 pm to 9 pm. The installed overhead signs shows 50 km/h (orange) in the detected traffic jam itself, 70 km/h (yellow) just upstream of the congestion and *BLK* (white) outside of the congestion. The black vertical lines denote the positions of interest from Table 4.1.



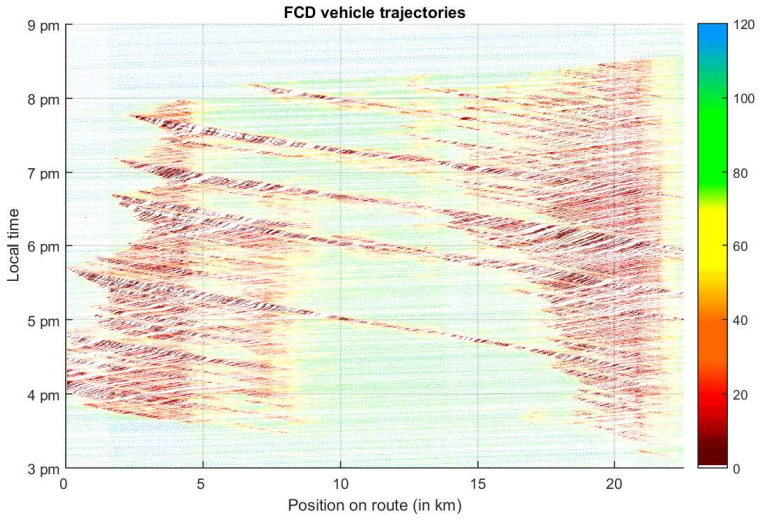
wave reaches the next controller location. The second upstream controller (within 1,000 m) is also alerted to activate an additional (less strict) speed advice, to slow down traffic more gradually. Note that the distances given are typical values, as the distance between loops can vary.

This system thus converts individual vehicle samples to incident reports and dynamic speed advice along the highway. As an individual sign at a specific location is valid on the route until the next downstream sign, the combined signage provides dynamic speed advices for the entire route with a granularity depending on the installed loop hardware. For the A27 route in our study, Fig. 4.2 shows the individual road signs throughout the day. The black vertical lines, denoting the positions of interest from Table 4.1, are plotted to explain some structural properties of the road e.g. the 3-to-2 lane reduction bottleneck at km 4.84.

4.3 FCD System

The aim of this paper is to show the potential of FCD for dynamic traffic management. In this section, we first elaborate on the FCD properties of the underlying FCD system before presenting the FCD algorithm.

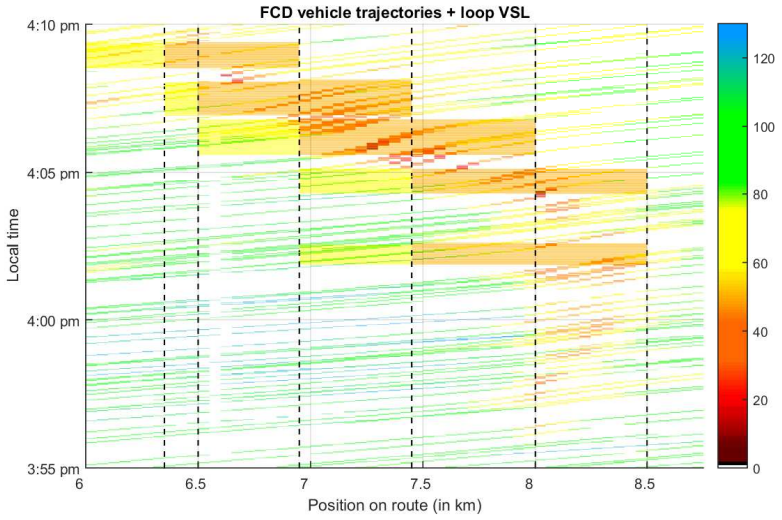
Figure 4.3 Individual FCD vehicle trajectories on the A27 highway on Thursday 14 January 2016, from 3 pm to 9 pm. Vehicles drive from the left to right, with their speeds being colour-coded in the figure. When multiple vehicles are on the same segments, their average speed is shown.



4.3.1 FCD data

The main difference between FCD and loop data is the distributed nature of vehicle tracking. FCD consists of individual probe vehicle measurement samples, each denoted with a timestamp, an anonymised identifier, a set of coordinates, a speed estimate v_s and a vehicle heading. As mentioned in Section 4.1, the FCD sampling rate varies significantly and impacts the additional processing required for converting the distributed data. For this study, we use high-frequency FCD, with probe vehicles generating data samples every second. This allows to minimise the data capturing delay the algorithm (see next section) experiences. The samples were matched to an underlying representation of the road network consisting of road segments of 50 m using their coordinates and heading. The system used to generate data in this study operates in the Netherlands, combining positioning data obtained from transport companies as well as from mobile apps. While the system works for the entire country, the area of study is limited to the A27 (see Fig. 4.1). The general system setup consists of instrumented vehicles that sample their own position and speed and relay this information to a central server for processing (to aggregate to average travel times). Fig. 4.3 shows the FCD consisting of individual vehicle trajectories matched to the underlying road.

Figure 4.4 FCD vehicle trajectories and loop variable speed limit signs, detailed view for 15 December 2015. Loop portal locations are indicated by the vertical dashed lines. FCD is plotted as in Fig. 4.3, loop VSL are plotted as an overlay using the same colour code as in Fig. 4.2.

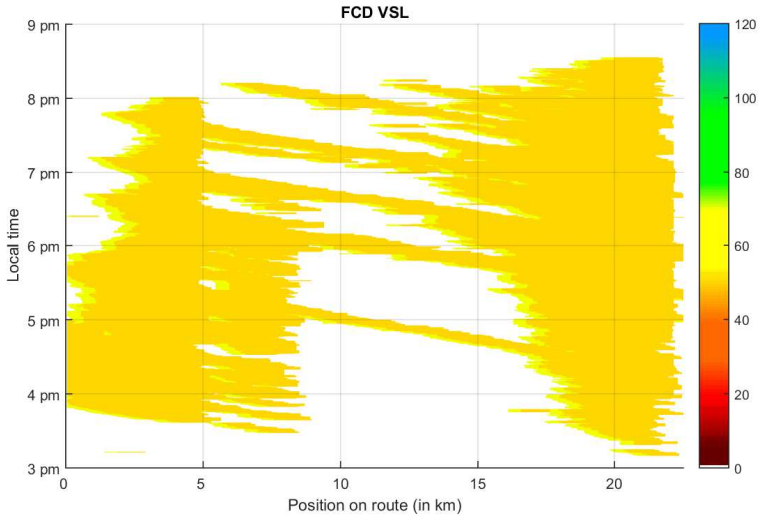


Compared to the loop data, the FCD provides data for road segments between loop locations, revealing the traffic conditions and congestion evolution along the route more clearly. This is illustrated in Fig. 4.4, showing the formation of a backward propagating traffic jam around km 8 at 4 pm. Traffic congestion develops at the loop around 3:56 pm. It travels a little further along the route before turning into a backwards propagating traffic jam. The loop VSL does not activate until the congestion reaches the loops at km 8. Also note the loop VSL switches off several times (e.g. around 4:05 pm) as the small congestion wave travels backwards. The v_{cur} of the detector at km 7.5 is not low enough to trigger the AID, while the v_{cur} at km 8 is already higher than the v_{OFF}.

4.3.2 FCD VSL

Using the high-frequency FCD described above, a VSL system similar to the installed loop AID/VSL system can be developed. While the loop system has a set of fixed portal locations roughly every 500 m, the matched FCD is available on each 50 m segment of the underlying map. On each such segment, a virtual road sign controller is defined, working on the FCD for that segment. As with the loop system, the virtual FCD road sign controller keeps a weighted sum of the vehicle samples on the segment with updates for every new sample according to the pro-

Figure 4.5 FCD VSL result on the A27 highway on Thursday 14 January 2016, from 3 pm to 9 pm. The FCD algorithm, using the data shown in Fig. 4.3, generates speed advice of 50 and 70 km/h which are colour-coded as orange and yellow, respectively.



cedure in Section 4.2. For a route of 22.5 km, this results in 450 segments with associated virtual controllers keeping a weighted sum. Whenever one of these values goes below the predefined threshold (similar to the loop system), the segment is considered to be congested. As with the loops, this triggers an alert for the current segment. However, to mimic the interconnected upstream/downstream loop controllers, the congested status of one virtual detector also triggers speed advice on all segments 500 m upstream and downstream, as well as a less strict advice for segments between 500 and 1,000 m upstream. The resulting FCD VSL is illustrated in Fig. 4.5. While the loop system works with blocks of 500 m, the FCD VSL has a higher granularity, depending on the road map segment size. This higher granularity allows for better traffic jam monitoring.

4.4 Evaluation methodology

To evaluate the quality of the FCD, we focus on the experimental data from the underlying FCD system and the installed loop AID. With the detailed loop logs and the high-frequency FCD with a 6-8% penetration rate, it is possible to focus on the experimental data instead of using micro-simulations for which the results would be tightly coupled to the simulation parameters. We compare the result

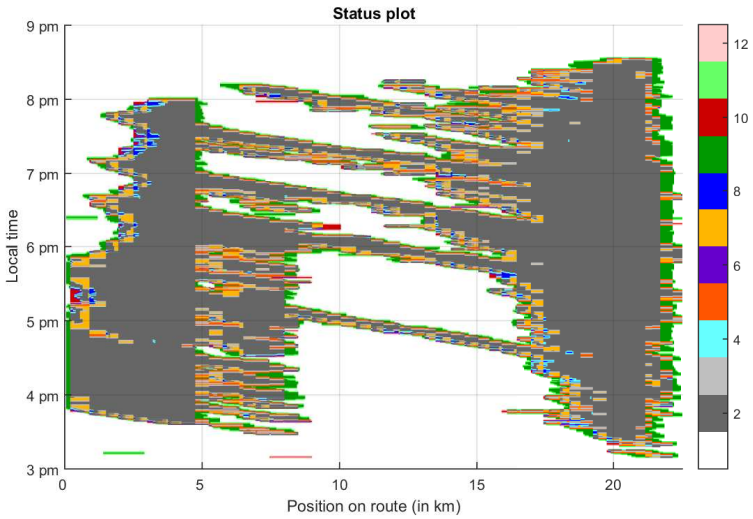
of the loop VSL to that of the FCD VSL, focusing on the signage results rather than on the data itself. This allows to fairly compare both systems, as they both provide speed advice for the entire space-time window of the study area. This is the biggest motivation for focusing on the signage results, as purely comparing loop data to FCD is infeasible, with both sources fundamentally differing in terms of data granularity, covered area and data frequency. Furthermore, data quality also depends on the intended application scenario. With both systems providing a full response for the spatio-temporal study area, the loop system is used as a reference for the FCD algorithm.

4.4.1 State modelling

The following analysis was done on the output of both VSL systems. However, comparing the FCD VSL to the loop VSL also needs careful modelling. A first, straightforward evaluation could be to compare the individual systems output for each timeslot and position in the considered space-time window. While this is simple, it fails at capturing the underlying traffic conditions and comparing both sources. If one source proposes “50” while the other gives no indication, it is unclear what the actual best sign should be. Similarly, if one system would decide on a “70” while the other is now at “50” after already displaying “70” for a few minutes, what penalty should be applied? To clearly differentiate between both systems, the temporal switching/evolution of the signs needs to be accounted for.

The first step in the following analysis consists of modelling the individual sources in temporal states. With the system being discretised in individual space-timeslots, each of those slots is classified according to its current, past and future sign. As the system under study allows for 3 different sign outputs (“BLK” in free flow, “50” and “70” in congestion), the classification yields $3^3 = 27$ possible states. The state for each space-timeslot is determined by looking at the temporal switching at each position. While the current sign is directly available as the output of the system at that position, the past and future signs need further processing. The future sign for the current space-timeslot is determined by taking the (temporally) next sign at that location. If the system currently proposes “70” at the considered location, the future sign is the sign it will switch to in the future. Note however, that this look-ahead is limited to the immediate future, with a window of 1 minute. If no sign switch would occur, the current sign is considered to persist and taken as the future sign as well. A similar definition is used for the past sign, but looking back in time at that location. Given these definitions, a sign that is currently showing “50”, but was showing “70” 30 seconds ago and will show nothing in 40 seconds, will be classified as being in the $(70, 50, \mathit{BLK})$ state.

Figure 4.6 Category classification on the A27 highway on Thursday 14 January 2016, from 3 pm to 9 pm.



4.4.2 State-based comparison

Using this VSL system classification, the 2 systems are more accurately compared. Each space-timeslot of the considered window is classified in one of the $N \times N$ possible $(\textit{stateLoop}, \textit{stateFCD})$ combinations. Here $N = 3^3 = 27$, yielding $27 \times 27 = 729$ combinations. To allow a more tangible analysis on the VSL system level, the states were first grouped according to the situation they correspond to. These categories are summarised in Table 4.2.

This classification gives an indication of the similarity of 2 VSL systems, as well as the specific conservative/relaxed differences between them. A VSL system is termed to be more conservative than the other if it shows a lower speed advice or if the advices are equal but it is going to show a lower speed advice e.g. $(\textit{BLK}, 50, 50)$ is more conservative than $(\textit{BLK}, 50, 70)$. An example of the classification is shown in Fig. 4.6. Each space-timeslot of 50 m and 1 s is colour-coded according to the categories of Table 4.2. With category 2 being prevalent during congestion, the main differences are on the edges of the traffic jam, due to differences in switching behaviour between both algorithms.

To obtain overall comparison metrics, the relative fractions of each category are calculated. This is shown in Table 4.2 for the study in this paper. The *Correspondence Score weight* is explained in the following section.

Table 4.2: Categorisation of VSL system combinations. Note that "−" is used to compactly denote "BLK" while "?" is a placeholder for "BLK", 50 and 70.

Category code	**Description**	**Correspondence Score weight**
1	Both systems are idle (e.g. low traffic), corresponding to state $(-,-,-)$-$(-,-,-)$	1
2	Both systems are in the same state	1
3	Both systems show the same sign but FCD VSL is in a more conservative/restricted state	0.5
4	Both systems show the same sign but FCD VSL is in a more relaxed state	0
5	Loop VSL shows nothing (but has shown or will show), FCD VSL is already showing	0.5
6	FCD VSL shows nothing (but has shown or will show), loop VSL is already showing	-1
7	Both show something but FCD VSL is more restrictive	0.5
8	Both show something but FCD VSL is more relaxed	-0.5
9	Loop VSL is idle $(-,-,-)$, FCD VSL shows something $(?,? \neq -,?)$	0
10	FCD VSL is idle $(-,-,-)$, loop VSL shows something $(?,? \neq -,?)$	-2
11	Loop VSL is idle $(-,-,-)$, FCD VSL shows "BLK" but is aware of problems $(?,-,?) \neq (-,-,-)$	0
12	FCD VSL is idle $(-,-,-)$, loop VSL shows "BLK" but is aware of problems $(?,-,?) \neq (-,-,-)$	-0.5

4.4.3 FCD VSL calibration

The proposed VSL system analysis can now be used as a tool in the high level system comparison and to test the effect of individual VSL system parameter changes. Changing specific parameters in the FCD algorithm (see Section 4.3.2), will result in some of the space-timeslots ending up in different categories, shifting the metrics. More interestingly, the analysis metrics can be used in an optimisation study of the FCD VSL parameters.

The FCD parameters were calibrated to best approximate the switching behaviour of the existing, countrywide implemented AID system. The 12 categories from above were weighted to produce a single metric, the Correspondence Score (CS). The CS aims at estimating the overall correspondence of the FCD AID system to the loop AID system. Each category received a weight factor (see Table 4.2) based on how well the category is desirable to mimic the loop system. The higher the weight, the more desirable the category is. These weights are currently chosen to obtain a conservative FCD VSL system, warning more proactively, in order to capture all situations in which the loop system also activates. Other weights can be specified according to user preferences.

Starting from the FCD of 14 January 2016, the FCD AID results were calculated along with the CS for different parameter combinations. In the optimisation procedure, the sample weighting factors α_{acc} and α_{dec} were varied from 0.1 to 0.6 in increments of 0.05. The upper threshold was varied from 40 to 55 km/h while the lower threshold was varied from 25 to 40 km/h, both in increments of 5 km/h. The influence range of the virtual detectors (500 m) was not varied as these were considered to correspond well to the current AID guidelines for loop placement on Dutch highways. Maximal correspondence was obtained for the presented use case for the parameters shown in Table 4.3. Comparing the values of the FCD VSL algorithm to those of the loop VSL (as set by the Dutch road operator), the upper and lower thresholds are the same. The α_{acc} is lower while α_{dec} is somewhat higher, indicating higher speeds are taken into account less while lower speeds more. This makes the average calculated by the virtual loop detectors react quicker to lower speeds, which results in going below the lower threshold quicker. It should however again be noted that optimising for correspondence aims at approximating the loop system. Due to its inherent technological properties, the loop system has its own advantages and limitations (e.g. spatial resolution, limited look-ahead), as well as the FCD system. Optimising for correspondence could lead to unwanted effects as the FCD system would overfit and try to optimise out some of its inherent advantages compared to the loop system. It does however show the potential of current FCD to generate VSL signage.

4.5 Evaluation

With the described evaluation above, the calibrated FCD VSL is compared to the loop VSL for an extended period of time. Table 4.4 shows the comparison for the weekdays between January 11 and January 22, 2016. Compared to the loop data, the FCD coverage varies between 6-8% of all traffic. For each day, the state categorisation from above was done, resulting in the relative percentages for each category. On top of the correspondence score (CS), the table also aggregates the different categories to overall percentages to show the similarity and conservative nature of both systems. The *Idle* state classification corresponds to the percentage that both systems were idle (category 1) while the *Identical* state denotes the portion that both systems were in the same 3-valued state from above (category 2). The *Conservative* classification represents the sum of the categories in which the FCD system was more cautious than the loop system (categories 3, 5, 7, 9 & 11). *Relaxed* denotes the opposite, with the loop system being more cautious (categories 4, 6, 8, 10 & 12). When the loop VSL is taken as the reference, the relaxed portion gives an indication of missed signs. Note however that the loop system itself has some disadvantages, as shown in Section 4.3.1.

Table 4.3: Optimal FCD parameters for conservative loop AID approximation for A27 highway on Thursday 14 January 2016, from 3 pm to 9 pm.

VSL PARAMETERS	**FCD**	**Loop**
α_{acc}	0.25	0.40
α_{dec}	0.25	0.15
lower threshold	35	35
upper threshold	50	50
Category		
1	44.4%	
2	36.9%	
3	5.4%	
4	0.8%	
5	3.4%	
6	0.3%	
7	2.3%	
8	0.3%	
9	3.3%	
10	0.2%	
11	2.3%	
12	0.4%	
Correspondence Score	0.8575	

Additionally, the table also sums the different categories but only taking into account the actual displayed sign for the drivers. This results in the *Identical* (categories 1, 2, 3, 4, 11 & 12), *Conservative* (categories 5, 7 & 9) and *Relaxed* (categories 6, 8 & 10) sign entries. These numbers give more insight into the perceived difference by an observer not taking into account temporal sign switching.

In the 2 weeks shown here, the 12th and 14th of January had severe congestion while Friday 15 January hardly had any congestion. Excluding these days, the average fraction idle states is 71.1%, with 15.9% identical, 11.7% more conservative and 1.4% more relaxed state classification. For the sign classification, the averaging of the 7 days yields 93.1% identical, 6.2% more conservative and 0.6% more relaxed signage. On the congested days, the VSL systems were active for more than half of the study window as indicated by the idle state percentages 46% and 44%. Compared to normal inactivity of 71.3%, the increase in the relaxed signs is limited as they only increased from 0.6% (average normal days) to 0.9% (average for the 2 congested days). This indicates that the increase in the relaxed classification is caused by the increased activity of the VSL systems. Looking at the non-congested Friday 15 January, the FCD VSL is only more relaxed in 0.1% of the window. Overall, the low fraction of the relaxed classification is due to the conservative calibration of the FCD system. While the numbers for the relaxed classification should not be minimised, it does indicate that the calibrated FCD VSL (using only 1 day) approximates the loop VSL and is a suitable alternative.

4.5.1 Real-time potential analysis

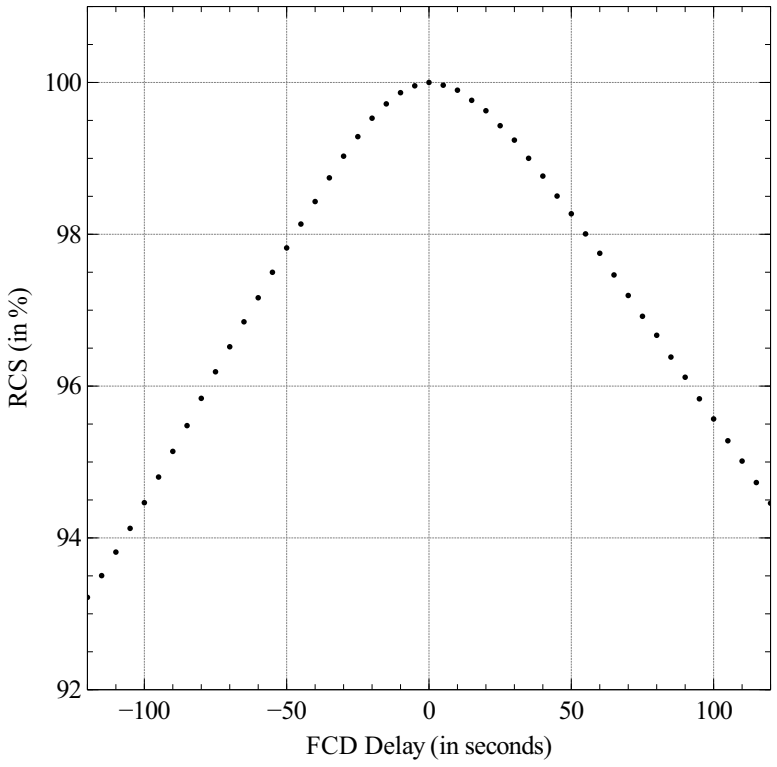
In the analysis above, all data was considered to be available instantaneously to the AID systems. In real applications, some delay will be present, mostly attributable to communication overhead and FCD transmission schemes. While the loop systems are operational, proving their feasibility, the tolerable latency for FCD is yet unknown. This section aims at investigating the effects of latency on the FCD VSL system.

To study the effect of latency, artificial delay is introduced in the FCD VSL algorithm. FCD samples are simulated to have a fixed communication overhead. While this latency in real life would be more random, it is sufficient here to show the performance impact. The FCD parameters in both cases are set to the optimal values from before. Delay was added in steps of 5 seconds. Fig. 4.7 shows the Relative Correspondence Score (RCS) for the A27. The RCS is the ratio of the CS for each specific latency to the maximum CS (when no FCD delay is present). Negative delay values (e.g. -60 s) correspond to situations where the loop data (used for the loop AID) would have delay before being accounted for in the overhead signs, as the CS is relative between both VSL systems. Interestingly, the conservative calibration of the FCD parameters results in a better tolerance for positive FCD

Table 4.4: Comparison FCD VSL to loop VSL for A27 highway for weekdays from Monday 11 January to Friday 22 January 2016, between 3 pm and 9 pm.

Date	**MON 11 Jan**	**TUES 12 Jan**	**WED 13 Jan**	**THU 14 Jan**	**FRI 15 Jan**
STATE					
Idle	72.7%	46.1%	70.0%	44.4%	95.0%
Identical	14.6%	36.0%	16.9%	36.9%	1.9%
Conservative	11.1%	15.7%	11.8%	16.7%	2.9%
Relaxed	1.6%	2.3%	1.3%	2.0%	0.2%
SIGN					
Identical	93.1%	90.7%	93.4%	90.1%	98.2%
Conservative	6.1%	8.3%	6.1%	9.0%	1.7%
Relaxed	0.8%	1.0%	0.6%	0.9%	0.1%
CS	0.8965	0.8586	0.8973	0.8575	0.9732
Date	**MON 18 Jan**	**TUES 19 Jan**	**WED 20 Jan**	**THU 21 Jan**	**FRI 22 Jan**
STATE					
Idle	77.9%	70.9%	67.9%	67.7%	70.4%
Identical	13.1%	16.0%	18.1%	17.3%	15.1%
Conservative	8.2%	11.7%	12.5%	13.5%	13.2%
Relaxed	0.9%	1.4%	1.4%	1.4%	1.4%
SIGN					
Identical	94.7%	93.2%	93.0%	92.1%	92.5%
Conservative	4.9%	6.0%	6.3%	7.3%	6.9%
Relaxed	0.4%	0.8%	0.8%	0.6%	0.6%
CS	0.9290	0.8975	0.8874	0.8884	0.8896

Figure 4.7 Latency simulation of FCD.



delay, with a graceful degradation of the RCS of 0.0055% per second of added latency, than for negative FCD delay, with degradation of 0.0065%. The conservative nature of the FCD calibration makes it warn earlier, lessening the impact of the FCD delay.

Note that the analysis here was done without optimising any FCD system parameter. In live systems, these parameters could be adjusted (based on a preliminary estimate of the experienced latency) to account for the expected latency. By tweaking the α_{dec} of the FCD VSL, it could be more proactive in alerting for congestion. Similarly, the influence range could be enlarged, to account for backwards congestion propagation. Furthermore, more specific approaches could be used in the FCD algorithm e.g. including the sample age (to account for variable latency) or taking into account downstream samples in the estimation for the current position. Specialised provisions for latency are beyond the scope of this paper as the focus here was to show the potential of FCD in terms of traffic estimation and VSL application.

4.6 Discussion

While the results above indicate the potential of FCD, some important remarks need to be made. Firstly, the algorithm was designed/calibrated to mimic the existing loop system, as this was the only available reference. While loop systems are widely used, they have known disadvantages e.g. localised view and prone to sensor errors. Aside from costly manual benchmarks and model-bound simulation results, no reference output exists. For this study, the operational system, having proven its value in the field, was considered a good target for FCD testing. Other VSL logs could however easily be integrated in the calibration/evaluation. This would also allow to use some FCD properties (e.g. time between samples or total number of samples) which were not included in the algorithm design to stick close to the reference. Fully utilising valuable FCD properties is left for future work.

Another point of interest is the used FCD. In our study, high-frequency FCD covering 6-8% of traffic was available. Compared to the existing literature (see introduction), this is a lot of data. The performance of the FCD algorithm with less data would decline. However, considering advances in vehicle communication and connectivity, cellular coverage and emerging tolling policies, data availability will only increase. By properly integrating our modern infrastructure (taking privacy into account), the penetration rates will improve, allowing for even better results. While results of our algorithm on smaller datasets would be interesting to determine the absolute required FCD penetration rate, the imperfect benchmark, varying desired performance and omitted algorithm optimisations render optimal parameters not sensible. Extensive optimisation studies are left for future work.

4.7 Conclusion

In this paper, we focused on the potential for Floating Car Data (FCD) to be used in dynamic traffic management systems. An FCD variable speed limiting (VSL) algorithm was designed similar to an existing loop VSL and evaluated by a state-based classification method. The FCD VSL was conservatively calibrated and compared to an existing loop VSL benchmark during afternoon rush hour for 2 work weeks. The FCD VSL followed loop signage throughout the study period, only not warning 0.6% of the time window on normal days and 0.9% on congested days. This live setup shows that current FCD system technology has acceptable latency and sufficient penetration rate to allow live VSL. With lower system costs, wider area coverage and ever-increasing vehicle connectivity, FCD presents a valuable alternative for loop data in dynamic traffic management systems.

Acknowledgements

Maarten Houbraken was a PhD fellow of the Research Foundation - Flanders (FWO-Vlaanderen). The authors would further like to thank the Dutch road operator Rijkswaterstaat for providing the data.

References

[1] M. Westerman, R. Litjens, and J.-P. Linnartz. *Integration Of Probe Vehicle And Induction Loop Data: Estimation Of Travel Times And Automatic Incident Detection*. Technical report, Institute of Transportation Studies, University of California, Berkeley, 1996.

[2] J. Kwon, B. Coifman, and P. Bickel. *Day-to-Day Travel Time Trends and Travel Time Prediction from Loop Detector Data*. Transportation Research Record, (1717):120–129, 2000.

[3] B. Coifman. *Estimating travel times and vehicle trajectories on freeways using dual loop detectors*. Transportation Research Part A: Policy and Practice, 36(4):351–364, 2002.

[4] B. Coifman. *Vehicle Re-Identification and Travel Time Measurement in Real-Time on Freeways Using Existing Loop Detector Infrastructure*. Transportation Research Record: Journal of the Transportation Research Board, 1643(July):181–191, jan 1998.

[5] J. Li, H. van Zuylen, and G. Wei. *Diagnosing and Interpolating Loop Detector Data Errors with Probe Vehicle Data*. Transportation Research Record: Journal of the Transportation Research Board, 2423:61–67, 2014.

[6] H. Bar-Gera. *Evaluation of a cellular phone-based system for measurements of traffic speeds and travel times: A case study from Israel*. Transportation Research Part C: Emerging Technologies, 15(6):380–391, dec 2007.

[7] D. GundlegaŁrd and J. Karlsson. *Handover location accuracy for travel time estimation in GSM and UMTS*. IET Intelligent Transport Systems, 3(1):87, 2009.

[8] S. Ancona, R. Stanica, and M. Fiore. *Performance boundaries of massive Floating Car Data offloading*. In 2014 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS), pages 89–96. IEEE, apr 2014.

[9] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. *Current map-matching algorithms for transport applications: State-of-the art and future research directions*. Transportation Research Part C: Emerging Technologies, 15(5):312–328, oct 2007.

[10] F. Chen, M. Shen, and Y. Tang. *Local Path Searching Based Map Matching Algorithm for Floating Car Data*. Procedia Environmental Sciences, 10(PART A):576–582, 2011.

[11] T. Miwa, D. Kiuchi, T. Yamamoto, and T. Morikawa. *Development of map matching algorithm for low frequency probe data*. Transportation Research Part C: Emerging Technologies, 22:132–145, 2012.

[12] B. Y. Chen, H. Yuan, Q. Li, W. H. Lam, S.-L. Shaw, and K. Yan. *Map-matching algorithm for large-scale low-frequency floating car data*. International Journal of Geographical Information Science, 28(1):22–38, 2014.

[13] E. Jenelius and H. N. Koutsopoulos. *Travel time estimation for urban road networks using low frequency probe vehicle data*. Transportation Research Part B: Methodological, 53:64–81, jul 2013.

[14] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. *Non-parametric estimation of route travel time distributions from low-frequency floating car data*. Transportation Research Part C: Emerging Technologies, 58:343–362, sep 2015.

[15] F. Zheng and H. Van Zuylen. *Urban link travel time estimation based on sparse probe vehicle data*. Transportation Research Part C: Emerging Technologies, 31(111):145–157, 2013.

[16] C. De Fabritiis, R. Ragona, and G. Valenti. *Traffic estimation and prediction based on real time floating car data*. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, (NOVEMBER 2008):197–203, 2008.

[17] S. Breitenberger, B. Grüber, M. Neuherz, and R. Kates. *Traffic information potential and necessary penetration rates*. Traffic Engineering and Control, 45(11):396–401, 2004.

[18] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen. *Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment*. Transportation Research Part C: Emerging Technologies, 18(4):568–583, aug 2010.

[19] M. Levin and G. M. Krause. *Incident detection: A Bayesian Approach*. Transportation Research Record, (682):52–58, 1978.

[20] A. I. Gall and F. L. Hall. *Distinguishing Between Incident Congestion and Recurrent Congestion: A Proposed Logic*. Transportation Research Record, (1232):1–8, 1989.

[21] E. Parkany, D. Bernstein, and E. Parkany. *Design of Incident Detection Algorithms Using Vehicle-to-Roadside Communication Sensors*. Transportation Research Record 1494, 67:67–74, 1995.

[22] K. Petty, A. Skabardonis, and P. Varaiya. *Incident Detection with Probe Vehicles: Performance, Infrastructure Requirements, and Feasibility*. IFAC Proceedings Volumes, 30(8):125–130, jun 1997.

[23] L. Yu, L. Yu, Y. Qi, J. Wang, and H. Wen. *Traffic Incident Detection Algorithm for Urban Expressways Based on Probe Vehicle Data*. Journal of Transportation Systems Engineering and Information Technology, 8(4):36–41, 2008.

[24] L. Yu, Y. Gao, L. Yu, G. Song, F. Zhang, and J. Liu. *Floating car data-based method for detecting flooding incident under grade separation bridges in Beijing*. IET Intelligent Transport Systems, 9(8):817–823, oct 2015.

[25] W. Vandenberghe, E. Vanhauwaert, S. Verbrugge, I. Moerman, and P. Demeester. *Feasibility of expanding traffic monitoring systems with floating car data technology*. IET Intelligent Transport Systems, 6(4):347–354, dec 2012.

[26] B. Kerner, C. Demir, R. Herrtwich, S. Klenov, H. Rehborn, M. Aleksi, and A. Haug. *Traffic state detection with floating car data in road networks*. In Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005., pages 700–705. IEEE, 2005.

[27] G. Klunder, H. Taale, and S. Hoogendoorn. *The Impact of Loop Detector Distance and Floating Car Data Penetration Rate on Queue Tail Warning*. In 3rd International Conference on Models and Technologies for Intelligent Transport Systems, 2013.

[28] L. Ambühl and M. Menendez. *Data fusion algorithm for macroscopic fundamental diagram estimation*. Transportation Research Part C: Emerging Technologies, 71:184–197, oct 2016. arXiv:j.trc.2016.07.013.

[29] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. *Floating Car and Camera Data Fusion for Non-Parametric Route Travel Time Estimation*. In 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), pages 1286–1291, 2014.

[30] M. Houbraken, P. Audenaert, D. Colle, M. Pickavet, K. Scheerlinck, I. Yperman, and S. Logghe. *Real-time traffic monitoring by fusing floating car data with stationary detector data*. In 2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2015, 2015.

[31] C.-s. Chou and A. P. Nichols. *Deriving a surrogate safety measure for freeway incidents based on predicted end-of-queue properties*. IET Intelligent Transport Systems, 9(1):22–29, feb 2015.

"Have you ever noticed that anybody driving slower than you is an idiot, and anyone going faster than you is a maniac?"

— George Carlin (1937 - 2008)

5

Automated Incident Detection using Real-Time Floating Car Data

Chapter 4 investigated the feasibility of a Variable Speed Limit (VSL) system working on FCD. Following those positive results, the entire setup was further developed and integrated in an operational live platform to test the performance under realistic conditions (e.g. delay, noise,...). This chapter presents the results of that study.

⋆ ⋆ ⋆

Maarten Houbraken, Steven Logghe, Marco Schreuder, Pieter Audenaert, Didier Colle, Mario Pickavet

Published in the Journal of Advanced Transportation, December 2017.

Abstract The aim of this paper is to demonstrate the feasibility of a live Automated Incident Detection (AID) system using only Floating Car Data (FCD) in one of the first large-scale FCD AID field trials. AID systems detect traffic events and alert upcoming drivers to improve traffic safety without human monitoring. These automated systems traditionally rely on traffic monitoring sensors embedded in the road. FCD allows for a finer spatial granularity in the traffic monitoring. However,

low penetration rates of FCD probe vehicles and the data latency have historically hindered FCD AID deployment. We use a live countrywide FCD system monitoring an estimated 5.93% of all vehicles. An FCD AID system is presented and compared to the installed AID system (using loop sensor data) on 2 different highways in the Netherlands. Our results show the FCD AID can adequately monitor changing traffic conditions and follow the AID benchmark. The presented FCD AID is integrated with the road operator systems as part of an innovation project, making this, to the best of our knowledge, the first full chain technical feasibility trial of an FCD-only AID system. Additionally, FCD allows for AID on roads without installed sensors, allowing road safety improvements at low cost.

5.1 Introduction

Traditionally, traffic monitoring has largely relied on dedicated roadside equipment. Sensor equipment such as inductive loop detectors, wireless signal receivers or cameras were installed on or embedded in the road surface to detect all individual vehicles passing the specific location. This yields a trove of traffic data to be used in traffic state estimation, travel time measurements and traffic management applications. While loops only monitor a single location, vehicle re-identification techniques can be applied to obtain vehicle trajectory information, allowing for various application e.g. estimating travel times [1, 2], highway monitoring [3] and incident detection [4–6]. A complete system of connected measurement locations can be installed, allowing road operators to monitor traffic. However, total system costs (see [7] for a selection of installed ITS systems and costs) scale linearly with the covered area size. While the actual loop hardware is relatively inexpensive, the installation costs (e.g. highways closures, road surface cuts, power supply) and maintenance (e.g. monitoring loop status, closing individual lanes, repairing broken network cables) greatly impact the total cost during the full system lifespan. While loop data quality has improved over the past years, loop systems are still prone to malfunction. In [8], diagnostics showed 31% of loop sensors in a Chinese city to be defect and 25% of the remaining sensors reporting errors $> 20\%$ compared to visual counts. This indicates the need for better data.

With mobile devices becoming prevalent in the past few years, an invaluable source of traffic data has become available termed Floating Car Data (FCD). The FCD itself consists of measurements coming from individually tracked probe vehicles reporting their location and speed. The data generally consists of timestamped positional information, along with the instantaneous vehicle speed. As vehicles are tracked on their route, the data is, in contrast to loop detector data, inherently spatially distributed. This results in a completely different FCD system cost model. While the hardware cost of an FCD system is limited to a few central servers, getting (or buying) live data from probe vehicles becomes the main issue. However,

the data can be sourced from other applications e.g. existing track & trace platforms or mandatory tolling systems, thus sharing the costs, and additional (profitable) services using the same data can be deployed (e.g. insurance). As more and more cars are getting connected, the amount of data will only increase, allowing for large scale FCD AID deployment at a fraction of loop-based system costs.

Additionally, the FCD differs from the loop detector data in terms of total vehicle coverage, as it only monitors a limited sample of the total traffic. Comparing FCD with loop detector data is focused on in [9], noting statistical differences in calculated travel times which need to be accounted for. The sampling rate and penetration coverage are studied in [10], showing the suitability of current FCD techniques in data fusion.

While most research is focused on traffic state and travel time estimation, FCD can be used for a wide range of applications like traffic policy evaluation [11] or road network mapping/generation [12]. Another closely related application is Automated Incident Detection (AID) in which traffic is monitored using live loop detectors, FCD and other sources to obtain a view on traffic conditions. They try to avoid congestion and collisions by activating overhead speed restriction signs in order to homogenise traffic and alert drivers for upcoming traffic events. This improves traffic by improving congestion recovery and safety by avoiding end-of-queue collisions [13]. Combining even small amounts of FCD with existing loop-based systems greatly increases performance [14]. Various data fusion algorithms have also been developed, focusing on combining FCD with roadside detector data. These algorithms combine the information in both sources to better estimate traffic variables [15–17]. However, this paper focuses on pure FCD data, a more comprehensive review of data fusion algorithms can be found in [18].

While FCD can definitely contribute to existing loop-based systems, it could also be directly used to feed incident detection algorithms. In [19], 2 AID algorithms (1 using probe travel times and 1 using shockwave theory) are presented using a simulated AIMSUM FCD dataset. Simulated FCD studies typically model the underlying network and fit the model to target traffic volumes, supply-demand matrices or loop detector measurements. The calibrated model is then sampled to obtain FCD to be used in experiments. This approach allows to vary the penetration rates and test FCD algorithms for performance. A traffic state classification AID is presented in [20] using simulated FCD generated by SUMO, calibrated using an unspecified set of real FCD traces near Pisa, Italy. They note the lack of real FCD the main reason for simulation. Traffic state estimation and AID was tackled together in [21] using a CORSIM simulation in a hybrid state estimation problem. However, all these results are still dependent on model parameters and assumptions.

Studies using real-time FCD, captured from actual vehicles driving on the studied routes, do not have model assumptions but large FCD systems are still being

rolled out. However, several field systems have been successfully deployed. The Mobile Century project [22] gives the results for 100 dedicated vehicles driving loops on a highway, giving an estimated 2-3% needed penetration rate. In Stockholm, a fleet of 1500 taxis has been used to estimate route travel times and travel time distributions in [23] and [24], focusing on eliminating inherent biases. Several comparable taxi FCD systems have also been deployed in Berlin (200 taxis), Vienna (400) and Nuremburg (500) in [25]. A real-time AID system using an underlying state classification for traffic on an hourly basis was reported in [26], although no specific information is given on the actual real-time detection time or FCD coverage. However, most of these systems are limited to cities.

To eliminate the model influences and city-limited FCD properties mentioned above, this paper presents the results of an FCD AID algorithm working solely on FCD from a countrywide system covering around 6% of all traffic with anonymised data being collected from a free transportation app. Data from this system was used to feed a real-time FCD AID algorithm. To further prove the operational feasibility, the output of the algorithm was transmitted real-time to Dutch road operators and integrated in their monitoring setup, contrary to most literature on FCD limited to local test setups. The main focus of this study is to examine the current feasibility of FCD systems to provide live signage for large area AID systems, taking into account technical issues and coupling back to existing infrastructure. By establishing an entire operational FCD AID chain, our study provides unique and quantifiable results on current day FCD AID feasibility.

The rest of this paper is structured as follows. Section 5.2 gives details on the installed loop AID algorithm and benchmark while Section 5.3 shows the FCD modifications to the existing algorithm. Section 5.4 elaborates on the evaluation methodology used in this paper, consisting of a state-based classification of both the loop-based AID and the FCD AID. Section 5.5 gives the performance results of the FCD AID on 2 Dutch highways, delving deeper into the FCD AID parameters and the impact of delay. In closure, Section 5.6 presents a discussion on the results of the live tests.

5.2 Existing monitoring & management

In this study, we focus on 2 different highways, both currently equipped with the same automated incident detection system (AID) based on roadside sensors (inductive loop detectors). This current system is to be compared to the FCD version. The AID system focuses on detecting congestion and other traffic incidents as quickly as possible and warns upstream traffic by using dynamic overhead speed advice signs. This allows incoming traffic to reduce their speed and avoid rear-end collisions. We first describe the AID system here.

The installed AID system monitors the highways using the loop data sensor embedded in the road every few hundred metres (typically 500 m). These loop data sensors are coupled to overhead speed signs. The speed sign controller keeps a running average of the individual vehicle speeds registered by each loop data sensor. The speed v_{cur} of the loop data sensor is updated when a car passes the sensor. When a car passes with speed v_s lower than v_{cur}, this indicates traffic slowing down and v_{cur} is changed to $(1 - \alpha_{dec}) \cdot v_{cur} + \alpha_{dec} \cdot v_s$. If $v_s > v_{cur}$, the update is done analogously but with a different weighting factor: $(1 - \alpha_{acc}) \cdot v_{cur} + \alpha_{acc} \cdot v_s$. This update process aims at quickly reacting to traffic slowing down (high α_{dec} = dropping quickly) while only gradually raising the average when the congestion is dissolving (lower α_{acc}). The α_{acc} and α_{dec} are limited to the [0, 1] range and can be modified by road operators to tune the system performance for the specific location.

The averages calculated by the loop sensors are monitored to detect congestion incidents. An incident is reported when v_{cur} of a specific location drops below a predefined threshold v_{ON}, which triggers an AID ON message and activates the overhead sign. When traffic conditions improve and the v_{cur} rises above the v_{OFF} threshold, it triggers an AID OFF message to deactivate the sign and end the incident. The thresholds can be set differently to avoid excessive switching of the road sign. Fig. 5.1 shows this process for a local jam on the A58.

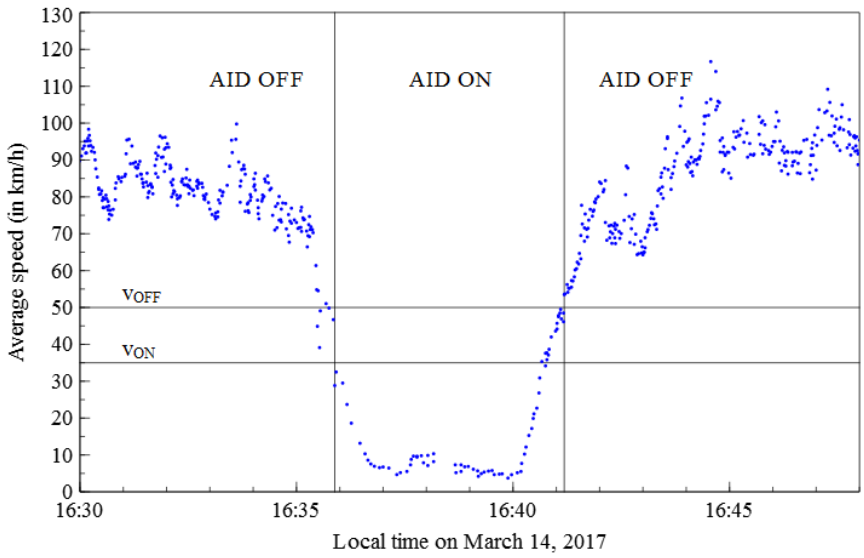
The AID ON message triggers the speed signs of all portals within a configurable range termed look-ahead distance *LAD*. This trigger overrules the normal maximum allowed speed (typically 130 km/h on the Dutch highways). While this behaviour is configured by the road operator for each specific location, for the roads in this study, the AID ON triggers a speed sign display of 50 km/h for all speed signs within a range of 700 m. With loop sensors and portals spaced approximately 500 m apart, generally 2 speed signs are activated. To avoid abrupt speed changes, a sign of 70 km/h is also triggered upstream of the first 50 km/h sign. To further draw attention to the speed limits changes, flashing lights accompany the speed signs when they differ from the limit upstream.

5.3 FCD extensions

The AID system from the previous section was developed throughout the 1970s and 1980s and further tuned in the past decades. While this system has proven its efficacy over time, it is still limited by its input (loop sensor) data, only available on specific roads (about a quarter of all Dutch motorways). By implementing this system using FCD and new and/or cheaper signage systems (e.g. in-car delivery, simpler roadside signs), its application area is greatly increased.

FCD consists of individual vehicle samples generated by tracked probe vehicles. Each sample consists of a pair of coordinates, an anonymised vehicle identi-

Figure 5.1 AID switching. The AID switches on and off based on the average speed of the loop sensor. The blue dots show the average of all samples across all lanes. The AID goes on when the average drops beneath the v_{ON} threshold just before 16:36 and goes off when traffic restores after 16:41.



fier, a timestamp, vehicle heading and a measured speed estimate v_s. The samples are matched to an underlying representation of the road network with segments of at most 50 m. This fine granularity allows to monitor traffic variances on a detailed level without the need for extensive trip generation to determine intermediate segments between successive probes.

In the FCD system of this study, samples are generated every second by the vehicle probe and sent to the central server every 10 s for further processing by the AID software. As the measured samples are subject to measurement noise, the FCD system applies several filters to reduce errors. First, erroneous measurements (e.g. missing fields, inconsistent data) and irrelevant samples (e.g. outside of study area) are discarded. For each remaining sample, the nearest road segments in the underlying network are identified and filtered (taking into account vehicle heading and road type) to obtain a segment match for the sample. For new vehicles, a new trip is then generated starting from this segment while for known vehicles, their trip is extended with the new segment. This creates/updates the vehicle trip history and helps to further reduce measurement noise as only valid trips are considered by the AID algorithm. A sample is only taken into account when the vehicle trip is guaranteed to be on the route. This eliminates e.g. vehicle samples from neighbouring roads for which the trip segment history is inconsistent (e.g. consecutive

segments are not connected in the network). It also omits samples from vehicles coming from entry ramps (as they have not yet been on the highway long enough) which are undesirable as they are still accelerating.

Using the trip samples from the validated vehicles, the FCD AID is implemented analogously as the loop AID. On each segment of the monitored routes, virtual sensors are defined in software. While the loop AID has physical sensors every 500 m on average, the FCD AID has simulated sensors on each segment of the underlying road network, spacing sensors roughly every 50 m. When valid vehicle probe samples are received, the current average speed v_{cur} on the mapped segment is updated analogously to the loop AID system (taking into account weighting factors according to the speed of the sample, see Section 5.2). This update process is event-driven, with averages being calculated as soon as the sample is received.

The calculated averages are then used (as in the loop AID) to trigger AID ON/OFF messages and incident reports. Each virtual sensor will trigger a congestion notification when it goes below the lower speed threshold v_{ON} and a free-flow notification when the average speed restores to above the upper speed limit threshold v_{OFF}. These notifications are monitored by virtual FCD speed signs, which are modelled on the locations where the physical loops are installed along the route. Each virtual FCD speed sign monitors the virtual FCD sensors in a configurable downstream range defined as the look-ahead distance *LAD*, typically 500-1,000 m. As the virtual sensors are only 50 m apart, each individual virtual speed sign is influenced by multiple virtual sensors, typically 10-20 assuming segments of 50 m. This fine granularity allows for finer signage and more smooth control. The virtual speed sign will trigger an AID ON message when one of its associated segments detects congestion. Note that congestion will be detected when the v_{cur} drops below the v_{ON} threshold. Due to α_{dec} smoothing the v_{cur} decline, this requires multiple samples, effectively ensuring that a single erroneous sample can't trigger a congestion event. When congestion dissipates, the AID OFF message is sent when all of the associated congested virtual sensors have reported free-flow again, similar to the loop AID. The created AID messages are sent in real-time to the local highway operator to be shown on the physical sign infrastructure. Note that the preceding 70 km/h advice of the loop AID is also replicated here but it is not taken into account for this analysis.

5.4 Evaluation methodology

To compare both AID systems and investigate the effect of FCD system delay, the FCD AID is benchmarked against a ground truth constructed out of the loop AID. While the FCD AID inherently covers the entire road network and allows for signage on every road (segment), the evaluation here is limited to the locations

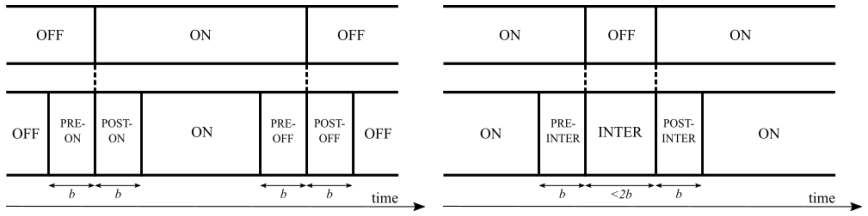
equipped with loop sensors. This favours the loop AID as their performance is best at those locations (in contrast to unmonitored locations between loops or on unequipped highways). Additionally, as the loop AID (with its shortcomings) is taken as the ground truth, the FCD AID is limited to mimicking and penalised for each difference, thereby limiting possible improvements. However, the loop AID is the best automated benchmark available, with realistic traffic conditions free of model assumptions.

The aim of the analysis is to determine if the input FCD AID is able to match the performance of the loop AID. Delay is considered the most important issue here, as alerting drivers needs to be done as soon as possible. The loop AID has little delay as the data measurements (= individual vehicles passing a loop sensor) are directly coupled to the sign controller at the road location itself. The FCD however requires several steps, introducing delay. First of all, it is dependent on the vehicle polling frequency. While vehicles measure their location at fixed intervals, transmitting the data is done in batches (grouping several GPS samples in one communication). This lowers the battery usage of the vehicle probes but adds buffering delay. If a vehicle would only transmit its data once every minute, this would introduce an average delay of 30 s. Next, communicating the data to the central sign controller over the public communication network also introduces a transmission delay of several seconds. Additionally, as only a small subset of all vehicles is sampled, extra sampling delay is accrued as the FCD AID needs to wait for a monitored vehicle to pass the location under study. The time between samples can vary significantly e.g. if there is very little traffic at night, hardly any samples will be available. In general, the delay is coupled to the penetration rate (as twice as many probes roughly halves the sampling delay). In our study, the delay was in the order of a couple of seconds to a few minutes. Finally, processing the data of a countrywide network requires some time (to filter out the relevant messages from the erroneous or irrelevant messages, mapping samples to road segments), adding processing delay (in the order of seconds) before being able to transmit instructions to an AID sign. The main focus of this paper is to determine if the FCD AID can overcome these delays.

To get a clear view on the FCD AID behaviour at AID change times for a single location, we construct a ground truth out of the benchmark AID consisting of several states. By calculating the time spent by the FCD AID in each of the different states, we can derive the high level behaviour of the AID.

Throughout the evaluation study period, the loop AID switches ON and OFF. To zoom in on the behaviour at switching times, we set a predefined buffer b (of e.g. 60 s) around each state change. When the loop AID switches from OFF to ON at time t_1, the period $[t_1 - b, t_1]$ is the PRE-ON state. Analogously, the period $[t_1, t_1 + b]$ is the POST-ON. The period $[t_1 + b, t_2 - b]$ is defined as the ON state (with t_2 the time at which the AID would switch back OFF), corresponding to a

Figure 5.2 State classification. The loop benchmark states are converted to a ground truth classification. Fig. a on the left denotes the buffering near normal ON/OFF switches, Fig. b on the right shows what happens when the loop switches $ON \rightarrow OFF \rightarrow ON$ in a short time window.

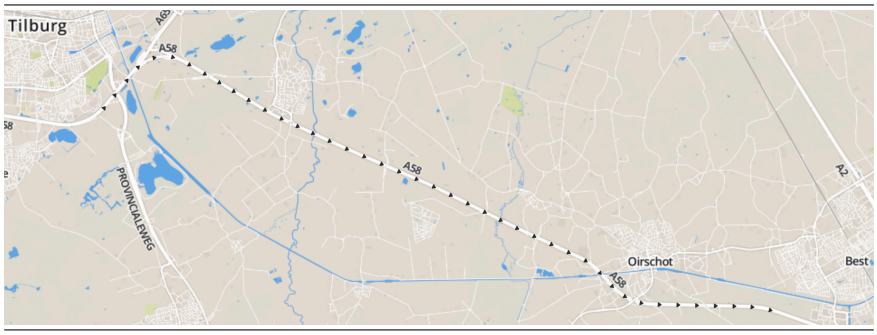


state when the AID should clearly be on. This classification is shown in Fig. 5.2a. When the FCD AID would switch on too early, it would spend time in the PRE-ON period. Analogously, we can define the PRE-OFF and POST-OFF states at the end of the loop AID event (t_2) and OFF for all time instants not near an AID event. With the state differentiation between PRE-ON and OFF, it is clear to see when the FCD AID is more pro-active in warning (by switching on earlier) instead of just wrong.

When AID changes occur rapidly, it is hard to judge if the AID should be on or off, given that the one system might still be on for the first event while the other is already on for the next event. With an imperfect benchmark, this is even harder. To mediate this problem, we group individual loop AID ON periods if they are close together in time. If 2 events are within $2b$, they are considered to be 1 big event. The time in between consists of an INTER state. The period before and after the gap (b s) is defined as PRE-INTER and POST-INTER. This mechanism is shown in Fig. 5.2b.

This classification yields 9 states covering the entire day/study period. The AID system under study can now be evaluated by calculating the time spent in every state, taking into account the loop AID result. This is done in Tables 5.2 and 5.3. When the loop AID system under study would be compared to itself, it would always score maximal for the ON or OFF state and 0 for the other. The total time per column is fixed for all evaluated AID systems. The FCD AID (and potential others) would differ from the loop AID by having non-zero values at the zero-positions (e.g. 10% of the max) and having non-maximal values at the other positions (e.g. 90%). These denote false positives (= indicating ON during the OFF state) and false negatives (= indicating OFF during ON) but also soft false positives (e.g. showing ON during PRE-ON and POST-OFF) and soft false negatives (e.g. showing OFF during POST-ON and PRE-OFF). The raw figures themselves indicate the time per state and can be used to characterise the system.

Figure 5.3 A58 test route running from Tilburg to Eindhoven, covering 19 km.



5.5 Results

5.5.1 Experimental setup

To study the FCD AID system, it was deployed on 2 different highway stretches in the Netherlands. The first test route, shown in Fig. 5.3, is the eastbound A58 running from Tilburg to Eindhoven between highway kilometre markers 35 and 16, spanning 19 km. On this route, 33 locations are equipped with loop sensors and AID overhead signs on which the evaluation will be focused. The route itself typically has a morning and evening rush hour peak during which traffic jams arise (at highway on/off ramps) that propagate upstream along the highway.

The second route, shown in Fig. 5.4, is the southbound A27 running from Utrecht to Gorinchem between highway kilometre markers 72 and 35 (spanning 37 km) with 61 AID equipped measurement locations.

The FCD used in this experiment was obtained from a commercial company monitoring the entire Dutch road network. The FCD penetration rate was determined for the A58 by counting all vehicles on an individual segment of both the FCD and the loop detectors between March 13 and March 17 resulting in an average weekday coverage of 5.93%. As mentioned above, samples were logged on device every second but only transmitted to the server every 10 s. Comparing the timestamp of the samples to the received timestamp, an average 2 s communication delay was found before the AID software received the data. The AID calculation was implemented in Go, mapping probe samples to road segments, maintaining speed averages and creating AID messages, adding on average another 2 s processing delay. After AID calculation, the results are transmitted to the Dutch road side traffic operators.

Figure 5.4 A27 test route running from Utrecht (north) to Gorinchem (south), covering 37 km.

To examine the effects of the individual AID parameters, 8 different configurations (see Table 5.1) were selected and deployed in parallel using the live FCD system. The α_{acc} and speed thresholds were kept constant as initial experiments indicated good correspondence to the benchmark. Having different speed thresholds as the ground truth, resulted in large discrepancies when traffic conditions evolved slowly. The look-ahead distance *LAD* and α_{dec} were varied to investigate more rapid ON-switching. Note that the main intent in this paper is to investigate the feasibility of the FCD AID system. Deriving optimal parameter configurations was not focused on as these values vary depending on the desired switching behaviour by the road operator. Furthermore, as with loop AID, the parameters can be varied along the route, depending on local road properties. To derive optimal (local) parameter configurations, many more configurations should be run, but this is out of scope for the limited system setup. Also note that the loop AID benchmark should not be too strictly followed as it has its own disadvantages (see Section 5.6).

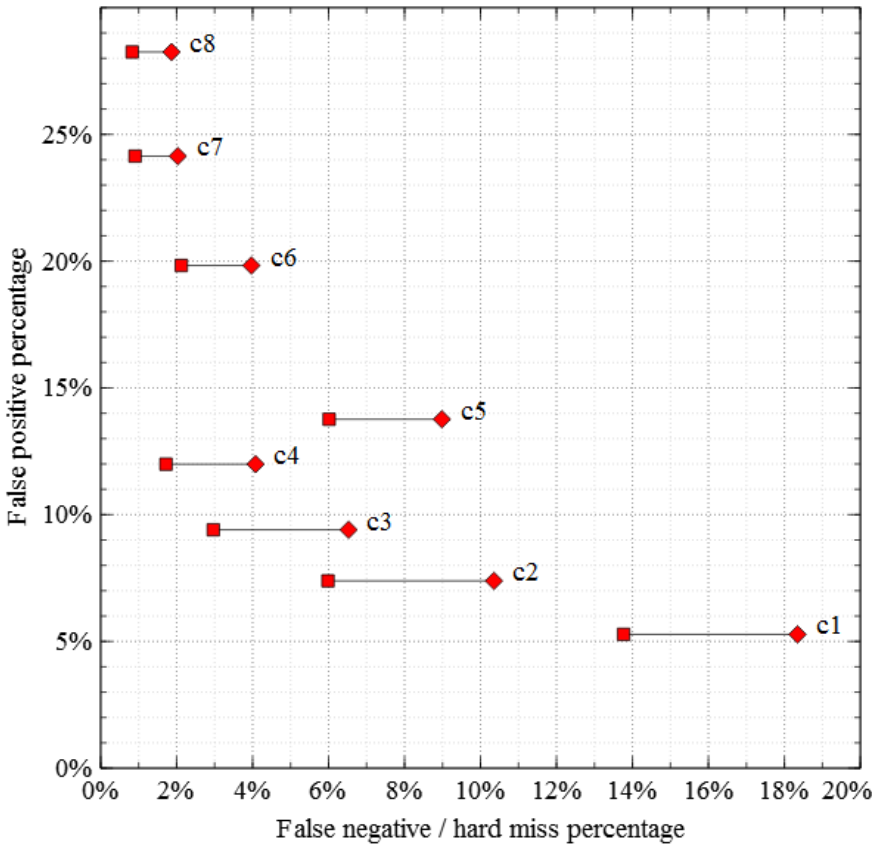
Table 5.1: Experimental FCD AID configurations deployed on the test routes.

CONFIG	α_{acc}	α_{dec}	v_{ON} (km/h)	v_{OFF} (km/h)	*LAD* (m)
c_1	0.4	0.3	35	45	750
c_2	0.4	0.4	35	45	750
c_3	0.4	0.5	35	45	750
c_4	0.4	0.5	35	45	900
c_5	0.4	0.3	35	45	1,250
c_6	0.4	0.4	35	45	1,250
c_7	0.4	0.5	35	45	1,250
c_8	0.4	0.5	35	45	1,400

5.5.2 FCD AID configuration results

The different FCD AID setups were compared to the installed loop-based AID logs provided by the Dutch highway operator for 4 weeks (1-28 March 2017, excluding 6th and 7th March for maintenance). The loop-based AID logs were categorised as described in Section 5.4 and used as the ground truth for the AID FCD. Note that the automated installed AID system can be manually overruled by road operators. These overrides were identified and removed from the analysis as we are only interested in the automated operation of the system. Tables 5.2 and 5.3 show the overall performance of the different setups. All percentages are relative to the total time the loop-based system is active (= everything except the OFF period state). Excluding the OFF periods (typically during the night) gives a clearer picture of the relevant active periods. The false positive percentages FP were calculated as the ratio of the time the FCD system triggers ON while the loop-based system shows

Figure 5.5 Trade-off between false positives and false negatives on the A58. Diamonds indicate the false negative percentages while the squares denote the hard miss percentages.



OFF to the active time. The false negative percentage FN denotes the percentage of active time the FCD AID is OFF while the loop AID is ON. Additionally, the 'hard misses' HM are also reported, denoting the percentage of time the loop-based AID is ON while the FCD AID is OFF and will stay OFF for more than 60 s. This represent the instances in which the FCD AID misses a traffic jam reported by the loop-based system. Fig. 5.5 and 5.6 show the trade-off between the false positives and the false negatives, along with the hard misses.

With these numbers, the difference in traffic conditions between both test routes can be quantified. While the A27 is almost twice as long as the A58 route, it is also more impacted by traffic jams. On average, a loop AID sign on the A27 was active 3.58% of each day (3,095 s) while an A58 loop AID sign was only active

Table 5.2: AID FCD scores per configuration on the A58. Reported percentages are all relative to the active time of the loop AID system. Values in green correspond to the benchmark state. Absolute numbers represent the total time (in seconds) all detectors on the studied route throughout the study period spent in that state. The (OFF, OFF) state results were reported in absolute numbers as it is not part of the active time.

		ON	OFF	PRE-ON	PRE-OFF	POST-ON	POST-OFF	INTER	PRE-INTER	POST-INTER	Total	
LOOP	ON	55.57%	0	0	8.44%	8.27%	0	0	3.06%	3.18%	1,600,843	ON
	OFF	0	6.72E+07	9.23%	0	0	9.31%	2.95%	0	0	67,594,192	OFF
											2,038,913	active
c_1	ON	48.91%	0.97%	0.46%	5.81%	1.85%	2.65%	1.24%	1.88%	1.71%	5.278%	FP
	OFF	6.66%	6.71E+07	8.77%	2.63%	6.42%	6.66%	1.71%	1.18%	1.46%	18.345%	FN
											13.77%	HM
c_2	ON	53.01%	1.37%	0.90%	6.89%	3.54%	3.41%	1.72%	2.41%	2.31%	7.40%	FP
	OFF	2.57%	6.71E+07	8.33%	1.55%	4.73%	5.90%	1.23%	0.65%	0.86%	10.36%	FN
											5.98%	HM
c_3	ON	54.45%	1.95%	1.37%	7.43%	4.79%	4.07%	2.01%	2.70%	2.63%	9.40%	FP
	OFF	1.13%	6.71E+07	7.86%	1.01%	3.48%	5.24%	0.94%	0.36%	0.54%	6.52%	FN
											2.97%	HM
c_4	ON	55.02%	2.63%	2.92%	7.54%	6.19%	4.20%	2.24%	2.80%	2.89%	11.99%	FP
	OFF	0.56%	6.71E+07	6.31%	0.90%	2.08%	5.11%	0.71%	0.26%	0.28%	4.08%	FN
											1.72%	HM

Table 5.2: AID FCD scores per configuration on the A58 (continued).

		ON	OFF	PRE-ON	PRE-OFF	POST-ON	POST-OFF	INTER	PRE-INTER	POST-INTER	Total	
c_5	ON	53.18%	4.67%	4.21%	6.02%	5.50%	2.90%	1.99%	2.34%	2.50%	13.77%	FP
	OFF	2.39%	6.71E+07	5.02%	2.42%	2.77%	6.41%	0.96%	0.72%	0.68%	8.98%	FN
											6.02%	HM
c_6	ON	54.91%	7.70%	5.99%	7.10%	6.91%	3.71%	2.42%	2.73%	2.89%	19.82%	FP
	OFF	0.67%	6.70E+07	3.24%	1.34%	1.35%	5.59%	0.53%	0.33%	0.28%	3.97%	FN
											2.12%	HM
c_7	ON	55.38%	10.21%	6.88%	7.61%	7.53%	4.42%	2.63%	2.91%	3.05%	24.14%	FP
	OFF	0.20%	6.70E+07	2.35%	0.83%	0.74%	4.89%	0.32%	0.15%	0.13%	2.05%	FN
											0.91%	HM
c_8	ON	55.40%	13.74%	7.27%	7.64%	7.61%	4.58%	2.66%	2.93%	3.06%	28.25%	FP
	OFF	0.17%	6.69E+07	1.96%	0.80%	0.65%	4.73%	0.29%	0.13%	0.12%	1.87%	FN
											0.83%	HM

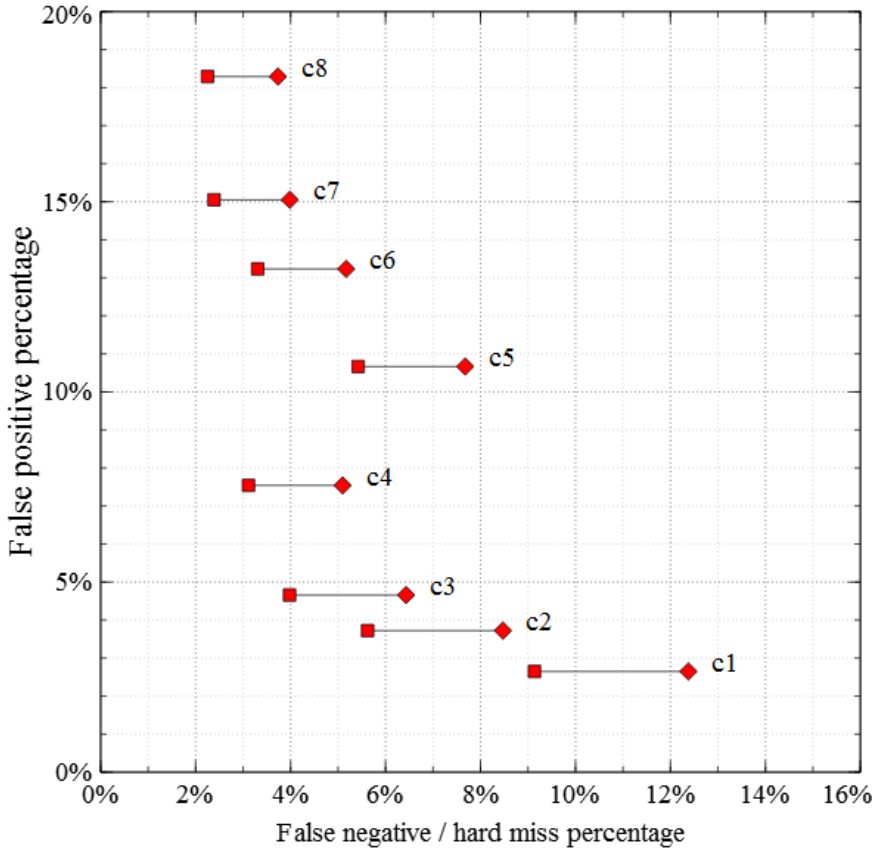
Table 5.3: AID FCD scores per configuration on the A27. Same definitions as for Table 5.2 were used.

		ON	OFF	PRE-ON	PRE-OFF	POST-ON	POST-OFF	INTER	PRE-INTER	POST-INTER	Total	
LOOP	ON	75.70%	0	0	4.35%	4.32%	0	0	2.01%	2.04%	3,545,188	ON
	OFF	0	9.49E+07	4.74%	0	0	4.74%	2.11%	0	0	95,402,814	OFF
											4,009,465	active
c_1	ON	69.66%	0.68%	0.56%	2.35%	1.72%	0.72%	0.68%	1.23%	1.08%	2.65%	FP
	OFF	6.04%	9.49E+07	4.17%	2.00%	2.60%	4.02%	1.43%	0.78%	0.96%	12.38%	FN
											9.14%	HM
c_2	ON	71.97%	0.98%	0.95%	2.68%	2.49%	0.92%	0.88%	1.41%	1.40%	3.72%	FP
	OFF	3.73%	9.49E+07	3.78%	1.67%	1.83%	3.82%	1.23%	0.60%	0.64%	8.47%	FN
											5.62%	HM
c_3	ON	73.00%	1.30%	1.24%	2.89%	2.97%	1.09%	1.03%	1.54%	1.59%	4.66%	FP
	OFF	2.70%	9.49E+07	3.50%	1.46%	1.35%	3.65%	1.08%	0.47%	0.45%	6.43%	FN
											3.98%	HM
c_4	ON	73.61%	2.75%	2.25%	2.96%	3.43%	1.25%	1.30%	1.60%	1.73%	7.54%	FP
	OFF	2.09%	9.48E+07	2.49%	1.39%	0.89%	3.49%	0.81%	0.41%	0.31%	5.10%	FN
											3.11%	HM

Table 5.3: AID FCD scores per configuration on the A27 (continued).

		ON	OFF	PRE-ON	PRE-OFF	POST-ON	POST-OFF	INTER	PRE-INTER	POST-INTER	Total	
c_5	ON	72.26%	5.66%	2.54%	2.55%	3.00%	1.17%	1.30%	1.44%	1.50%	10.67%	FP
	OFF	3.44%	9.52E+07	2.19%	1.80%	1.32%	3.57%	0.81%	0.57%	0.54%	7.68%	FN
											5.42%	HM
c_6	ON	73.60%	7.26%	3.06%	2.87%	3.47%	1.44%	1.48%	1.60%	1.71%	13.24%	FP
	OFF	2.10%	9.52E+07	1.67%	1.48%	0.86%	3.30%	0.63%	0.41%	0.33%	5.17%	FN
											3.31%	HM
c_7	ON	74.15%	8.50%	3.35%	3.07%	3.70%	1.64%	1.57%	1.70%	1.81%	15.05%	FP
	OFF	1.55%	9.51E+07	1.39%	1.28%	0.62%	3.10%	0.54%	0.31%	0.23%	3.98%	FN
											2.39%	HM
c_8	ON	74.26%	11.36%	3.49%	3.13%	3.75%	1.83%	1.62%	1.73%	1.83%	18.29%	FP
	OFF	1.44%	9.50E+07	1.25%	1.22%	0.58%	2.91%	0.49%	0.29%	0.21%	3.73%	FN
											2.25%	HM

Figure 5.6 Trade-off between false positives and false negatives on the A27. Diamonds indicate the false negative percentages while the squares denote the hard miss percentages.



2.31% (1,999 s). While the A27 has several intersections (with other highways) causing turbulence and congestion near on- and off-ramps, the A58 is less congested, largely consisting of backward propagating traffic jams. As indicated by the hard miss percentages, FCD AID performs slightly better for the A58 with a more predictable traffic pattern than for the A27, with its more dynamic traffic. Looking at configuration c_4, the hard miss percentages for the A58 can be reduced to below 2%, while most configurations for the A27 stay above 3%. The false negative percentages of 4.08% resp. 5.10% for the A58 and A27 roughly translate to a period of 82 resp. 158 seconds per day an FCD AID overhead sign is not showing while the loop AID would be.

Overall, the figures show the typical trade-off between triggering more (having more false positives and less false negatives) and triggering less (resulting in less false positives and more false negatives). As shown by the plotted hard misses, on average 46% of the false negatives on the A58 are soft misses (35% on the A27), resulting from the FCD AID switching on later than the loop AID. To counter this delay and obtain the desired recall rate (as deemed necessary by the road operator), the configuration parameters can be tuned. Increasing the look-ahead distance results in more virtual FCD AID loop sensors being taken into account and therefore more AID ON triggers. While the loop AID has an average inter-loop distance of 700 m, the look-ahead distances in the experimental setup are higher, as the FCD AID needs to counter the inherent delays (predominantly sampling and buffering delay). By having a higher *LAD*, backwards propagating traffic jams are more adequately detected. Having a higher α_{dec} also reduces false negative rates, as low speed samples impact the average speed more, reaching the lower threshold quicker. Having too high α_{dec} however leads to overweighting of individual samples, attributing too much importance to individual probe vehicles.

5.5.3 Full state comparison

While the desired trade-off is dependent on the intended use case, configuration c_4 is chosen here to further investigate, corresponding to a use case where both false positives and false negatives are to be minimised, but with a higher focus on false negatives (for safety). Table 5.4 shows a complete comparison between the loop AID and FCD AID on the A58 with the FCD AID also being categorised in 9 states as done for the loop AID. Table 5.5 shows the same for the A27 test case. This breakdown better displays the different contributing factors to the false negative and false positive percentages. The non-diagonal numbers indicate the mismatch between both systems. Of the total reported 4.08% FN for the A58 setup, about 1/3 (1.23%) originates from the (POST-ON, PRE-ON) state, indicating the FCD AID switching on too late. Interestingly, for the total 11.99% false positive rate, the major contributions are from the (POST-OFF, PRE-OFF) state (2.66%)

corresponding to the FCD switching off too late and the (PRE-ON, POST-ON) state (1.83%) corresponding to the FCD AID switching on earlier than the loop AID. For the A27 setup, the biggest contribution to the false negative rate comes from the (PRE-OFF, POST-OFF) state (0.84%), with FCD AID thus switching off sooner.

Looking at the column/row totals for the 3 INTER states in both AID systems, FCD AID spends 91,571 s in all 'inter' states on the A58 while it spends 273,597 s on the A27. Similarly, the loop AID spends 187,268 s for the A58 and 245,707 s on the A27. These numbers again show the difference between both routes. Taking into account that the A27 covers nearly twice as many connectors as the A58 but only reports 31% more time, the A58 has more 'inter' time per AID sign compared to the A27. This is due to the systematic backwards propagating traffic jams resulting in stop-and-go traffic and the associated AID INTER switching. FCD AID only spends half the time the loop AID does for the A58 as with only a limited subset of vehicles, it smooths out these patterns, yielding less frequent switching. For the A27, the FCD AID spends roughly the same amount of time in the 'inter' states as the loop AID.

5.6 Discussion

In the previous sections, a state classification methodology was presented along with the results of a 4-week field test in which different FCD AID system configurations were deployed using data from a countrywide FCD system monitoring 5.93% of all traffic present on 2 routes with different traffic and congestion properties. Results were successfully coupled back to the road operators, proving full system technical feasibility. To score the FCD AID, it was compared to the installed loop AID and false negative percentages of 4-5% (of the total active benchmark AID) were obtained for a selected FCD AID configuration. This translates to a false negative of 82 resp. 158 s per AID sign per day on the A58 and A27 route respectively. Before considering the applicability, this number needs to be further put into context.

First, note that the loop AID is considered to be a golden standard, with its performance validated and tuned over the past decades. However, it also has its known disadvantages, for example its limited spatial resolution. The benchmark relies on local loop measurements but is blind for traffic problems/congestion forming between successive loop locations on the same route while FCD systems can operate on every part of the route. The evaluation itself was limited to the locations with loop equipment, with loop AID signage between locations by default being that of the upstream AID sign. The AID for those intermediate locations inherently suffers a delay (which the FCD AID does not have) but is not considered here. Due to the golden standard, instances where FCD would warn earlier than loops are

Table 5.4: Full state comparison for the A58 in configuration c_4. Column states correspond to the benchmark state while rows denote the state of the FCD AID. Cells marked in grey are false positives while cells marked in red are false negatives. Dark red denotes hard misses, light red are soft misses. Reported results are the percentages of time spent in each of the (loop AID, FCD AID) state combinations throughout the study period over all loops.

Benchmark→ FCD AID↓	**ON**	**OFF**	**PRE-ON**	**PRE-OFF**	**POST-ON**	**POST-OFF**	**INTER**	**PRE-INTER**	**POST-INTER**	**TOTAL**
ON	48.23%	0.72%	0.57%	3.35%	2.28%	0.88%	1.41%	1.91%	1.98%	1,370,397
OFF	0.14%	66,990,674	2.15%	0.33%	0.61%	1.01%	0.11%	0.07%	0.05%	67,090,357
PRE-ON	0.28%	2.45%	3.39%	0.02%	1.23%	0.03%	0.12%	0.08%	0.05%	170,545
PRE-OFF	0.39%	0.87%	0.02%	3.24%	0.05%	2.66%	0.06%	0.05%	0.10%	166,197
POST-ON	1.26%	0.53%	1.83%	0.10%	3.09%	0.04%	0.12%	0.22%	0.14%	163,633
POST-OFF	0.04%	3.64%	0.03%	0.45%	0.01%	3.46%	0.06%	0.02%	0.04%	173,156
INTER	0.06%	0.29%	0.19%	0.02%	0.05%	0.16%	0.36%	0.07%	0.13%	29,613
PRE-INTER	0.11%	0.15%	0.05%	0.15%	0.04%	0.22%	0.27%	0.32%	0.06%	30,784
POST-INTER	0.20%	0.14%	0.20%	0.04%	0.18%	0.03%	0.19%	0.06%	0.36%	31,174
TOTAL	**1,133,109**	**67,186,943**	**188,169**	**172,071**	**168,528**	**189,768**	**60,133**	**62,376**	**64,759**	**69,225,856**

Table 5.5: Full state comparison for the A27 in configuration c_4. Same definitions as for Table 5.4 were used

Benchmark→ FCD AID↓	ON	OFF	PRE-ON	PRE-OFF	POST-ON	POST-OFF	INTER	PRE-INTER	POST-INTER	TOTAL
ON	65.73%	0.90%	0.70%	0.94%	1.77%	0.35%	0.74%	0.96%	1.06%	3,104,646
OFF	0.73%	94,682,338	0.88%	0.40%	0.36%	1.32%	0.18%	0.13%	0.08%	94,855,611
PRE-ON	0.30%	1.96%	1.30%	0.02%	0.41%	0.05%	0.13%	0.03%	0.10%	181,840
PRE-OFF	1.10%	0.42%	0.03%	1.54%	0.09%	0.59%	0.03%	0.07%	0.05%	166,840
POST-ON	0.65%	0.70%	1.07%	0.04%	1.16%	0.01%	0.04%	0.04%	0.12%	162,425
POST-OFF	0.40%	1.16%	0.02%	0.84%	0.03%	1.68%	0.08%	0.09%	0.02%	182,575
INTER	0.55%	0.39%	0.15%	0.06%	0.05%	0.25%	0.38%	0.14%	0.09%	87,265
PRE-INTER	0.85%	0.19%	0.06%	0.24%	0.06%	0.17%	0.15%	0.38%	0.05%	91,399
POST-INTER	0.82%	0.25%	0.26%	0.04%	0.16%	0.06%	0.26%	0.06%	0.34%	94,933
TOTAL	**3,018,880**	**94,936,314**	**189,610**	**174,243**	**173,011**	**189,769**	**84,208**	**80,197**	**81,302**	**98,927,534**

also counted as false positives, limiting FCD performance to just mimicking the loop AID and not allowing improvements. Secondly, the FCD AID presented here was not tuned to specific locations (e.g. ramps, intersections) but used the same logic on every part of the route. This was done to better investigate the potential of a countrywide system, for which initial manual tuning for each road would be cumbersome. However, as seen by comparing the test cases, different traffic situations warrant different configurations. For the installed loop AID, each location is also tuneable in its configuration. Thirdly, the results were generated on a pilot AID system with a limited sample of live traffic while the loop system monitors all vehicles and is directly coupled into the AID signage. This results in the FCD AID being currently disadvantaged as it could perform even better with dedicated infrastructure designed to optimise delay. Lastly, the deployed FCD AID was made to mimic the current operational algorithm (with weighted averages, upper and lower thresholds, ...) to allow best integration with the existing systems. More advanced methods with forecasting models or flow optimisation can ultimately be used.

Given these disadvantages, the FCD AID results do indicate that FCD presents a valid source of traffic data for live traffic management systems. For the studied period, road setup and coverage, the study here shows that a live AID system can be deployed with reasonable differences. While it is tempting to put an exact number on the performance of the system, it is practically void as a 'perfect' AID system cannot be defined, as trade-offs will always need to be made. It is however clear that the presented AID system presents a cost-efficient alternative or addition to loop AID, as data fusion techniques can only improve the information for the loop AID. For areas currently without loop equipment, FCD AID offers a quick and practical solution. By leveraging FCD technology in these areas, we can reduce traffic congestion, combat pollution and increase safety.

Acknowledgements

Maarten Houbraken was a PhD fellow of the Research Foundation - Flanders (FWO-Vlaanderen). The authors would further like to thank the Dutch road operator Rijkswaterstaat (RWS) for providing the benchmark data. The authors would also like to thank RWS and the rest of the project partners for the fruitful discussions.

References

[1] B. Coifman. *Estimating travel times and vehicle trajectories on freeways using dual loop detectors*. Transportation Research Part A: Policy and Practice, 36(4):351–364, may 2002.

[2] H. X. Liu, X. He, and W. Recker. *Estimation of the time-dependency of values of travel time and its reliability from loop detector data*. Transportation Research Part B: Methodological, 41(4):448–461, may 2007.

[3] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, and Z. Jia. *Freeway Performance Measurement System: Mining Loop Detector Data*. Transportation Research Record: Journal of the Transportation Research Board, 1748:96–102, jan 2001.

[4] B. Yao, P. Hu, M. Zhang, and M. Jin. *A support vector machine with the tabu search algorithm for freeway incident detection*. International Journal of Applied Mathematics and Computer Science, 24(2):397–404, 2014.

[5] J. Xiao, X. Gao, Q.-J. Kong, and Y. Liu. *More robust and better: a multiple kernel support vector machine ensemble approach for traffic incident detection*. Journal of Advanced Transportation, 48(7):858–875, nov 2014.

[6] C. Oh, S. Park, and S. G. Ritchie. *A method for identifying rear-end collision risks using inductive loop detectors*. Accident Analysis and Prevention, 38(2):295–301, 2006.

[7] U. D. of Transportation. *DOT ITS Knowledge Resources*, 2017. Available from: https://www.itscosts.its.dot.gov/.

[8] J. Li, H. van Zuylen, and G. Wei. *Diagnosing and Interpolating Loop Detector Data Errors with Probe Vehicle Data*. Transportation Research Record: Journal of the Transportation Research Board, 2423:61–67, 2014.

[9] S. Cohen and Z. Christoforou. *Travel Time Estimation Between Loop Detectors and Fcd: A Compatibility Study on the Lille Network, France*. Transportation Research Procedia, 10(July):245–255, 2015.

[10] A. D. Patire, M. Wright, B. Prodhomme, and A. M. Bayen. *How much GPS data do we need?* Transportation Research Part C: Emerging Technologies, 58:325–342, 2015.

[11] M. Vanlommel, M. Houbraken, P. Audenaert, S. Logghe, M. Pickavet, and P. De Maeyer. *An evaluation of section control based on floating car data*. Transportation Research Part C: Emerging Technologies, 58:617–627, 2015.

[12] R. Neuhold, M. Haberl, M. Fellendorf, G. Pucher, M. Dolancic, M. Rudigier, and J. Pfister. *Generating a lane-specific transportation network based on floating-car data*. In Advances in Intelligent Systems and Computing, volume 484, pages 1025–1037, 2017.

[13] C.-s. Chou and A. P. Nichols. *Deriving a surrogate safety measure for freeway incidents based on predicted end-of-queue properties*. IET Intelligent Transport Systems, 9(1):22–29, feb 2015.

[14] G. Klunder, H. Taale, and S. Hoogendoorn. *The Impact of Loop Detector Distance and Floating Car Data Penetration Rate on Queue Tail Warning*. In 3rd International Conference on Models and Technologies for Intelligent Transport Systems, 2013.

[15] K. Liu, M.-y. Cui, P. Cao, and J.-b. Wang. *Iterative Bayesian Estimation of Travel Times on Urban Arterials: Fusing Loop Detector and Probe Vehicle Data.* PloS one, 11(6):e0158123, 2016.

[16] Y. Yuan, H. Van Lint, F. Van Wageningen-Kessels, and S. Hoogendoorn. *Network-Wide Traffic State Estimation Using Loop Detector and Floating Car Data*. Journal of Intelligent Transportation Systems, 18(1):41–50, jan 2014.

[17] M. Houbraken, P. Audenaert, D. Colle, M. Pickavet, K. Scheerlinck, I. Yperman, and S. Logghe. *Real-time traffic monitoring by fusing floating car data with stationary detector data*. In 2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2015, 2015.

[18] N.-e. E. Faouzi, H. Leung, and A. Kurian. *Data fusion in intelligent transportation systems: Progress and challenges A survey*. Information Fusion, 12(1):4–10, jan 2011.

[19] Y. Asakura, T. Kusakabe, L. X. Nguyen, and T. Ushiki. *Incident detection methods using probe vehicles with on-board GPS equipment*. Transportation Research Part C: Emerging Technologies, dec 2016.

[20] E. D'Andrea and F. Marcelloni. *Detection of traffic congestion and incidents from GPS trace analysis*. Expert Systems with Applications, 73:43–56, may 2017.

[21] R. Wang, D. B. Work, and R. Sowers. *Multiple Model Particle Filter for Traffic Estimation and Incident Detection*. Ieee Transactions on Intelligent Transportation Systems, 17(12):3461–3470, 2016.

[22] J. C. Herrera, D. B. Work, R. Herring, X. J. Ban, Q. Jacobson, and A. M. Bayen. *Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment*. Transportation Research Part C: Emerging Technologies, 18(4):568–583, aug 2010.

[23] E. Jenelius and H. N. Koutsopoulos. *Travel time estimation for urban road networks using low frequency probe vehicle data*. Transportation Research Part B: Methodological, 53:64–81, jul 2013.

[24] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. *Non-parametric estimation of route travel time distributions from low-frequency floating car data*. Transportation Research Part C: Emerging Technologies, 58:343–362, sep 2015.

[25] R.-P. Schäfer, K.-U. Thiessenhusen, and P. Wagner. *A Traffic Information System By Means of Real-Time Floating-Car Data*. In ITS world congress, number October, pages 1–8, 2002.

[26] A. Kinoshita, A. Takasu, and J. Adachi. *Real-time traffic incident detection using a probabilistic topic model*. Information Systems, 54:169–188, 2015.

"Great things are done by a series of small things brought together"

— Vincent Van Gogh (1853 - 1890)

6

Real-time Traffic Monitoring by fusing Floating Car Data with Stationary Detector Data

In the previous chapters, network modelling was mostly used in the process of turning the raw technical data samples into structured information. However, the data sources were mostly kept separated, to be compared to each other using performance metrics and to get an idea of the specific properties of each source. In this chapter, we break that separation and start combining both sources. This allows to take the best of both sources and to obtain a richer, more detailed view on the traffic conditions.

The main contribution of the development technique to fuse stationary detector data with floating car data (compared to the basic Treiber-Helbing Filter (THF)), is to first normalise both data sources to ensure data compatibility. This normalisation entails accounting for the specific properties (e.g. sampling frequency) of the data sources in their combination to avoid one source becoming too dominant. The data samples are weighted by their relevance in time and space. For loop detector data, this is related to the sampling frequency, the number of observed vehicles and the underlying measured segment length. For FCD, GPS samples are first mapmatched which results in a list of segments and a time spent on each segment by the indiviual vehicles. This time spent is used in normalising the con-

tribution of an individual vehicle. This evens out the difference in sampling rate between different vehicles, as well as the difference in speed.

The development of this technique was done within the Dutch research project 'Spookfiles' and has since been patented (see publication 3 in Section 1.7.4). The presented chapter is an application of that work.

⋆ ⋆ ⋆

Maarten Houbraken, Pieter Audenaert, Didier Colle, Mario Pickavet, Karolien Scheerlinck, Isaak Yperman and Steven Logghe

Presented at the 2015 conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), June 2015 in Budapest, Hungary.

Abstract Applying the current technological possibilities has led to a wide range of traffic monitoring systems. These heterogeneous data sources individually provide a view on the current traffic state, each source having its own properties and (dis-)advantages. However, these different sources can be aggregated to create a single traffic state estimation. This paper presents a data fusion algorithm that combines data on the data sample level. The proposed system fuses floating car data with stationary detector data and was implemented on live traffic. Results show the fusion algorithm allows to eliminate individual source bias and alleviates source-specific limitations.

6.1 Introduction

With the advances in communication technology and location-aware devices, more and more cars connect to or use Intelligent Transportation Systems (ITS) to get real-time traffic information and optimise their journey. These real-time ITS and their underlying traffic models vary depending on the available inputs, specific use case and intended features. However, all these models need accurate and real-time traffic state information to assess the current traffic conditions. In this work, we focus on large scale traffic state estimation using both real- time floating car data and stationary detector data.

Floating Car Data (FCD) denotes data generated by moving vehicles called probes. These vehicles collect data with on-board positioning devices that record their position (often at a fixed interval). This data is then transmitted to a data collection server that processes the data for all probes (see [1, 2] for details on sample processing). From there on, it can be used for a wide range of applications. One of the earliest consisted of calculating average travel times on routes throughout the day [3–5]. Note that this does not necessarily require real-time sample

transmission/processing as it can be done using historic data. However, the main strength of FCD lies in its real-time applications. Real-time sample aggregation and processing can be used for live traffic state detection [2] and estimation [6].

Stationary Detector Data (SDD) consists of data captured on fixed points in the road network by cameras, embedded inductive loops or other measuring equipment. This offers high resolution monitoring of the specific location (often with samples generated every 10-60 seconds). A full view on the traffic state along the trajectory can be made by combining the SDD from several points with macroscopic flow theory. Kalman filtering models the traffic as flows of cars and estimates the traffic state space for each time period [7]. Further extensions to the model have been made to account for irregular sample generation by using multiple modes of filter operation [8].

While both sources separately can create a full traffic view, their combination allows a more detailed and accurate estimation of the traffic state. This is called *data fusion* and aims to fully capture the information in the individual sources. While SDD provides a detailed view on the traffic state on the monitored locations, installing and maintaining these sensors on the entire road network would require high investments. FCD is more easily used for this type of traffic monitoring as its distributed design inherently monitors large geographical areas instead of fixed locations. However, FCD accuracy depends on the number of monitored probe vehicles, often limited to a small fraction of the total traffic.

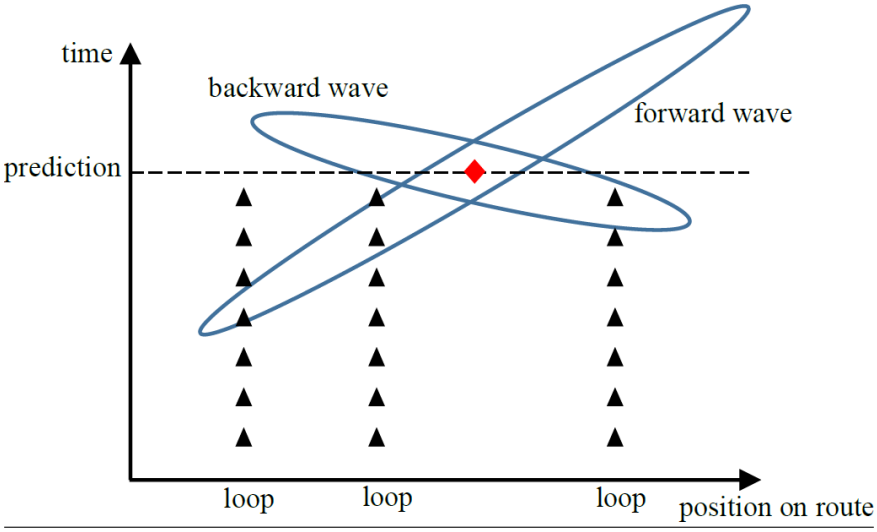
Data fusion itself has been done with a wide range of techniques, e.g. neural networks [9–12] or Kalman modelling [13], fusing additional data sources e.g. automatic licence plate recognition [14]. More example techniques and an overview of applications can be found in [15, 16]. In this paper, we focus on an adaptive smoothing technique proposed in [17, 18], also known as the extended and generalised Treiber-Helbing filter (EGTF). This filter uses kinematic wave theory applied to road traffic to merge individual samples to a full traffic state estimation.

6.2 Basic data fusion algorithm

For this EGTF technique, the road network is modelled as a dynamic system in which traffic flows along the roads, with cars driving from position A at time t_A to position B at time t_B. In free-flow traffic, the traffic conditions move along with the traffic, meaning that the traffic condition at position A at time t_A will be very similar to the traffic conditions at position B at time t_B. This implies that the traffic state at position B at a certain time t_x (possibly in the near future) can be predicted by using a traffic state from position A at an earlier time t_y.

In congested traffic, however, the traffic conditions move in the opposite direction of the traffic. While the traffic at the head of a traffic jam starts to move again, the back of the traffic jam is still stationary, with even more cars queuing

Figure 6.1 Schematic representation of the traffic wave in the EGTF.



on. This results in the head of the jam moving backwards and causes stop-and-go waves. More importantly, the prediction of the traffic state at position A should take into account the traffic state at position B (further along the highway in the normal driving direction).

These two mechanisms are modelled as separate waves in the Treiber-Helbing filter, moving at different speeds and in different directions. A schematic representation of the system is shown in Fig. 6.1. To estimate the traffic state for the position denoted by the diamond, the loop samples (denoted by triangles) on positions further along the route are taken into account by the backward propagating wave (denoted by the ellipsoid going from the top left to the bottom right). Analogously, the loop samples contributing to the forward wave are found in the forward ellipsoid (bottom left to top right). Following the above reasoning, the data samples within both ellipses are considered most relevant to the traffic state and will therefore be weighted more in the estimation of the traffic state at position P.

To calculate the traffic state, the available samples are processed in a weighted sum. The estimated speed $s(x,t)$ at position x and time t for one wave is then calculated as a weighted sum of the speed of all samples in the region of interest around the (x,t) point. This weighted sum is given by

$$s(x,t) = \frac{\sum_i s_i \cdot w_i(x,t)}{\sum_i w_i(x,t)} \tag{6.1}$$

with $w_i(x,t)$ the weight of a sample at position x_i, taken at time t_i, contributing to the state at position x at time t. The individual weights $w_i(x,t)$ are calculated as follows

$$w_i(x,t) = \exp(-\frac{|x - x_i|}{\sigma} - \frac{|t - t_i - ((x - x_i)/v)|}{\tau}) \tag{6.2}$$

with σ and τ denoting the width and time window, respectively, of the region of influence around the (x,t) point under estimation. This weight function favours samples according to the ellipsoids shown in Fig. 6.1. The σ and τ are tuned to the availability of samples, which depends on the average distance between installed detectors and their sampling interval. The v in the formula denotes the propagation speed of the wave and differs between the forward free-flow wave (v_{FF}) and the backward wave (v_{CONG}). This results in 2 speed estimates $s_{FF}(x,t)$ and $s_{CONG}(x,t)$.

To obtain a single speed estimate $v(x,t)$ from the 2 estimates, they are combined using the following equations

$$v(x,t) = z(x,t) \cdot s_{CONG}(x,t) + (1 - z(x,t)) \cdot s_{FF}(x,t) \tag{6.3}$$

$$z(x,t) = \frac{1}{2}(1 + tanh(\frac{V_c - min(s_{CONG}(x,t), s_{FF}(x,t))}{dV})) \tag{6.4}$$

with V_c the speed at which free-flow traffic transitions to congested traffic and dV the sensitivity around this threshold.

In Eq. 6.4, the two speed values (from the different propagating waves) are combined to estimate whether the traffic is in congestion ($z > 1/2$) or in free-flow ($z < 1/2$). If $s_{CONG}(x,t)$ or $s_{FF}(x,t)$ are below the transition threshold, the congested estimate becomes more dominant in the calculation of the final speed estimate $v(x,t)$.

6.3 Data source normalisation

The above calculation merges individual samples to a full traffic state estimate. However, when fusing different data sources like FCD and SDD, the individual samples from both sources have different fundamental properties, as they are generated using different measuring techniques. In the EGTF, this difference is taken into account by first calculating speed estimates for each source and then combining them using an extra weight factor denoting the estimated confidence/accuracy of the speed estimate. However, the fusion process can be done on the sample level, thereby needing only a single THF. The only requirement is that the individual samples from the different sources are taken into account equally to avoid bias towards an individual source. In the case of FCD and SDD, a correction is needed as both generate samples with different properties and frequency.

SDD typically consists of samples taken with a fixed frequency (e.g. 1 sample per minute). However, this fixed interval can vary from detector to detector. This

would result in some detectors producing more samples in the same time interval and therefore being favoured in the sample aggregation. Analogously, this frequency also needs to be tuned to the FCD generation which uses full trips (or trips in progress) to generate data samples, possibly multiple samples if the trip passes multiple road segments. This can be solved for each source individually by tuning the frequency of the sample generation and their importance in the sample weighting. By equalising the number of samples and their contribution to the calculated estimates, the estimation avoids favouring a specific source.

Depending on the source (and individual detector), the specific normalisation also needs to take into account the specific source measuring properties. SDD consists of data measured at fixed positions in the network. In contrast, FCD originates from tracking vehicles. The reported values in SDD are therefore inherently temporal averages while the FCD reports spatial averages (see [19] for a more detailed description of the difference). To merge the reported values, they need to be converted to the same averaging (spatial/temporal depending on the desired estimation) before being summed.

A final normalisation is needed to account for the coverage provided in the data sources. While SDD provides aggregated data for all vehicles, FCD only covers a small fraction of the total traffic on the road. This results in different absolute numbers of samples, varying according with traffic intensity as well as the FCD probe vehicle penetration rate. This can be solved by taking the penetration rate into account in the weighting.

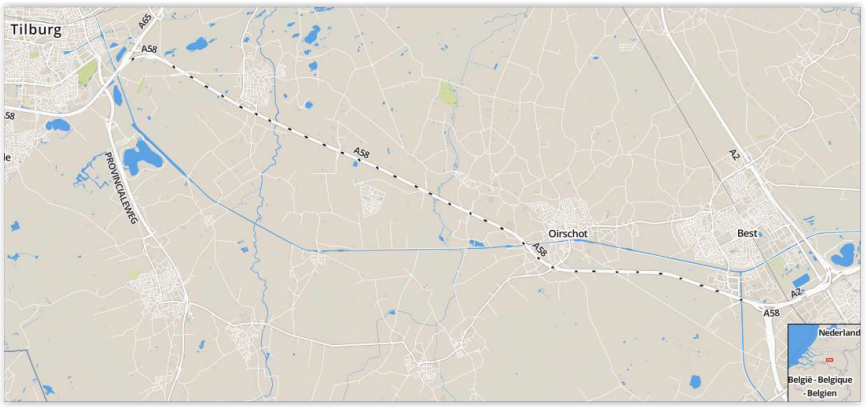
6.4 Experimental application: A58 use case

To validate the above method, the system was implemented for the 20 km of A58 highway from Tilburg to Eindhoven shown in Fig. 6.2. While the entire area of study in the 'Spookfiles' project (www.spookfiles.nl) entailed the neighboring A2 and A67, only results for the A58 are given here.

The A58 is monitored by 33 dual induction loops spaced evenly over the highway, resulting in available SDD roughly every 500 m. The live data from these loops was collected every minute and processed real-time to obtain the results below. The FCD was supplied by Be-Mobile, obtained from monitoring an estimated 3% of all traffic on the highway, higher than the 1.5% reported in [2]. Note that as the data is collected real-time, there needs to be accounted for delay, caused by the sample averaging of the measurement equipment, communication overhead and processing time.

Fig. 6.3 shows the output of the SDD plotted at their respective positions along the route. This data was combined with the aggregated FCD (depicted in Fig. 6.4) and processed using the THF described above. The results are plotted in Fig. 6.5. At 6:00 AM, the traffic on the route is smooth. The first congestion starts to form

Figure 6.2 A58 highway. The highway spans 20 km and contains 33 loop detectors, indicated in black.



around 6:50 AM near the Moergestel exit. This congestion propagates backwards (as a stop-and-go wave) to the start of the route (where 2 highways merge to form the A58) and reinforces the jam that was forming there. Around 7:00 AM, a congestion wave also start at the end of the route. This wave also propagates backwards and reaches the Oirschot intersection around 7:30 AM. This causes the traffic there to become disturbed for the rest of the morning rush hour. On top of the first wave, 2 more congestion waves propagate backward to Moergestel.

Comparing the SDD source to the fusion result, the fusion algorithm more clearly captures the traffic state when the distance between individual detectors becomes larger (e.g. between the start of the route and the Tilburg entry) or when traffic is turbulent (e.g. between the Moergestel exit and the preceding rest stop). Compared to the FCD source, the fusion algorithm provides a smoother traffic state with a clearer distinction between individual waves (e.g. between Tilburg and Moergestel).

The SDD provides accurate speed observations but is limited in spatial resolution compared to the FCD. The FCD in turn fluctuates very rapidly (as shown by the sharp colour differences in Fig. 6.4). The data fusion processing combines both data sources in a smooth traffic state. Note again that the prediction for a specific time and position is made using only older samples as data is collected and processed real-time.

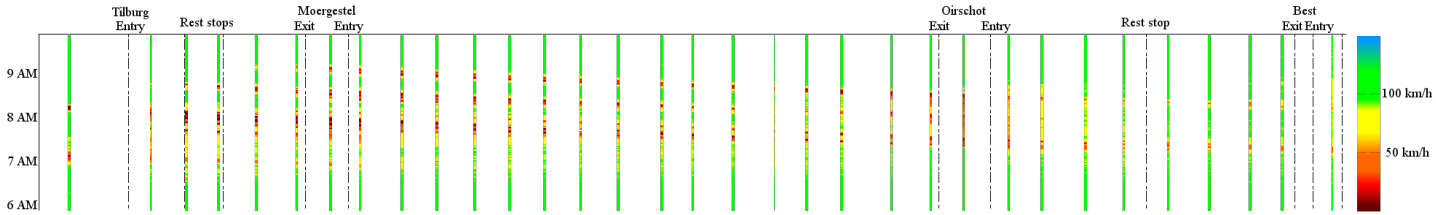


Figure 6.3: SDD on the A58 on 9th February 2015 between 6 and 10 AM (local time).

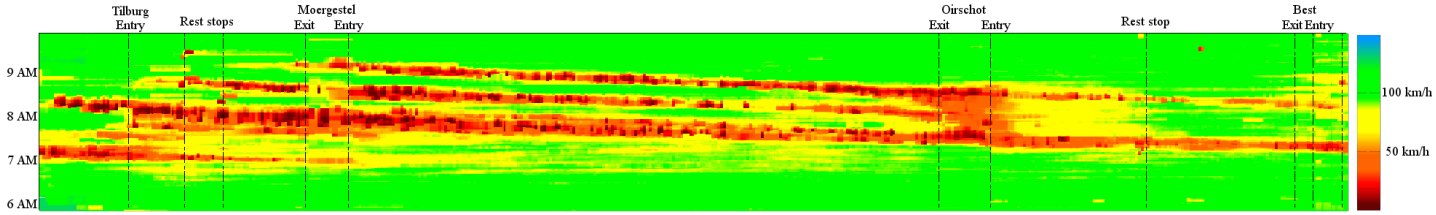


Figure 6.4: Aggregated FCD (same period as above).

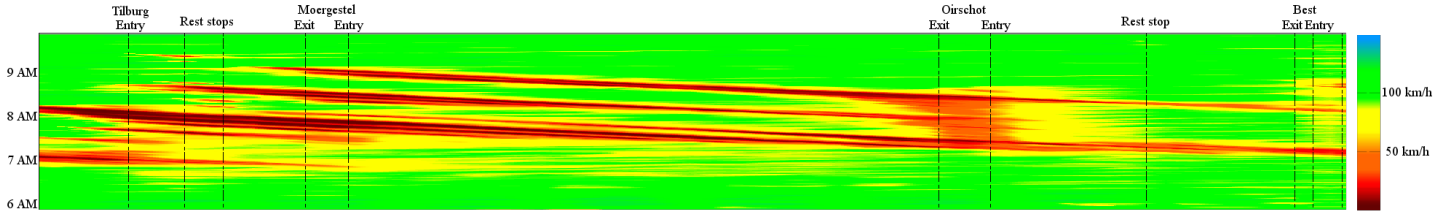


Figure 6.5: Fused traffic state (same period as above).

6.5 Conclusion and future work

With more and more heterogeneous traffic monitoring, the traffic state estimation has to incorporate these individual sources in a consistent framework. By normalising the individual data samples to take into account the measuring properties, they can be combined in a single data fusion algorithm to estimate traffic state. This approach was applied to fuse stationary detector data with floating car data, both gathered and processed real-time. Results show that this corrects individual data source bias, resulting in a cleaner traffic state estimation.

While our study was limited to the well-monitored A58 highway, most road infrastructure is not equipped with traffic sensors, thereby limiting the availability of SDD. However, the data fusion algorithm does not require a single source to provide a full traffic view. As samples from different sources can be merged, the algorithm can be applied to roads with fewer sensors. More interestingly, it may also be used to avoid having to install an extensive network of detectors, thereby lowering infrastructure cost.

Acknowledgements

Maarten Houbraken is supported by a PhD fellowship grant by the Research Foundation Flanders (FWO-Vlaanderen). The authors would also like to thank Be-Mobile for their cooperation and supplying the traffic data.

References

[1] M. Vanlommel, M. Houbraken, P. Audenaert, S. Logghe, M. Pickavet, and P. De Maeyer. *An evaluation of section control based on floating car data*. Transportation Research Part C: Emerging Technologies, 58:617–627, 2015.

[2] B. Kerner, C. Demir, R. Herrtwich, S. Klenov, H. Rehborn, M. Aleksi, and A. Haug. *Traffic state detection with floating car data in road networks*. In Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005., pages 700–705. IEEE, 2005.

[3] M. Vanlommel, P. De Maeyer, and S. Logghe. *Dynamical network management and historical analysis based on floating car data*. In 2nd international conference on Models and Technologies in Intelligent Transportation Systems, 2011.

[4] E. Brockfeld, B. Passfeld, and P. Wagner. *Validating travel times calculated on the basis of taxi floating car data*. Proceedings of 14th ITS Conference, 2007.

[5] M. Reinthaler, B. Nowotny, R. Hildebrandt, and F. Weichenmeier. *Evaluation of speed estimation by floating car data within the research project DMotion*. In PROCEEDINGS OF THE 14TH WORLD CONGRESS ON INTELLIGENT TRANSPORT SYSTEMS (ITS), 2007.

[6] C. de Fabritiis, R. Ragona, and G. Valenti. *Traffic Estimation And Prediction Based On Real Time Floating Car Data*. In 11th International IEEE Conference on Intelligent Transportation Systems, pages 197–203. Ieee, oct 2008.

[7] Y. Wang and M. Papageorgiou. *Real-time freeway traffic state estimation based on extended Kalman filter: A general approach*. Transportation Research Part B: Methodological, 39(2):141–167, 2005.

[8] T. Tettamanti, M. T. Horváth, and I. Varga. *Road traffic measurement and related data fusion methodology for traffic estimation*. Transport and Telecommunication, 15(4):269–279, 2014.

[9] J. N. Ivan. *Neural network representations for arterial street incident detection data fusion*. Transportation Research Part C: Emerging Technologies, 5(3-4):245–254, 1997.

[10] H. Dia and K. Thomas. *Development and evaluation of arterial incident detection models using fusion of simulated probe vehicle and loop detector data*. Information Fusion, 12(1):20–27, 2011.

[11] Q. Zhang and B. Li. *A low-cost GPS/INS integration based on UKF and BP neural network*. 5th International Conference on Intelligent Control and Information Processing, ICICIP 2014 - Proceedings, pages 100–107, 2015.

[12] Q. Ou, H. van Lint, and S. P. Hoogendoorn. *Fusing Heterogeneous and Unreliable Data from Traffic Sensors*. In Interactive Collaborative Information Systems, pages 511–545. 2010.

[13] E. Cipriani, S. Gori, L. Mannini, and S. Brinchi. *A procedure for urban route travel time forecast based on advanced traffic data: Case study of Rome*. 2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014, pages 936–941, 2014.

[14] M. Rahmani, E. Jenelius, and H. N. Koutsopoulos. *Floating Car and Camera Data Fusion for Non-parametric Route Travel Time Estimation*. Procedia Computer Science, 37:390–395, 2014.

[15] D. Schmidt, B. Friedrich, and F. Weichenmeier. *Data Fusion Techniques for Traffic State EstimationDino Within Dmotion*. 14th ITS World Congress, 49(0):1–6, 2007.

[16] N.-e. E. Faouzi, H. Leung, and A. Kurian. *Data fusion in intelligent transportation systems: Progress and challenges A survey*. Information Fusion, 12(1):4–10, jan 2011.

[17] M. Treiber and D. Helbing. *Reconstructing the Spatio-Temporal Traffic Dynamics from Stationary Detector Data*. Cooper@tive Tr@nsport@tion Dyn@mics, 1(3):1–24, 2002.

[18] J. W. C. Van Lint and S. 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.

[19] A. Kesting and M. Treiber. *Traffic flow dynamics: Data, models and simulation*. Springer: Berlin, 2012.

"I may not have gone where I intended to go, but I think I have ended up where I needed to be."

— Douglas Adams (1952 - 2001)

7 Conclusions & Future Challenges

This concluding chapter serves to link together the different contributions in this work and provide an overall overview of the attained results. The results are used to revisit the research challenges and quantify the progress towards them, as well as to identify the remaining future possibilities.

7.1 Conclusions

Within this dissertation, we delved deeper into the power of network modelling for algorithmic applications, focusing on the field of Intelligent Transportation Systems (ITS). In this field, data generation schemes, network analysis tools and traffic management applications all heavily rely on network modelling to provide a high-level abstraction of the underlying traffic infrastructure. Starting from that framework model, algorithms are used to convert the raw input data into the desired functional output. It is here that this dissertation finds its merit.

Traditionally, traffic data is generated and collected using roadside sensor equipment installed along the road or embedded in the road surface. While these sensors provide a detailed view on the local traffic conditions, they are limited to monitoring their location. Obtaining a network-wide traffic monitoring system is therefore hindered by high deployment and maintenance costs. To fill this need for wide area traffic monitoring, Floating Car Data (FCD) technology was introduced. This

provided spatially distributed network data to feed various applications. However, as a new technology, it remained unclear just what and how much information is available through FCD. With limited system deployments and unknown use cases reinforcing each other, FCD has several hurdles on its way to a mature technology. Two of these hurdles in the context of algorithmic development were considered in this dissertation and put forth as research challenges.

7.1.1 Network analysis

The first of the challenges consisted of **evaluating the potential of FCD for network analysis** and was considered in Chapters 2 and 3. Chapter 2 studies traffic characteristics along 2 Belgian highways by looking at the distribution of the speeds in the FCD and its evolution along the routes. The FCD consisted of individual vehicle trajectories polled every 30-60 s collected from a live FCD platform, this in contrast to traditional studies using loop data or simulated FCD. The changes in speed distribution were found to coincide with structural differences in road conditions such as road works and speed limit enforcement measures (section control systems and speed cameras). The speed distribution itself was used to compare the effectiveness of the different speed limit enforcement measures, providing valuable insights in driver behaviour for road operators. However, the fact that our FCD showed these differences is even more important with regards to our research challenge. After applying network modelling and processing to the raw positional vehicle data, the generated FCD allowed for speed analysis, showcasing its capability to analyse individual routes.

Further proving the suitability of FCD for network analysis, Chapter 3 focused on how FCD can be processed to intuitive speed profiles suited for congestion analysis. Among other data models, a custom speed profile was presented that focuses on capturing rush hour dynamics in the speed profile parameters. This model had the best modelling performance when evaluated on a large FCD dataset for the Netherlands. We designed the model to have intuitive, easy-to-use model parameters, which was demonstrated in a congestion analysis application detecting structural bottlenecks in the Netherlands.

With Chapter 2 illustrating local road analysis and Chapter 3 presenting wide area network analysis, it is clear that FCD offers a valuable data source for network analysis. Its inherent distributed nature allows for large area coverage at low costs compared to traditional roadside infrastructure.

7.1.2 Dynamic traffic management

The second challenge for FCD lies in its **applicability for Dynamic Traffic Management (DTM)**. For DTM, the road conditions need to be determined and acted upon in real-time, requiring more resources than offline network analysis. While for network analysis, the low penetration rate and data latency are mitigated by statistical aggregation and offline archiving, these issues need to be taken into account for live DTM systems.

Chapter 4 provided a first offline feasibility study for a Variable Speed Limit (VSL) system working on high-frequency FCD. Compared to other FCD research, our study uses high-frequency FCD collected from an operational country-wide FCD platform, thereby coupling our results closely to the current state-of-the-art systems. The proposed FCD VSL system was developed to mimic the behaviour of the installed VSL system working on inductive loop data. However, in contrast with the installed system, only a small part of all traffic is monitored using FCD. To evaluate the feasibility of VSL with only a fraction of all traffic, the FCD VSL performance was evaluated against the installed VSL system using a state-based scoring criterion on the generated VSL signs. Taking into account the known imperfections in the loop VSL, the FCD VSL performed well, successfully countering low penetration rates by taking into account upstream traffic conditions. However, only limited testing was done towards delay handling. This was more elaborately investigated in Chapter 5, further developing the VSL algorithm to work in a live production setting. This allowed for full chain performance testing with realistic live data properties to be handled by the VSL system. To the best of our knowledge, this is the first full chain technical and scientific feasibility trial of an FCD-only VSL system using real high-frequency FCD. Results here show that with careful implementation and minimising the different delay sources, the accumulated delay is sufficiently countered by tuned model parameters and the fine spatial granularity of FCD.

While Chapters 4 and 5 showed FCD to be a valid option for DTM compared to loop data, Chapter 6 shows how they can be combined to obtain the best of both worlds. The presented data fusion allows the high detail local information of inductive loops to be combined with the spatially distributed FCD. This is very interesting for DTM systems working with traffic properties not measurable (yet?) by FCD such as, for example, traffic intensities.

Considering our second research challenge, the last 3 chapters present successful use cases of real FCD for VSL. These examples show that, given the proper technical implementation, data availability and FCD (sampling) properties, FCD provides a valuable opportunity to improve our VSL systems and overall traffic conditions.

7.2 Future challenges

Overall, the work in this dissertation towards the research challenges has added to the technological maturity of FCD. Since its inception, FCD has slowly been growing and claiming its place as a solid data source in ITS. Raw data sources are becoming increasingly available, FCD providers have operational processing software and applications exist in both the private and the public sector. Our work showed how under the right circumstances FCD can be used for large scale network analysis and dynamic traffic management, each requiring their own approach and processing algorithms. However, our contributions only help FCD to take the next step and are definitely not the end of the road. Our recommendations for future research directions are grouped in the following categories:

- Technical/practical challenges on how to cope with future technological changes
- Methodological challenges concerning how to model our data
- Valorisation challenges on how to effectively introduce and use the data in the real world

7.2.1 Technical challenges

In the coming years, the FCD context will continue to grow and inevitably change. The number of connected vehicles has been steadily growing, thanks to more and more connected vehicle technologies (e.g. autonomous vehicles) and government regulations (e.g. GPS/GNSS-based tolling and the European eCall mandate), and it will continue to do so, with various estimates of +90% penetration rate for connected vehicle technology by 2020-2025. Data exchange will also further increase. As proven in Chapters 4 and 5, high-frequency FCD allows for new dynamic traffic management applications requiring low latency input data. FCD providers need to be ready for the increased data volumes and heterogeneity of sources. Standardisation will become necessary to ease source integration as well as allow FCD applications to work with different FCD providers. However, more research is needed into what data properties are most important (e.g. latency, accuracy and reliability) in various scenarios (e.g. dynamic traffic management and fleet monitoring) before a unified standard can be developed.

Another technical question left unanswered is “how much data we actually need?”. In our work, we aimed at showing the potential of our FCD and used all our available FCD. However, for each application, data requirements differ and changes in data quality affect the results differently. For policy makers and road operators, knowing these quality trade-offs along with the FCD quality available in their intended application area, would greatly help when deciding on implementing

specific applications. For our FCD VSL system, the quality trade-off between FCD penetration rate and system performance would be of particular interest when considering FCD VSL roll-out in new regions.

In terms of technical challenges, an ever-present challenge is how to utilise new technological advances. FCD itself is only possible thanks to advances in location services (e.g. GPS) and network development (e.g. large bandwidth) available to the general public. Even more technical advances are coming our way the coming years (e.g. Galileo satellites with increased accuracy and data flow guarantees in IPv6) that need to be followed to benefit from.

7.2.2 Methodological challenges

One of the most interesting directions for improving our traffic models and systems, is combining all our data sources with our traffic models. As shown in Chapter 6, data fusion allows to integrate FCD with loop data to estimate traffic state variables. These data fusion methods can definitely be further expanded e.g. short term traffic forecasting and long term data analysis.

Aside from more FCD data, more and more information about the vehicle is becoming available (e.g. engine status, turn signal indicators and windshield wipers). This data is very valuable when modelling the vehicle and its environment (e.g. traffic status and current weather conditions). While some research into getting and using this data (mostly information from the in-car CAN bus) is being done, there are still significant steps to be taken into fully incorporating this data into the traffic models.

This growth in content will not be limited to cars. Other modes of transport like buses, cyclists and pedestrians will also participate in this growing data exchange as they too become connected and want to optimise their journey. This trove of floating transport data will allow the individual entities to communicate and interact. Combining this data in a single model is no easy task and will need several more steps as all the different actors have their own characteristics to be incorporated. Combining all this information and possibly even input from stationary systems (e.g. traffic lights) will ultimately yield a full traffic model.

Considering FCD only, an interesting challenge still lies in making traffic models more data driven. There still exists a gap between the actual FCD collected from real situations and the more theoretically founded models. Ideally, these should go hand in hand, with model assumptions being validated, supported and/or corrected by FCD in a collaborative cycle. In general, a large field of possibilities exists for models and data to meet.

A more specific challenge in modelling for VSL systems is the need for more generic benchmarks independent of individual data sources. The FCD VSL system in our Chapters 4 and 5 was scored based on the existing loop-based VSL system.

This inherently introduces a bias towards loop-based implementations. This is unwanted and should be replaced by a scoring systems more founded in traffic theory, independent of the current implementation.

With more and more data becoming available, some of the current limitations in FCD traffic modelling can also be revisited. Currently, a major shortcoming for FCD is its inability to derive traffic intensities while this traffic information is heavily used in existing systems. Another disadvantage of current FCD platforms is that most results are only available on entire road segments and not on individual lanes. As location sensor accuracy increases, models could be developed with integrated lane separation, resulting in lane-based results. Overall, with the increased data volumes and/or new FCD models, new methods could be developed to improve details on existing variables or obtain estimates for currently unknown variables, allowing a wider application of FCD.

A final methodological challenge lies in the recent General Data Protection Regulation (GDPR) legislation. This set of privacy protection rules is relevant for the FCD collection schemes and its impact remains to be fully seen. Care needs to be taken when using individual vehicle trajectories and passing on the derived results to third parties. FCD providers need to ensure the privacy of the user. For example, a vehicle trip at 8:30 a.m. can easily be recognised as a morning commute, possibly revealing the user and his home address. This information needs to be protected by, for example, removing the first and last part of the trip. However, techniques typically used to avoid driver re-identification usually also reduce the data content and quality. An interesting research opportunity lies in developing data protection techniques that do not impact the traffic data content and retain the useful information.

7.2.3 Valorisation challenges

The biggest valorisation challenge for FCD lies in government legislation and the inherent uncertainty. While traffic management is primarily a governmental concern, most FCD platforms are in the hands of private industrial companies. As in the telecommunication sector, a balance needs to be found between both parties objectives to ensure both a viable business model for the private sector as well as an improvement to the public traffic situation.

As FCD is still a relatively young technology, it is not fully integrated in existing systems. This allows for significant gains to be reached by optimising those systems by introducing FCD. Other (more expensive) data sources could be reduced and model assumptions can be validated, yielding large savings and increased accuracy. For example, using FCD VSL could reduce the current loop infrastructure in the Netherlands, allowing for operational savings to be made. With regards to validating model assumptions, FCD is already being used to origin-

destination models to validate mobility assumptions. The challenge here is to identify the places and applications where FCD can be of value.

Closely related to reducing other data sensors, is using FCD to replace other data sources in existing applications. As shown in Chapters 4 and 5, FCD allows VSL without the need for road side sensor equipment. For roads without local sensors, this opens the door for DTM, allowing for an increased application area of the existing application.

Lastly, while existing applications can be integrated with FCD, new applications can be developed as well, working on the specific information available in FCD. Some innovative projects are already underway (e.g. instrumenting traffic lights on FCD to optimise traffic throughput, European project Traffic Management as a Service (TMaaS) in Ghent) but even more are possible. As the quality of FCD further improves, traffic management can be made more personal, bringing the advice into the car. As total system times decrease, closed loop traffic advice systems are possible, allowing advice to be given on an individual vehicle level real-time. Ultimately, FCD could be used in autonomous systems, to supply them with the near instantaneous traffic data they need. Developing these systems and optimally benefiting from the information in FCD is key in achieving optimal system gains.

On top of the challenges for new or adapted applications in the previous paragraphs, a secondary challenge lies in getting these applications up and running. Before the broad public can benefit from them, extensive testing, validation and integration testing is required. This is illustrated by our FCD VSL application. While Chapter 4 shows an offline study, Chapter 5 included interfacing with the existing system and a live trail is currently underway, even more phases of testing, evaluation and tuning are coming. Successfully passing all these phases, and possibly reducing the required time, is crucial in valorising the applications and making our traffic a safer place.

"Every creature is a living instruction that runs the algorithm of nature."

— Joey Lawsin

A

Fault Tolerant Network Design inspired by *Physarum polycephalum*

While the main chapters of this dissertation focus on network modelling for algorithmic design in the context of ITS, this appendix focuses on a more general network modelling algorithm. Networks in general are not static and are prone to link failures. The presented algorithm focuses on the design phase of network development and aims at incorporating fault tolerance in the network design. We take a look at one of nature's network builders, Physarum polycephalum, and how we can use its design principles to design networks of our own.

⋆ ⋆ ⋆

Maarten Houbraken, Sofie Demeyer, Dimitri Staessens, Pieter Audenaert, Didier Colle, Mario Pickavet

Published in Natural Computing, June 2013.

Abstract *Physarum polycephalum*, a true slime mould, is a primitive, unicellular organism that creates networks to transport nutrients while foraging. The design of these natural networks proved to be advanced, e.g. the slime mould was able to find the shortest path through a maze. The underlying principles of this design have been mathematically modeled in literature. As in real life the slime mould can

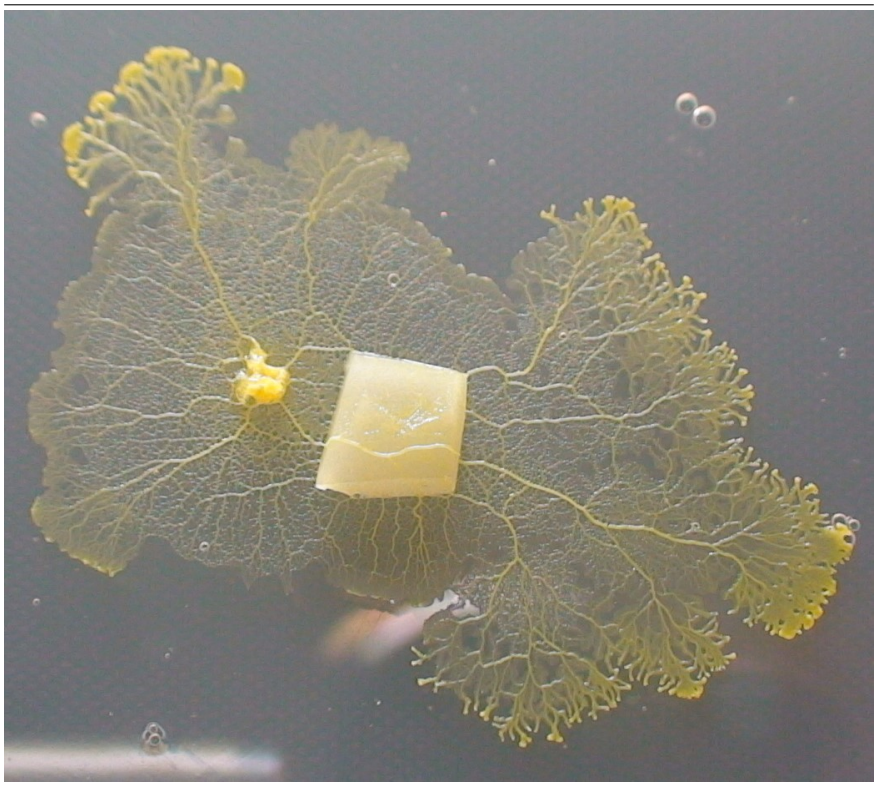
design fault tolerant networks, its principles can be applied to the design of man-made networks. In this paper, an existing model and algorithm are adapted and extended with stimulation and migration mechanisms which encourage formation of alternative paths, optimise edge positioning and allow for automated design. The extended model can then be used to better design fault tolerant networks. The extended algorithm is applied to several national and international network configurations. Results show that the extensions allow the model to capture the fault tolerance requirements more accurately. The resulting extended algorithm overcomes weaknesses in geometric graph design and can be used to design fault tolerant networks such as telecommunication networks with varying fault tolerance requirements.

A.1 Introduction

Transport networks are used to minimise the total effort that has to be invested for carrying information or goods from one point to another. Some well-known examples of man-made transport networks are road, computer and telephone networks. Not all transport networks are man-made however: many (more subtle) networks exist in nature. In fact, the human organism itself relies on several natural networks, like its vascular and nervous system, for everyday operation. Even more examples can be found in nature: trees transporting water from the roots to the leaves. *Physarum polycephalum*, a true slime mould, also creates networks while foraging. By studying all these natural networks and their design, we can improve our man-made networks. In this paper, we will look at how *Physarum polycephalum* designs networks and how we can extend these principles to design fault tolerant, robust networks.

Physarum polycephalum, shown in Fig. A.1, is an unicellular organism whose body consists of thousands of cell nuclei. It is a slime mould in which nutrients are transported through protoplasmic streaming. This protoplasmic streaming is well studied [1–4] and is based on tube morphogenesis. Inside the body of *Physarum*, a network of tubes exists and is used during foraging. When presented with food sources, *Physarum* concentrates around these sources to extract nutrients. These nutrients are dissolved in protoplasm and transported through the tubes to the rest of the body. The tubes grow bigger when transporting a lot of protoplasm. By growing bigger, the tubes are better suited for future transport as bigger tubes offer less resistance to the protoplasmic flow. Tubes that do not transport enough protoplasm shrink and eventually disappear due to a lack of flow. This mechanism is essentially a feedback loop that reinforces active paths and eliminates unused ones.

Figure A.1 *Physarum polycephalum* on agar surface. The slime mould is presented with a food source (centre of image) and uses a network of fine tubes during foraging to transport nutrients. The principles of network formation can be used to design fault tolerant networks. Image courtesy of Dr. Tanya Latty, University of Sydney.



All research on *Physarum* and the ease by which it can be cultured has led to various applications. [5] and [6] show how *Physarum* can implement logic functions while [7, 8] show that *Physarum* can be used as an unconventional computer and [9] explores the ability of *Physarum* to generate sound.

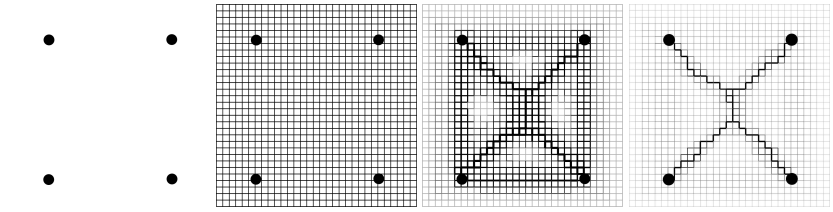
A more graph-related application of *Physarum* can be found in [10] and [11] which show that *Physarum* is capable of finding the shortest path through a maze. Aside from navigating a maze, *Physarum* was also shown to be able to anticipate periodic environmental changes in [12]. This behaviour is remarkable, considering the fact that *Physarum* has no central coordinating consciousness, and indicates an advanced underlying foraging strategy. More evidence of the capabilities of the slime mould is presented in [13] and [14] where the organism is shown to optimise its foraging activity to obtain an optimal nutritional diet.

The networking capabilities were further studied in [15] and [16] by letting the slime mould connect multiple food sources. A lot of attention has also gone to redesigning existing road networks. [17] and [18] model the cities in the Iberian peninsula and Mexico respectively as food sources and let the slime mould connect them. The resulting networks are compared to the road networks present in those areas. Road networks however are subject to historical evolution and geographical constraints, both having an influence on the resulting networks which cannot be neglected.

To capture the fundamentals of the slime mould, [19] and [20] formulate a mathematical model of the inner workings of the slime mould. In [21], it is used to redesign the rail network in the Tokyo area. The result of *Physarum* was very close to the existing network, proving its applicability to the network design problem. Another model of the fundamentals of the slime mould is presented in [22] using a multi-agent approach. This model was applied in [23] showing that it is close to the natural mechanism of *Physarum*, but it fails to find the shortest path through a maze.

In this paper, the mathematical model and algorithm from [19] are adapted to design fault tolerant networks as needed in for instance telecommunication. Telecommunication networks require high fault tolerance as they have to operate continuously and are prone to link and node failure (e.g. power outages, cable breaks, ...). Networks have to be designed to handle these problems [24, 25] but at the same time an operator wants to avoid over-dimensioning and unnecessary costs. As an important part of the installation cost is trenching and digging [26], low total lengths of the networks are desired. To better achieve these goals, 2 extensions are made to the model which also allow more automated and unambiguous design. The extended model is less representative of the biological slime mould but better equipped to generate fault tolerant networks.

Figure A.2 Typical simulation: Fig. a shows the food sources at the start of the algorithm. A fine square mesh of nodes and edges is then added to the food sources in Fig. b. Only the edges connecting the nodes are shown as lines, nodes are situated on all line intersections. Fig. c and d show the network changing during the simulation. The conductivities of the edges are denoted by the thickness of the edges with thick edges representing edges with a high conductivity.



A.2 Mathematical model

The basis for the mathematical model further used in this article is developed in [19]. In this model, the food sources (FS) are represented by nodes which are interconnected through a network of tubes, modeled by edges in a graph. During execution of the algorithm, the properties of the edges constantly change until a state of equilibrium is reached. A typical simulation is given in Fig. A.2.

The space in which the network will be built initially only contains nodes representing food sources (see Fig. A.2a). To interconnect the FS, a fine mesh of nodes is added to the space along with edges connecting these nodes as shown in Fig. A.2b. This models a network of fine tubes which transports the nutrients from one FS to the rest of the organism. All edges are assumed to be bidirectional.

The flow through an edge is a result of the pressure differential between the edge's endpoints. This type of flow is Poiseuille [1, 27] and can be expressed as

$$Q_{ij} = \frac{\pi a_{ij}^4}{8\kappa} \cdot \frac{p_i - p_j}{L_{ij}} \tag{A.1}$$

where a_{ij} is the radius of the tube, κ the kinematic viscosity of the fluid, L_{ij} the length of the edge, p_i the induced pressure in the node N_i and Q_{ij} the flow through the edge between N_i and N_j. The first term on the right hand side of the equation can be contracted to a single variable, simplifying the equation to

$$Q_{ij} = \frac{D_{ij}}{L_{ij}} \cdot (p_i - p_j) \tag{A.2}$$

where D_{ij}, the conductivity of an edge, indicates the suitability for transport. A high D_{ij} value indicates that the edge offers low resistance for transport (per unit

length) while an edge with a low D_{ij} value offers more resistance and is less suited for transport.

Based on Eq. (A.2), an iterative algorithm can be developed that modifies the initial network and its edges to create a network connecting the FS. In each iteration, a pair of FS is randomly selected between which a flow is set up. When a flow is imposed between a source and a sink node, it will spread throughout the network according to the following set of equations

$$\sum_j \frac{D_{ij}}{L_{ij}} \cdot (p_i - p_j) = \begin{cases} I & N_i \text{ is source} \\ -I & N_i \text{ is sink} \\ 0 & \text{else} \end{cases} \tag{A.3}$$

where I denotes the size of the imposed flow. These equations express the conservation of flux inside the network: for each node N_i which is not a source or a sink in the network, the total amount of flow entering the node must equal the total amount of flow exiting the node. Only in the source and sink node can the transported fluid leave the network.

The flow through each edge of the network Q_{ij} can be calculated by combining Eq. (A.2) with the solution of Eq. (A.3). Using these flows, the conductivities of all edges can be adjusted, simulating the growth of the tubes in the slime mould. This adaptation can be done by using the following equation

$$\frac{D_{ij}^{n+1} - D_{ij}^{n}}{\Delta t} = f(|Q_{ij}^{n}|) - D_{ij}^{n} \tag{A.4}$$

where $f(|Q_{ij}^{n}|)$ denotes the growth function of the tube which is chosen to mimic the behaviour of the real slime mould. For low flow values, the edge should decay while for higher flow values, the edge should thicken. There is however a limit on the size of the tube. These constraints translate to a monotonically increasing function saturating for high values. The function used further in this paper is

$$f(|Q|) = \frac{(1+a)|Q^{\mu}|}{1 + a|Q^{\mu}|} \tag{A.5}$$

with $1 \leq \mu \leq 2$, $a > 0$ resulting in a sigmoidal curvature for $f(|Q|)$.

After the adaptation, the algorithm chooses another pair of (source, sink) nodes and repeats the calculations. The (source, sink) pairs are randomly selected which each node having a predetermined probability of being selected. A node can be made more important by increasing the probability of being selected. A higher chance to be selected results in more flow passing through the tubes surrounding the node, possibly resulting in more edges surviving.

Another optimisation consists of calculating and averaging several flows in one iteration before the conductivities are adapted and is presented in [28]. This can be used to limit fluctuation of Q_{ij} and D_{ij}. Ideally, all (source, sink) combinations

should be calculated in one iteration. This would require solving a lot of linear systems as there are $\frac{n\cdot(n-1)}{2}$ such combinations. Only a few combinations are used in the same iteration as a trade-off between combinatorial completeness and execution time. Fluctuation on the values was sufficiently reduced by this (limited) averaging.

A.3 Extensions

To improve the fault tolerance of the created networks and automate the design process, we propose two extensions to the algorithm of [19]. These extensions consist of the migration mechanism (Section A.3.1) and the stimulation mechanism (Section A.3.2). As these extensions are intended to improve fault tolerance, the end results of the extended algorithm will differ from the networks created by the biological slime mould and the original algorithm. The general outline of the algorithm is given in Algorithm A.1. As explained in Section A.2, the core of the algorithm consist of iteratively selecting a (source, sink) pair, determining the flow through the network and updating the model instance. One iteration consists of lines 3 through 12. The (source, sink) pair is selected randomly with each node having a predetermined probability of being selected. When multiple (source, sink) combinations in the same iteration are wanted to reduce fluctuation of Q_{ij} and D_{ij}, lines 4 through 8 are executed multiple times and the resulting Q_{ij}^k averaged. The algorithm stops on line 13 when the average fluctuation of the D_{ij} falls below a predetermined threshold or after a certain number of iterations.

Algorithm A.1: The extended algorithm in pseudocode

```
1:  initialisation
2:  repeat
3:    for k = 1 to k = nrPairs do
4:      Select new (source, sink) pair
5:      Calculate flows Q_ij^k                          // Eq. (A.2) & (A.3)
6:      if Q_max^k > τ · I then
7:        stimulate alternative paths                   // Section A.3.2
8:      end if
9:    end for
10:   Average flows Q_ij^k to Q_ij
11:   Update D_ij                                       // Eq. (A.4)
12:   Migrate nodes                                     // Section A.3.1
13: until stopping criterion
```

A.3.1 Migrating nodes

The model from [19] as presented in Section A.2 assumes all nodes and edges to be at a fixed location. While this is useful to determine the optimal route to be followed through a maze [10, 19], it is less convenient in (design) situations where the location of only some nodes is predetermined. By constraining the position of the nodes and edges, the result of the original algorithm is limited to the paths present in the initial mesh. The type of mesh used then determines the possibilities in the end result. Optimal paths not present in the initial mesh can be unintentionally excluded. Moreover, the presence of alternative paths can influence the flow distribution at the start of the algorithm by taking their part of the flow. This alters the reinforcement of the edges and ultimately the competition between the different edges.

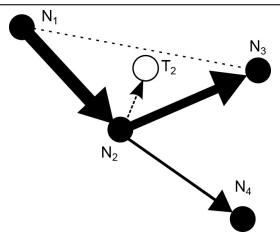
A possible approach to approximating the optimal paths in the initial mesh could be to use a very fine-grained mesh. This would however increase the number of nodes and edges significantly, which in turn would slow down computations severely as the number of equations in the linear system to be solved increases. The optimal configuration could then still not be present in the initial mesh. Another possibility is to use non-uniform meshes with an increased number of edges in the areas of interest but this could favour specific configurations.

To preserve fairness among the edges in the initial mesh and to limit the complexity of the linear system, we propose to let the nodes in the mesh move. By using a simple moving mesh, more mobility is incorporated in the extended algorithm. This effectively reduces the influence of the initial mesh on the end result as the edges can be redistributed. It also simplifies post-processing of the networks e.g. no more need to manually straighten edges. At the end of each iteration, the coordinates of the nodes are adjusted according to the flows at the nodes themselves. This adjustment is not applied to nodes representing FS, as they have to remain fixed. For each other node N_i, a set of 'target' coordinates T_i is calculated as follows

$$T_i = \frac{\sum\limits_{\forall j} C_j \cdot |Q_{ij}|}{\sum\limits_{\forall j} |Q_{ij}|} \tag{A.6}$$

These T_i coordinates are weighted sums of the coordinates of the neighboring nodes, C_i, with the flows on the edges between them acting as weights. The coordinates will be two-dimensional when on a planar map, but they can easily be extended to support higher dimensional spaces. Fig. A.3 shows the basic mechanism at work. The thickness of the edges between the nodes represent the amount of flow between the nodes. N_2 receives flow from N_1 and sends the bulk part of it to N_3. A smaller amount of flow is sent to N_4. When T_2 is calculated, N_3 has a larger impact than N_4. T_2 is situated closer to the straight edge between N_1 and N_3 than C_2.

Figure A.3 Migration mechanism applied on a single node. N_2 wants to migrate towards T_2 which is predominantly determined by N_1 and N_3 because of the size of the flows towards and from N_2, respectively.



Once the T_i's are calculated, the nodes will migrate towards their target location. If in the following iterations in the extended algorithm a similar flow distribution is encountered, the path of the dominant flow will be shorter, requiring less energy for transport. To prevent extensive migration, node movement is restricted to the area around the initial position. To this end, the movement vector M_i is calculated by $\overrightarrow{M_i} = T_i - C_i^0$ with C_i^0 denoting the coordinates of N_i at the start of the algorithm.

The movement vector can then be used to calculate (and limit) the distance of T_i to C_i^0. To limit the migration to a disc of size ϵ around the initial coordinates, it suffices to calculate the norm of $\overrightarrow{M_i}$ and clip those movement vectors that have a norm $> \epsilon$. To find the final target coordinates ($\boldsymbol{FC}_i$), the clipped movement vectors can be added to the original coordinates C_i^0. More elaborate schemes can be used to prevent the node from going into forbidden areas e.g. walls in a maze.

Using the current coordinates C_i^n, the new coordinates of the nodes are then calculated as

$$C_i^{n+1} = \frac{C_i^n + \psi \cdot \boldsymbol{FC}_i}{1 + \psi} \tag{A.7}$$

where ψ is used to smooth migration over several iterations in the algorithm. As mentioned in Section A.2, only a few (source, sink) combinations are used each iteration. As a result, the T_i can vary across iterations. The smoothing prevents excessive fluctuation of C_i. In the later iterations, most redundant edges have disappeared, resulting in fewer contributions to and fluctuation on T_i. The new coordinates can then be used to calculate the new lengths of the edges in the next iteration.

A.3.2 Stimulation of alternative paths

A second extension to the model from [19] focuses on improving fault tolerance of the resulting networks. The biological slime mould forms its networks by thickening the tubes that carry a lot of flow. The cultivated *Physarum* networks consist

of many tubes with varying radii. To extract a useful network from these experiments, edges have to be selected. In [17], the selection is done based on weights associated with the edges. The weights are calculated as the ratio of the experiments where the edge occurred to the total number of experiments performed. The authors use a threshold to identify important edges. The threshold greatly influences fault tolerance, as using a low threshold and retaining a lot of edges will result in a higher fault tolerance than when only the thickest tubes are retained.

Furthermore, once a thick tube is formed between 2 parts of the network, it tends to carry most flow between the 2 parts. This dominance leaves only little flow for the alternative paths which then often disappear due to the lack of reinforcement. The lack of alternative paths is reflected in a low fault tolerance.

To prevent paths from becoming too dominant, the flows in the extended algorithm are redistributed when a node has too much flow passing through it before they can affect the edge's conductivity. First, the node K with the maximal amount of flow going through it, Q_{max}, is determined by

$$Q_{max} = \max_n \frac{1}{2} \sum_i |Q_{in}| \tag{A.8}$$

$$K = \arg\max_n \frac{1}{2} \sum_i |Q_{in}| \tag{A.9}$$

If Q_{max} is larger than some (predetermined) part of the total amount of flow through the network $(\tau \cdot I)$, the total flow should be redistributed. The total flow can now be divided in 2 parts: a part Q'_{ij} that represents an amount of flow passing through K and the remainder that follows alternative routes. An approximation of Q'_{ij} can be found by solving

$$\sum_j \frac{D_{ij}}{L_{ij}} \cdot (p'_i - p'_j) = \begin{cases} -2Q_{max} & N_i = K \\ Q_{max} & N_i = N_r \text{ or } N_k \\ 0 & \text{else} \end{cases} \tag{A.10}$$

$$Q'_{ij} = \frac{D_{ij}}{L_{ij}} \cdot (p'_i - p'_j) \tag{A.11}$$

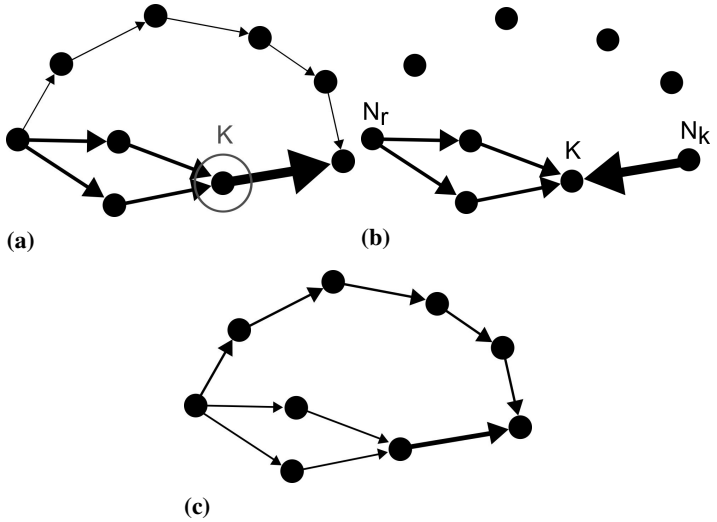
where N_r and N_k refer to the original source and sink used in the current iteration of the iterative algorithm, respectively. The set of equations are similar to Eq. (A.3), only now the original source and sink act as sources, both sending a flow of Q_{max} to K. The resulting Q'_{ij} values represent an amount of flow Q_{max} going from the original source to the original sink, passing through K. The remainder is then given by $Q_{ij} - Q'_{ij}$. To increase the flow through alternative paths, it suffices to set the new Q_{ij} as follows

$$Q_{ij}^{new} = \alpha \cdot \tau \cdot I \cdot \frac{Q'_{ij}}{Q_{max}} + \beta \cdot (1 - \tau) \cdot I \cdot \frac{Q_{ij} - Q'_{ij}}{(I - Q_{max})} \tag{A.12}$$

where α and β can be set to increase the relative importance of the terms. The Q_{ij}^{new} values can then be used in Eq. (A.4) to adapt the conductivities.

The first term in Eq. (A.12) contains the Q'_{ij} values representing a flow of size Q_{max} from the source to the sink passing through K. This flow is rescaled to a flow of size $\tau \cdot I$. As the redistribution is only done when too much flow passes through a node, this rescaling lowers the amount of flow passing through the forming dominant path. The second term represents flow from source to sink using other paths. This flow increases in size, resulting in an increased reinforcing of the alternative paths to improve the fault tolerance of the network.

Figure A.4 Steps in stimulation process: Fig. a shows too much flow passing through node K. The flows converging in K are calculated by Eq. (A.10) and shown in Fig. b. The flow is redistributed by Eq. (A.12) as shown in Fig. c.



Because the stimulation mechanism is integrated in the extended algorithm, the fault tolerance is integrated in the design process. This results in conductivities that are coupled with the importance of the edges for fault tolerance. The selection of edges based on conductivities/radii now more accurately captures the fault tolerance objectives.

A.4 Simulations

This section presents the results of the extended algorithm. First, the general methods used in the simulations are briefly explained in Section A.4.1. Then, two more technical aspects of the algorithm are discussed. The extended algorithm is shown

find the shortest path through a maze in Section A.4.2 and the applied thresholding is justified in Section A.4.3. Section A.4.4 shows the benefits of the extensions while Section A.4.5 relates to the work in [17]. An application to telecommunication networks is given in Section A.4.6.

A.4.1 Methods and parameters

The starting networks used in the following simulations were generated by considering the points to be connected as food sources (FS), characterised by their Euclidean coordinates. To these FS nodes, a square 100x100 mesh was added with each FS connected to its closest neighbors. As the mesh will be changed by the migration mechanism, the simple square configuration suffices to get different interconnection structures. The linear systems of equations were solved by calculating the minimum norm residual solutions. To minimise fluctuations on D_{ij}, $nrPairs = 2$ different (source, sink) pairs were calculated and averaged. Unless specified otherwise, the different pairs were randomly generated with each of the FS having equal probability of being selected.

The parameters used in the calculation of flows were $(a, \Delta t) = (1, 0.01)$. The ψ parameter used in the migration was set to 0.3. The clipping of movement vectors was done using an increasing disc size $\epsilon = (0.15 \cdot l)^i$ with l the average initial link length and i the current iteration. The parameters were chosen to limit the migration in the first 1,000 iterations. The stimulation parameters were set to $(\alpha, \beta, \tau, I) = (1, 2, 0.5, 1)$. The simulations were stopped after 10,000 iterations. This stopping criterion could be improved as the network evolution stabilised sooner and final network structure could be extracted after fewer iterations.

To evaluate the simulation results, several metrics are calculated for comparison. The total length is the sum of the lengths of all edges present in the end result with a conductivity higher than 5% of the maximal conductivity present (see Section A.4.3). All lengths and distances are calculated using the Euclidian distance measure. The single (resp. double) fault tolerance indicates which percentage of single (resp. double) link faults can be handled by the rest of the network. A fault is successfully handled when all nodes are still connected after the link has failed. The probability of a link failing is taken to be proportional to its length. This definition is closely related to the availability of a (telecommunication) network, which indicates the percentage of time a network is operational. The calculation of availability would however require estimating realistic recovery times and link failure rates. The diameter of a network is the longest path among all shortest paths in the graph. The fault diameter is the maximum diameter of the graph after a fault has occurred [29]. The average internodal distance is calculated by averaging the shortest paths between all pairs of nodes.

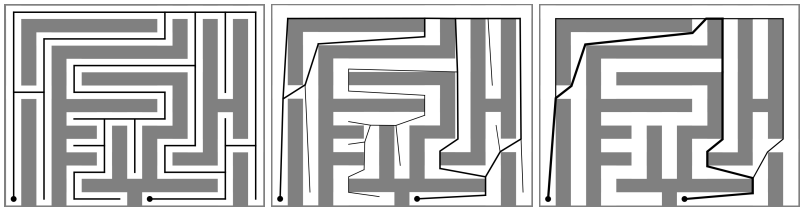
To compare the simulation results to more theoretical graphs, the Gabriel Graph (GG), Relative Neighborhood Graph (RNG) and Minimal Spanning Tree (MST) are used. These graphs are part of the Toussaint hierarchy of proximity graphs [30]: MST$\subseteq$ RNG $\subseteq$ GG and can be created based on the coordinates of all nodes. The GG is created by adding an edge between vertex a and vertex b if no other vertex is in the closed disc with line segment $|ab|$ as diameter. The RNG is constructed by adding a link between two nodes if no other node is closer to both nodes than they are to each other [31]. The MST is the graph with the lowest weight that still directly interconnects all nodes. The length of the MST is a good indication of the minimal length of any network connecting all nodes (without Steiner points). The Euclidean Minimal Spanning Tree (= MST with inclusion of Steiner points) was not used as the difference in length was very small ($\sim$ 3.5%) and both have no redundancy.

A.4.2 Validation

As the original model is extended with two extensions, the path-finding abilities of the model might not be present in the extended model. Fig. A.5 shows the result of the extended algorithm on a maze. The walls were considered forbidden territory during migration. The extended algorithm can (still) find the shortest path through the maze. The extended algorithm cannot attain maximal single fault tolerance as there are no completely disjoint paths from source to sink. The result does contain 2 alternatives which increase fault tolerance. These alternatives can be eliminated by increasing the μ parameter, resulting in a trade-off.

During the simulations, the two phases from [19], dead end cutting and selection of the solution path from the competitive paths, are also observed. The dead ends in the maze do not receive flow due to the lack of FS and die out quickly. The competition between the different paths is more complex than the dead end cutting and requires more iterations.

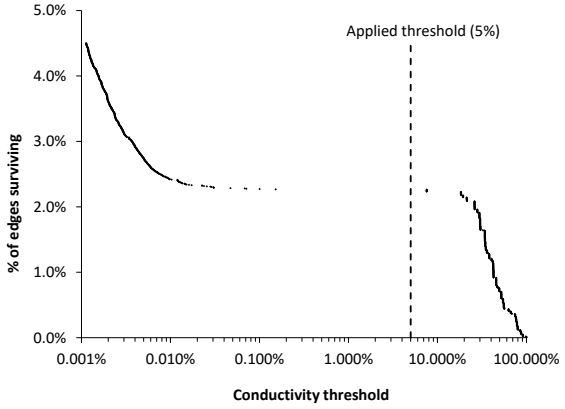
Figure A.5 Simulation of maze-solving capabilities. Fig. a shows the start of the simulation with all edges having the same conductivity. Fig. b shows the dead end cutting after a few iterations and Fig. c shows the final state of the network. Edges with a high conductivity are drawn thicker than edges with a low conductivity.



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

A.4.3 Effects of extensions on thresholding

Figure A.6 Relationship between edge survival and the applied conductivity threshold. The graph shows the percentage of edges present in the end result (compared to the total number of edges in the initial mesh) when the conductivity threshold is varied. Using a low threshold results in a lot of edges remaining while higher thresholds result in more edges being cut. No edges were present in the end result with a conductivity value in the range of 0.2% to 7%, resulting in a wide gap around 1%. The threshold of 5% used in all experiments is denoted by the dashed line. Threshold values are displayed using a logarithmic scale.



During analysis of the end results of the simulations, a threshold is applied on the conductivities of the edges. This thresholding discards all edges with a low conductivity as they offer too much resistance to the flow. The percentage of edges surviving in a simulation using the extended algorithm is shown in Fig. A.6. Using low thresholds results in more edges from the initial graph 'surviving' the computation. A threshold of 0.001% (of the maximum conductivity present) results in 4.5% of the initial edges being present in the end result. As the initial graph contained a lot of redundant edges, added along with the fine mesh, it is natural that only a small percentage is left in the end. Increasing the threshold results in more edges being considered unsuitable and cut from the end result. This lowers the total length but could eliminate alternative paths, lowering the fault tolerance of the networks. However, as Fig. A.6 shows, there is a clear separation in the conductivity values. In the experiment of Fig. A.6, no edges were present in the end result with a conductivity value in the range of 0.2% to 7%. Varying the threshold in this range would not influence the end result (no edges would be dropped). As this separation was present in all simulations, a static threshold could be applied

independent of all other parameters. Based on a limited number of simulations, the threshold was set at 5% and used in all further simulations.

A.4.4 Comparison to original model

To show the effects of the extensions, the original and the extended algorithms are used on a model of Belgium. A set of 16 Belgian cities was made based on the number of inhabitants and/or regional importance: Aalst, Antwerp, Arlon, Bruges, Brussels, Charleroi, Ghent, Hasselt, Kortrijk, La Louvière, Leuven, Liège, Mechelen, Mons, Namur and Wavre. Fig. A.7 shows some simulation results of the extended algorithm together with the MST, RNG and GG. The importance of Antwerp and Brussels was increased to incorporate their relative importance. Fig. A.7a shows how the extended algorithm connects all cities. By increasing μ in Eq. (A.5), fewer edges survive as shown in Fig. A.7b and Fig. A.7c, similar to results in [19]. The RNG and MST do not have high fault tolerance as several cities can be disconnected by a single fault.

Fig. A.8 compares the results of the original algorithm to those of the extended. The μ parameter from Eq. (A.5) was varied between 1.3 and 1.9 to obtain the simulation results, all other parameters were as described in Section A.4.1. The analysis results varied nicely with μ. High μ values resulted in few edges surviving, corresponding to the results in the left part of the graph. When μ was lowered, the resulting graphs had more edges resulting in an increased total length and fault tolerance. The range of μ values was chosen to show the trade-off between cost and fault tolerance.

The effect of the stimulation mechanism can be best seen in the top portion of the fault tolerance graphs. The extended algorithm can attain maximal single fault tolerance while the original algorithm cannot. Arlon, the most southern city on the map, is relatively far away from the rest of the graph making it very costly to connect to the rest of the graph with more than one path. While an extra path greatly increases the fault tolerance, it also increases the total network length significantly. This results in very few analysis results (for the extended algorithm) around $1.6MST_{length}$. Both situations (extra path and no extra path) can also be seen in Fig. A.7b and Fig. A.7c. The influence of the increase in μ is too big for the stimulation mechanism to handle without increasing the stimulation parameters in Fig. A.7c.

The effect of the migration mechanism is best seen in the left portions of the fault tolerance graphs in Fig. A.8. On top of the (limited) fault tolerance increase of the stimulation mechanism, the length of the networks is shorter due to the paths being made straight by the migration. This shows the importance of the initial mesh for the end result.

Figure A.7 Results for Belgian network: Fig. a-c show some results for the extended algorithm with constant stimulation parameters and μ respectively 1.3, 1.4 and 1.6. Fig. d, e and f show the MST, RNG and GG connecting the cities.

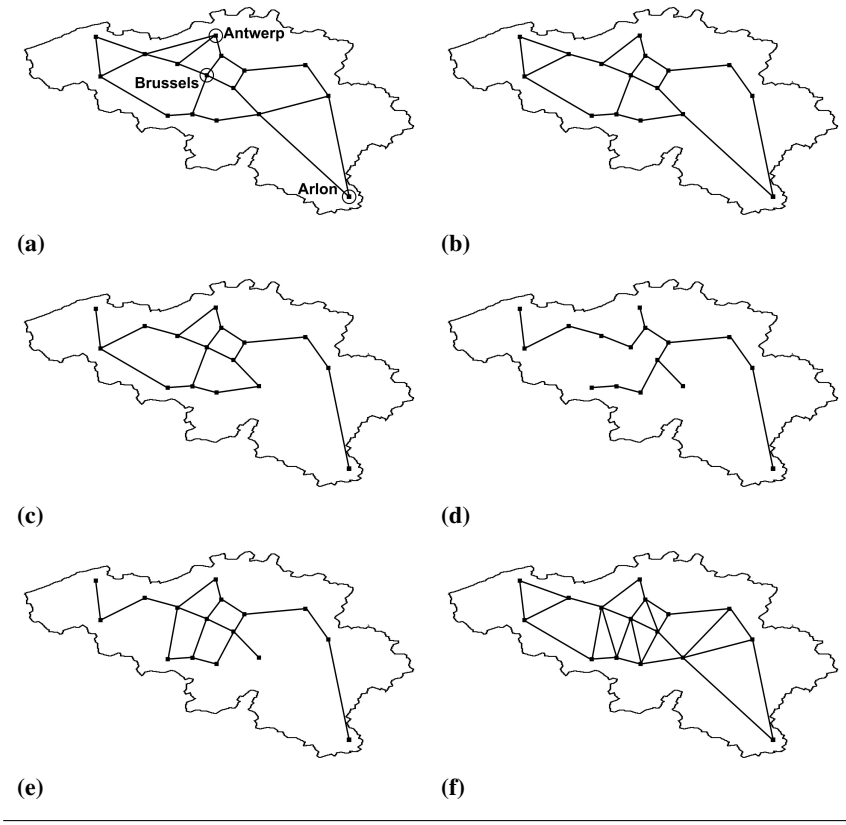


Figure A.8 Analysis of simulation results. The fault tolerance and the average size of the minimum cut of the results are compared to the total length. The indicated lengths are calculated by dividing the total length of the networks by the total length of the MST. The simulation results using the model without using any extension are represented by crosses. The simulation results using both extensions are denoted by circles. The MST has unit length. The RNG and GG of the network are represented the diamond and the triangle.

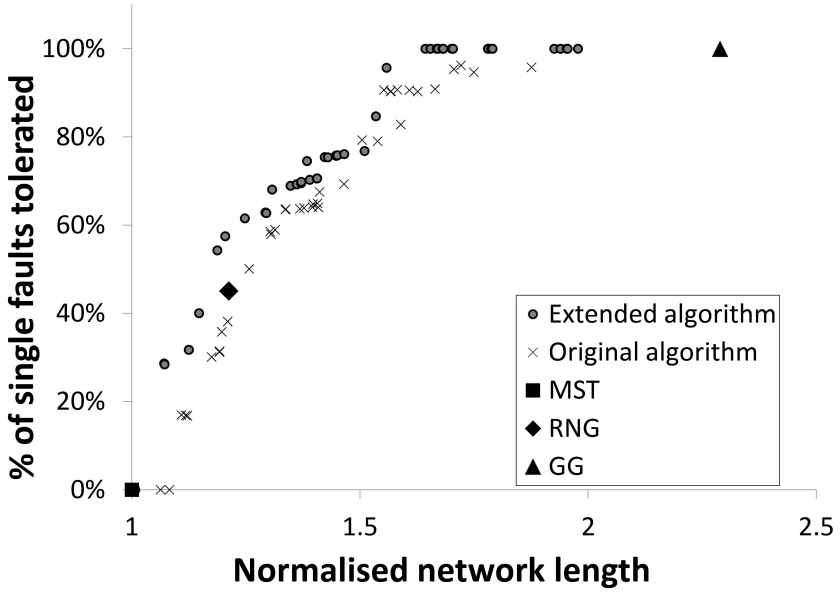
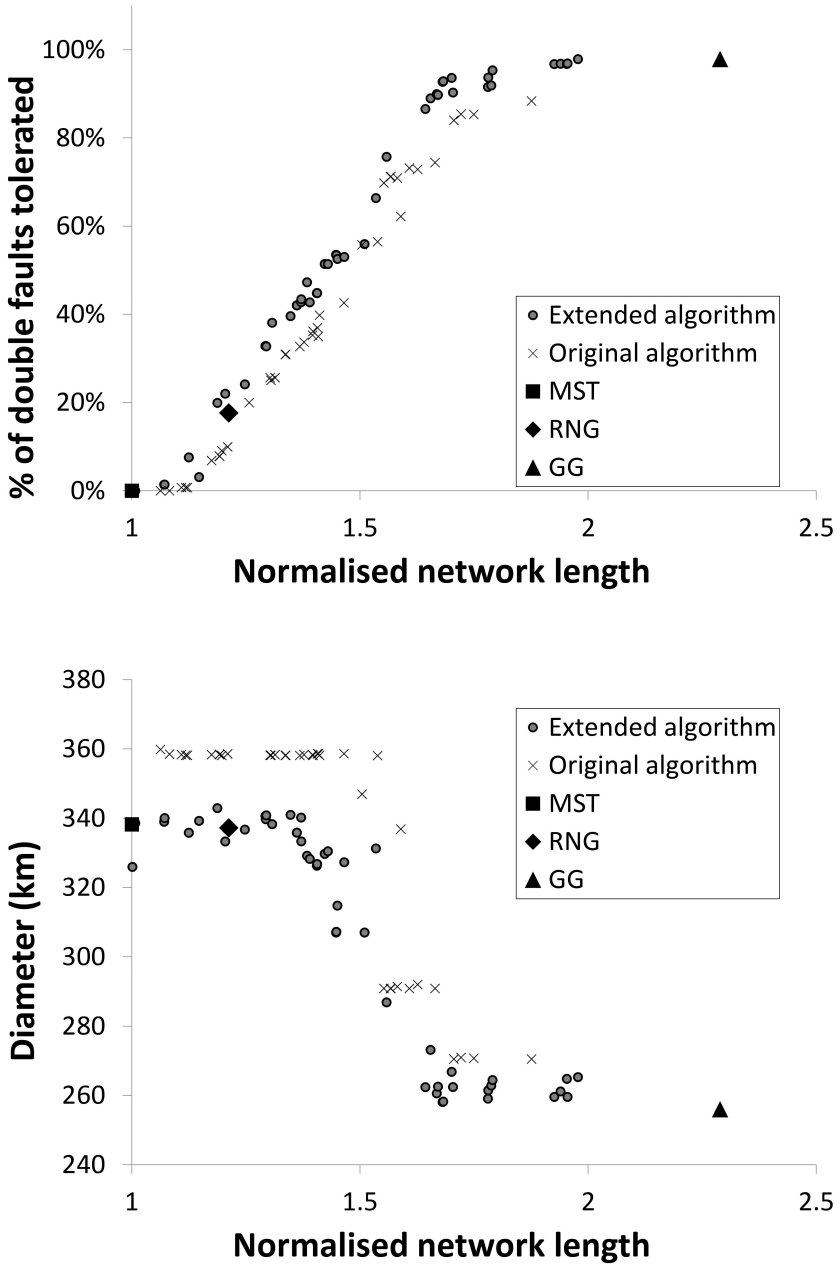


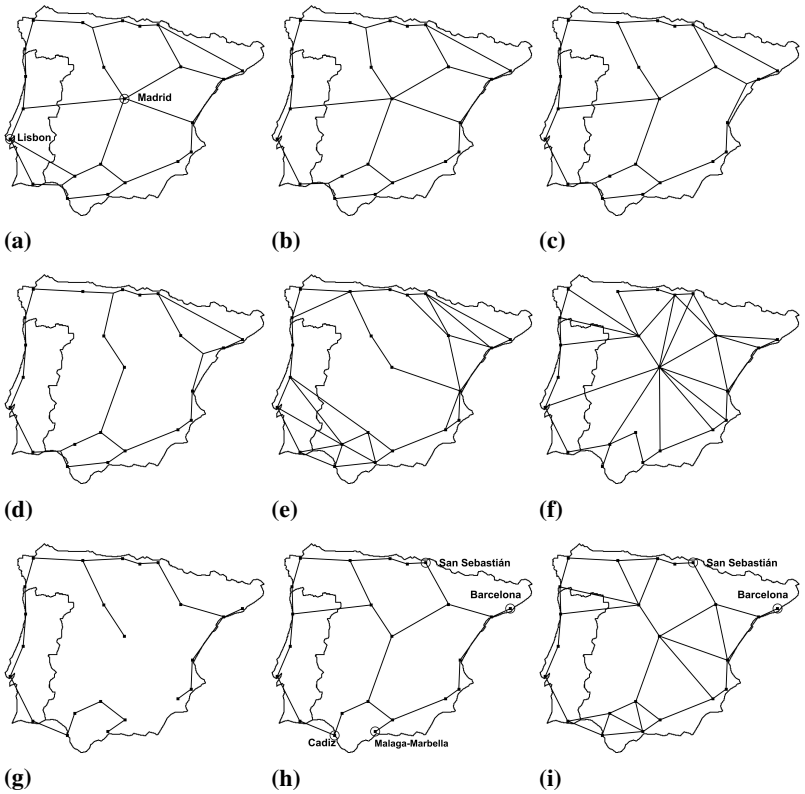
Figure A.8 Analysis of simulation results (continued).



A.4.5 Comparison to living slime mould

To compare the extended algorithm to the biological slime mould, the extended version of the algorithm is applied to the same input dataset of [17] and the results are compared. The same set of cities on the Iberian peninsula is used as input for the network generation. The biological network and the road network are recreated by using straight interconnections to obtain the same logical structures. The MST, RNG and GG are created based solely on the model of the Iberian peninsula. Fig. A.9 shows some results of the extended algorithm. More important cities (Madrid, Lisbon) were given a larger weight in the selection process to incorporate their relative importance. Table A.1 shows an analysis of the different networks.

Figure A.9 Results for the Iberian peninsula: Fig. a - d show the results of the extended algorithm with varying μ values respectively 1.2, 1.3, 1.4 and 1.5. Fig. e shows the results of the living slime mould and Fig. f shows the existing road network, both taken from [17]. Fig. g, h and i show the MST, RNG and GG connecting the cities.



The length of the existing road network and the length of the result of the living slime mould are much higher than the lengths of the simulation results. In the man-made network, this is caused by the multiple (redundant) roads starting in a.o. Madrid (center of Spain). The length of the biological network depends on the cut value [17]. By raising the cut value, lower lengths can be obtained but this would disconnect Madrid from the rest of the graph. Varying the cut value would only provide limited gains. The simulation results do not have this problem as the conductivities are tied to the fault tolerance. The cut value could be kept constant for all simulations as the conductivities of the edges sharply fell around the applied threshold (see Section A.4.3). The best (fault) diameter is found in the road network together with the highest total length.

All networks, aside from the MST, RNG and GG, have maximal single fault tolerance. The MST, RNG and GG cannot handle all single link faults as a single link failure can isolate Barcelona from the rest of the network. The fault diameter of these networks should be ∞ (infinite distance from any node to Barcelona). The values given in Table A.1 for these networks however are calculated by ignoring faults that cause the graph to become disconnected. Fault diameters calculated by ignoring some failures are denoted with an asterisk ($*$). The RNG and GG were expanded by adding the link (Barcelona, San Sebastian) and the link (Malaga-Marbella, Cadiz) (already in GG). The numerical results of these networks are denoted by RNG+ and GG+. The results of the RNG+ are good, considering their construction time, and similar to the results of the extended algorithm. The extended algorithm does have a notably lower fault diameters. The average intern-odal distance is slightly lower for the simulation results and the double fault tolerance values of the simulation results are higher than the values for the reference networks. The (human) adjustment to the RNG greatly increases its resilience but this dependency is not desirable for automated design of more complex networks. The GG+ also has good results but has a high network cost.

In summary, the results of the extended algorithm on the Iberian peninsula show the benefits of the extended algorithm. Thanks to the coupling of the conductivity values to the fault tolerance, the cut value for retaining edges in the end result could be kept constant while it had to be determined manually for the living slime mould. Manual corrections, needed for the proximity graphs, were not needed in the results of the extended algorithm. The algorithm further allows to design networks with various levels of fault tolerance by varying μ.

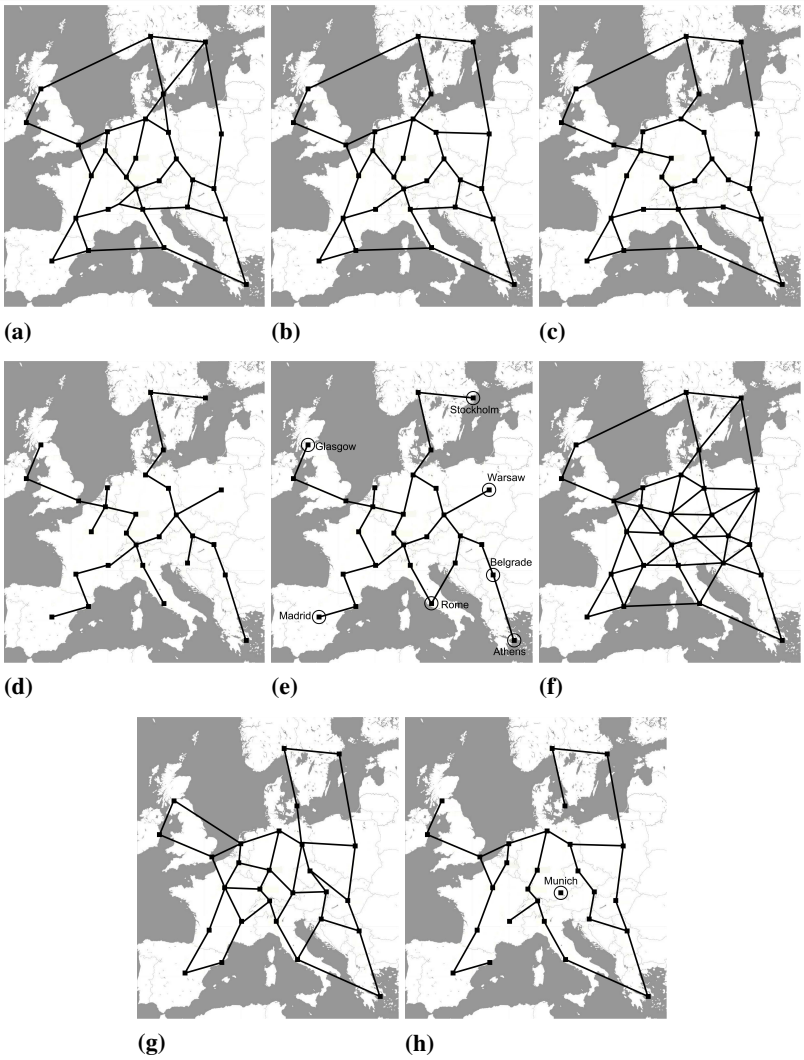
A.4.6 Application to telecommunication networks

To show the potential of the extended algorithm to design telecommunication networks, it is compared to a realistic telecommunication network design. The reference material for this network is publicly available in [32]. Fig. A.10 shows some simulation results, the reference telecommunication network and the proximity graphs. The numerical results of the experiment are shown in Table A.2.

With the exception of the MST and RNG, all networks had maximal single fault tolerance. The RNG could handle 45.55% of all link failures, the MST (by definition) none. The fault diameter was calculated as in Section A.4.5. The results of the extended algorithm greatly resemble the reference telecommunication network. Fig. A.10h shows all edges present in both the reference network of Fig. A.10g and the simulation result in Fig. A.10b. Most differences are found in the center of the graph. Only one city (Munich) has no edges that appear in both the reference network and the simulation result. Table A.2 shows that the total length is lower, the average internodal distance, diameter and fault diameter are smaller and the double fault tolerance is higher. As in Section A.4.5, the network designer can vary μ to design networks with a desired network resilience. Fig. A.10c attains a 13% reduction in total length compared to the reference telecommunication network in exchange for a lower network resilience.

The reference RNG has a low single/double fault tolerance. The RNG construction process has difficulties connecting the cities due to the specific network topology. For several cities like Athens, Glasgow, Madrid, Stockholm and Warsaw an extra edge would be desirable to increase fault tolerance but there's always another city nearby that causes the city to have no other edge (e.g. an edge between Athens and Rome would be desirable but Belgrade is closer to both Athens and Rome than Athens is to Rome, preventing the RNG construction process to add an edge). This effect was also visible in the RNG on the Iberian peninsula, though only limited to Barcelona, and the RNG in the Belgian experiment. Manually adjusting the result is more ambiguous now as a lot of adjustments are needed to attain maximal single fault tolerance. This shows that, despite its simplicity and the good results the RNG(+) produced for the Iberian peninsula, its fault tolerance is dependent on the topology of the graph and can be quite low. For designing fault tolerant networks, this dependency is undesirable. The GG achieves maximal fault tolerance but at an increase in total length of 30% compared to Fig. A.10b. The MST again has the shortest length but no fault tolerance.

Figure A.10 Telecommunication networks: a-c show some simulation results of the extended algorithm obtained by varying μ between 1.2 and 1.6. d, e and f show the MST, RNG and GG connecting the cities. g shows the reference network from [32] while h is the graph containing all common edges between b and g.



A.5 Conclusion

In this paper, the potential of the true slime mould *Physarum polycephalum* to design fault tolerant networks is analysed. The mathematical model from [19] is implemented and adapted. While the resulting networks of the original algorithm can be steered towards fault tolerance, the original model has no special provisions focused on it. The fault tolerance in the networks is very sensitive to the threshold applied during post-processing edge selection on the generated networks. To reduce this dependency, we extended the original model with a stimulation mechanism that redistributes the flow through the network. This enables alternative paths to survive and results in higher fault tolerance and smaller fault diameters in the generated networks. The threshold used for edge selection could be kept constant as the conductivities were more tightly coupled to the fault tolerance and sharply fell around the threshold. Another dependency of the original model lies in the initial configuration. The initial positions of the nodes and their interconnections determine the possibilities in the resulting networks. By extending the model with a migration mechanism, the possible paths can change during execution, offering more freedom to the algorithm without severely increasing computation times. The extensions were tested on models of Belgium, Europe and the Iberian peninsula. Our extended algorithm can achieve higher fault tolerances than the original algorithm and can be tuned according to the desired level of fault tolerance. Comparisons to reference road and telecommunication networks show that the extended algorithm can be used when fault tolerant networks are desired and total length is to be minimised. It can also overcome specific weaknesses in geometric graph designs such as the Minimum Spanning Tree and the Relative Neighborhood Graph and find the shortest path through a maze. By varying the μ parameter, a trade-off between the fault tolerance of the resulting network and its total length can be made.

Table A.1: Results for Iberian peninsula.

Network	Total length (km)	Double fault tolerance (%)	Average internodal distance (km)	Diameter (km)	Fault diameter (km)
Simulation					
Fig. A.9a	5,946.18	98.37	618.95	1,175.82	1,563.10
Fig. A.9b	5,199.35	96.88	623.85	1,175.64	1,563.36
Fig. A.9c	4,957.57	96.15	643.38	1,317.38	1,565.75
Fig. A.9d	4,226.58	87.77	670.23	1,372.04	2,209.85
Reference					
Biological	7,059.65	98.78	636.60	1,284.12	2,025.91
Roads	8,373.09	96.99	600.31	1,154.39	1,294.85
GG	6,335.71	96.33	595.81	1,310.18	1,524.65*
GG+	6,738.59	99.32	594.36	1,310.18	1,524.65
RNG	4,254.27	82.50	664.34	1,337.40	1,970.23*
RNG+	4,825.37	95.99	649.64	1,329.82	1,714.45
MST	3,121.00	0.00	1,026.00	3,121.00	3,121.00*

Table A.2: Results for the European telecommunication network.

Network	Total length (km)	Double fault tolerance (%)	Average internodal distance (km)	Diameter (km)	Fault diameter (km)
Simulation					
Fig. A.10a	17,587.99	98.25	1,298.72	3,443.64	4,658.34
Fig. A.10b	16,751.63	97.77	1,307.62	3,443.02	4,657.71
Fig. A.10c	15,039.37	95.97	1,409.65	3,425.32	5,032.62
Reference					
SNDlib	17,344.00	96.71	1,360.82	3,615.40	4,763.34
GG	23,946.12	99.29	1,169.33	3,213.70	4,316.25
RNG	10,612.77	16.65	1,472.88	3,888.82	4,895.38*
MST	9,361.08	0.00	1,639.53	4,381.95	4,381.95*

References

[1] N. Kamiya. *Protoplasmic streaming*. 1959.

[2] K. Gotoh and K. Kuroda. *Motive force of cytoplasmic streaming during plasmodial mitosis of physarum polycephalum*. Cell Motility, 2(2):173–181, 1982.

[3] A. Tero, R. Kobayashi, and T. Nakagaki. *A coupled-oscillator model with a conservation law for the rhythmic amoeboid movements of plasmodial slime molds*. Physica D: Nonlinear Phenomena, 205(1-4):125–135, jun 2005.

[4] R. Kobayashi, A. Tero, and T. Nakagaki. *Mathematical model for rhythmic protoplasmic movement in the true slime mold*. Journal of Mathematical Biology, 53(2):273–286, 2006.

[5] S. Tsuda, M. Aono, and Y. P. Gunji. *Robust and emergent Physarum logical-computing*. BioSystems, 73(1):45–55, 2004.

[6] J. Jones and A. Adamatzky. *Towards Physarum binary adders*. BioSystems, 101(1):51–58, 2010. arXiv:1005.2303.

[7] A. Adamatzky. *From reaction-diffusion to physarum computing*. Natural Computing, 8(3):431–447, 2009.

[8] A. Adamatzky. *Physarum Machines: Computers from Slime Mould*. 2010.

[9] E. R. Miranda, A. Adamatzky, and J. Jones. *Sounds Synthesis with Slime Mould of Physarum Polycephalum*. Journal of Bionic Engineering, 8(2):107–113, jun 2011.

[10] T. Nakagaki, H. Yamada, and Á. Tóth. *Maze-solving by an amoeboid organism*. Nature, 407(6803):470, 2000. arXiv:0105001v2.

[11] T. Nakagaki, H. Yamada, and Á. Tóth. *Path finding by tube morphogenesis in an amoeboid organism*. Biophysical Chemistry, 92(1-2):47–52, 2001.

[12] T. Saigusa, A. Tero, T. Nakagaki, and Y. Kuramoto. *Amoebae anticipate periodic events*. Physical Review Letters, 100(1):1–4, 2008.

[13] T. Latty and M. Beekman. *Food quality affects search strategy in the acellular slime mould, Physarum polycephalum*. Behavioral Ecology, 20(6):1160–1167, 2009.

[14] A. Dussutour, T. Latty, M. Beekman, and S. J. Simpson. *Amoeboid organism solves complex nutritional challenges.* Proceedings of the National Academy of Sciences of the United States of America, 107(10):4607–4611, mar 2010.

[15] T. Nakagaki, R. Kobayashi, Y. Nishiura, and T. Ueda. *Obtaining multiple separate food sources: behavioural intelligence in the Physarum plasmodium*. Proceedings of the Royal Society B: Biological Sciences, 271(1554):2305–2310, 2004.

[16] T. Nakagaki, H. Yamada, and M. Hara. *Smart network solutions in an amoeboid organism*. Biophysical Chemistry, 107(1):1–5, 2004.

[17] A. Adamatzky and R. Alonso-Sanz. *Rebuilding Iberian motorways with slime mould*. BioSystems, 105(1):89–100, 2011.

[18] A. Adamatzky, G. Martinez, S. Chapa-Vergara, R. Asomoza-Palacio, and C. Stephens. *Approximating Mexican highways with slime mould*. Natural Computing, pages 1–20, 2010. arXiv:arXiv:1010.0557v1.

[19] A. Tero, R. Kobayashi, and T. Nakagaki. *A mathematical model for adaptive transport network in path finding by true slime mold.* Journal of theoretical biology, 244(4):553–564, feb 2007.

[20] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, and T. Nakagaki. *Flow-network adaptation in Physarum amoebae*. Theory in Biosciences, 127(2):89–94, 2008.

[21] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki. *Rules for biologically inspired adaptive network design*. Science, 327(5964):439–442, 2010.

[22] J. Jones. *Approximating the Behaviours of Physarum polycephalum for the Construction and Minimisation of Synthetic Transport Networks*. In Unconventional Computing, pages 191–208. 2009.

[23] M. Becker. *Design of fault tolerant networks with agent-based simulation of Physarum polycephalum*. In Evolutionary Computation (CEC), 2011 IEEE Congress on, pages 285–291. IEEE, 2008.

[24] J. Vasseur, M. Pickavet, and P. Demeester. *Network recovery: protection and restoration of optical, SONET-SDH, IP, and MPLS*. Morgan Kaufmann Publishers, 2004.

[25] M. Pickavet, P. Demeester, D. Colle, D. Staessens, B. Puype, L. Depré, and I. Lievens. *Recovery in multilayer optical networks*. Journal of Lightwave Technology, 24(1):122–134, jan 2006.

[26] K. Casier, S. Verbrugge, R. Meersman, D. Colle, M. Pickavet, and P. Demeester. *A clear and balanced view on FTTH deployment costs*. J. Inst. Telecommun. Prof, 2(3):27–30, 2008.

[27] G. K. Batchelor. *An introduction to fluid dynamics*. Cambridge University Press, 2000.

[28] S. Watanabe, A. Tero, A. Takamatsu, and T. Nakagaki. *Traffic optimization in railroad networks using an algorithm mimicking an amoeba-like organism, Physarum plasmodium*. Biosystems, 105(3):225–232, sep 2011.

[29] M. Krishnamoorthy and B. Krishnamurthy. *Fault diameter of interconnection networks*. Computers & Mathematics with Applications, 13(5):577–582, 1987.

[30] J. Jaromczyk and G. Toussaint. *Relative neighborhood graphs and their relatives*. Proceedings of the IEEE, 80(9):1502–1517, 1992.

[31] G. T. Toussaint. *The relative neighbourhood graph of a finite planar set*. Pattern Recognition, 12(4):261–268, 1980. arXiv:9809069v1.

[32] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. *SNDlib 1.0-Survivable Network Design Library*. Networks, 55(3):276–286, 2010.

"He who would search for pearles must dive below."

— John Dryden (1631 - 1700)

B

The Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS): Exploiting Symmetry for Faster Subgraph Enumeration

In this final chapter, we take a big step back from the ITS application context and focus on more general network analysis. One of the key strengths of networks is their modular scalability, allowing networks to be comprised of smaller networks, which in turn can be composed of even smaller networks and so on... A network consisting of a lot of subnetworks can be analysed by studying its individual components, thereby abstracting away some of the complexity of the smaller components and allowing more high level analysis. This chapter presents an algorithm to find and enumerate subnetworks, matching a predefined network structure, in a larger network. The proposed algorithm exploits the inherent symmetries in the network structure to reduce search space and speed up network querying.

⋆ ⋆ ⋆

Maarten Houbraken, Sofie Demeyer, Tom Michoel, Pieter Audenaert, Didier Colle, Mario Pickavet

Published in PLOS One, May 2014.

Abstract Subgraph matching algorithms are used to find and enumerate specific interconnection structures in networks. By enumerating these specific structures/ subgraphs, the fundamental properties of the network can be derived. More specifically in biological networks, subgraph matching algorithms are used to discover network motifs, specific patterns occurring more often than expected by chance. Finding these network motifs yields information on the underlying biological relations modelled by the network. In this work, we present the Index-based Subgraph Matching Algorithm with General Symmetries (ISMAGS), an improved version of the Index-based Subgraph Matching Algorithm (ISMA). ISMA quickly finds all motif instances in a network by intelligently exploring the search space and taking into account easily identifiable symmetric structures. However, more complex symmetries (possibly involving switching multiple nodes) are not taken into account, resulting in superfluous output. ISMAGS overcomes this problem by using a customised symmetry analysis phase to detect all symmetric structures in the network motif subgraphs. These structures are then converted to symmetry-breaking constraints used to prune the search space and speed up calculations. The performance of the algorithm was tested on several types of networks (biological, social and computer networks) for various subgraphs with a varying degree of symmetry. For subgraphs with complex (multi-node) symmetric structures, high speed-up factors are obtained as the search space is pruned by the symmetry-breaking constraints. For subgraphs with no or simple symmetric structures, ISMAGS still reduces computation times by optimising set operations. Moreover, the calculated list of subgraph instances is minimal as it contains no instances that differ by only a subgraph symmetry. An implementation of the algorithm is freely available at https://github.com/mhoubraken/ISMAGS.

B.1 Introduction

In modern society, technology has been applied to create and study numerous advanced systems in various fields as biology, sociology, informatics and others. To understand their internal dynamics, many of these systems can be modelled using graph theory. By interpreting the systems as graphs of interconnected components, a vast array of network processing methods enables detailed analysis of the underlying, fundamental properties.

More specifically in biology, graphs are very well suited to model interactions between different proteins. A graph can be constructed by modelling proteins and interactions among them as nodes and edges respectively. A powerful analysis technique is described in [1] and consists of finding *network motifs* in the graph.

These network motifs denote small interactions patterns between several proteins that are unusually more present in the graph than expected by chance. They can be modelled as small subgraphs which can then be searched in the larger network representing all known interactions between all proteins. By discovering these network motifs, our understanding of the underlying mechanisms of the network can be improved.

To find these network motifs, several tools and algorithms have been developed. Mfinder [2] was one of the early tools to mine graph data for network motifs. Similarly, the FANMOD [3] tool was developed which, compared to Mfinder, improves performance by using the RAND-ESU algorithm [4]. It uses unbiased sampling of subgraphs to speed up the calculations and includes isomorphism tests by using the Nauty [5] isomorphism tools which offer a description of the internal symmetry of the subgraphs. More advanced network motif finding techniques, focusing on graph properties and data structures, are proposed in [6] and [7]. G-Tries [6] are multi-way trees that encode the set of subgraphs/network motifs to be found in a single data structure. When two subgraphs that have to be enumerated have a common substructure, the matching for the substructure can be done simultaneously, speeding up queries significantly compared to doing both searches separately. In contrast to speeding up the network analysis by combining all different tests, Grochow and Kellis [7] optimise the individual subgraph matches by generating symmetry-breaking rules to prune the search space. By incorporating the symmetry of the subgraph in their search, they reduce the search space exploration and speed up queries.

The algorithms mentioned above use subgraph enumeration to find network motifs in the network. However, a different but related network analysis approach [8] is based on calculating graphlet degree distributions. A *graphlet* is a small graph for which the instances in a larger network will be counted. However, contrary to network motifs, if in the graphlet no edge exists between 2 nodes, no edge is allowed in the graphlet instance. While the network motif analysis consist of finding unusually frequent subgraphs, the graphlet-based analysis aims at characterising entire graphs by counting the occurrences for each graphlet from a predefined set. Similar to a node degree distribution, the counts form a distribution that represents the structure of the network in terms of graphlets. As with network motif analysis, the graphlet analysis heavily relies on the enumeration of the graphlet instances which should be optimised.

While the discussed algorithms so far were developed to analyse full networks (by using network motifs and graphlets), the aim of this paper is to present a general subgraph matching algorithm. This algorithm can be used on its own to count or enumerate all specific occurrences of a subgraph in a larger network but can also be used as a building block for a full network analysis algorithm. Such an algorithm needs to be carefully designed as the subgraph isomorphism problem

is proven to be NP-complete [9, 10]. As network modelling is used in various applications, the subgraph isomorphism problem has many variants and several classes of algorithms exist for solving it. In this paper, we focus on exact algorithms for which a strict correspondence between the specified subgraph and the requested instances in the graph is required. Well-known algorithms in this class are the Ullmann [11], the VF [12] and the VF2 [13] algorithms. Ullmann uses a matrix-based representation of the search space and iteratively prunes uninteresting branches in the search tree. Pruning is done by applying a refinement procedure to eliminate candidate nodes (for mapping to a subgraph node) based on the neighbours of the candidate and the required connectedness to the neighbours of the subgraph node. While the Ullmann algorithm is versatile, as it can be used in a wide range of isomorphism problems, it is matrix-based, which causes high memory requirements. Less memory is required by VF and VF2 algorithms which are graph-based. These algorithms search the network by creating an initial partial mapping between the source graph (= the large network) and the subgraph (= the network motif) and iteratively generating candidate pairs to be added to the mapping. Aside from speeding up the search, the graph modelling in VF2 significantly reduces the memory requirements as it only requires $O(N)$ memory while Ullmann requires $O(N^3)$. As biological networks tend to be very large (millions of nodes in some applications), reducing the memory requirements allows for a greater applicability.

In previous work, the Index-based Subgraph Matching Algorithm (ISMA) [14] was presented and compared against the above-mentioned subgraph matching algorithms. Like VF2, ISMA also searches the source graph for subgraph instances by creating a partial subgraph-to-graph-node mapping and expanding it iteratively. However, ISMA intelligently determines the order in which the partial mapping is expanded and avoids unnecessary computations. These optimisations greatly reduce the search space and speed up query times. In this paper, we introduce the Index-based Subgraph Matching Algorithm with General Symmetries (ISMAGS) in which search space size and query times are further reduced by incorporating the internal symmetry of subgraphs as constraints into the algorithm. Our constraint-based pruning is similar to that in [7] in which the breaking of the symmetry is done by iterating the symmetry analysis during the search. Based on the partial mapping constructed at that point, constraints are generated to avoid exploring symmetric parts of the search tree. However, the symmetry analysis in [7] is repeated several times and requires generating an exhaustive list of isomorphisms of the subgraph. In ISMAGS, only one symmetry analysis is needed to obtain a compact set of generating permutations and constraints to break the symmetry. Compared to [7], we also present results for larger networks with multiple edge types.

In the rest of this paper, we first briefly outline the ISMA algorithm and its functionality for incorporating simple symmetric structures. As only basic symmetric relations are incorporated by ISMA, we then continue by presenting our approach to incorporating symmetry in the search tree. After explaining how ISMA deals with symmetry, the symmetry detection in ISMAGS is presented and validated in a group-theoretical context. Subsequently, we show the derivation of the symmetry-breaking constraints and integrate them into the global algorithm. We then show the performance gain of ISMAGS over ISMA for multiple networks with various properties (size, edge types) and various subgraphs. We also compare against the VF2 algorithm and (part of) the Grochow-Kellis algorithm (as the full Grochow-Kellis algorithm was developed to find network motifs while ISMAGS is only concerned with the subgraph enumeration method).

B.2 Methods

The following section contains a brief introduction to the core aspects of ISMA, followed by closer examination of its internal symmetry handling. Next, the core features of ISMAGS to take into account all symmetric structures are presented along with a description of the global algorithm.

B.2.1 ISMA

In previous work [14], ISMA was developed to find matches for composite network motifs (subgraphs with type-annotated edges) in large graphs by dynamically optimising the order in which nodes are investigated during the search process. More formally, the algorithm searches in a graph $G = \{V, E\}$ with V denoting the set of nodes and E denoting the set of edges. Each edge $e \in E$ can be represented by a triplet (u, v, t) with u and v the start and end node respectively and the type t of the edge, defining properties such as whether it is directed or undirected.

Adopting the terminology of [14], a subgraph *SG* is defined as a string of tokens representing the specific subgraph topology. As (anti-)parallel edges are not allowed, a subgraph of s nodes has a maximum of $K = \frac{s(s-1)}{2}$ edges and can be represented by a string of K tokens, with each token encoding the type of the edge (including the direction) at that position. The first token in a subgraph string denotes the edge going from node 1 to node 2, the next 2 tokens denote the edges going from node 1 and 2 to node 3 and so on. The string of tokens "ABCDEF..." thus denotes a subgraph with an edge of type A from node 1 to node 2, an edge of type B from node 1 to node 3, an edge of type C from node 2 to node 3, an edge of type D from node 1 to node 4 and so on. A more elaborate description is given in [14].

In ISMA, the main goal is to reduce the search space by carefully selecting the next node to be matched. At any given stage in the search process, a partial mapping of graph nodes (in G) to subgraph nodes (in *SG*) is maintained. Initially, the mapping is empty as no nodes have been matched. The algorithm then selects a subgraph node based on the number of candidate nodes in G that can be mapped on that node. At the start of the algorithm, the candidate set for each node is constructed based on the edges arriving/departing in/from that node. For example, if the first node sgn_1 in the subgraph has an outgoing edge of type A and an incoming edge of type B, the candidate set C_1 is calculated as the intersection of the set of nodes in G with outgoing edges of type A and the set of nodes in G with incoming edges of type B. The subgraph node (sgn_s) with smallest candidate set is then selected to investigate next.

Once sgn_s is found, the (partial) mapping is expanded by iteratively mapping every graph node n in its candidate set to sgn_s. Mapping n to sgn_s introduces new constraints for the rest of the subgraph instance: if a subgraph node sgn_k is connected to sgn_s with an edge of type B, the graph node mapped to sgn_k also has to be connected to n with an edge of type B. To select the next node to investigate, the constraints are incorporated in the candidate sets by intersecting the old candidate sets with the neighbour set of the newly mapped node n. The mapping process can then repeat for each node in the subgraph. By iteratively mapping and backtracking, all subgraph instances in the graph are found.

The subgraph node selection method is optimised in ISMA to avoid unnecessary set operations. As described above, the candidate set C_i is calculated as the intersection of a number of other sets. Instead of calculating all C_i sets (*sg* in total) each requiring intersecting a few sets to find the smallest candidate set, ISMA keeps track of the smallest set for each subgraph node. The size of this set is a heuristic estimate for the size of C_i. The algorithm is further optimised by using custom data structures for set operations.

B.2.2 Symmetry in ISMA

While ISMA can find all subgraph instances in a graph for any subgraph, it was designed to exploit symmetric properties. While a brief description of the symmetry-handling in ISMA is given here, the reader is referred to [14] for the full approach. Two nodes have a *reflection symmetry* if and only if they can be switched without changing the subgraph topology. If a subgraph contains two reflection symmetric nodes sgn_a and sgn_b, the graph nodes a and b, mapped to sgn_a and sgn_b respectively, can be switched with the result being a valid subgraph instance. This property can be exploited and allows to only examine half of the search space.

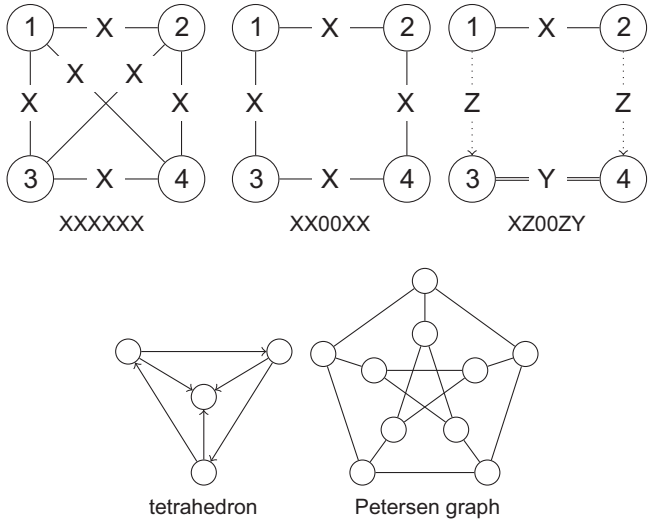
The symmetry is exploited in ISMA by adding extra constraints to the candidate set generation. If subgraph node sgn_a is present in the partial mapping, the

algorithm takes this into account when mapping sgn_b. ISMA will prohibit nodes, that were previously mapped to sgn_a, to be considered for mapping onto sgn_b.

When the candidate set C_a of subgraph node sgn_a is determined, it will consist of all nodes that are valid to be mapped onto sgn_a and by symmetry also onto sgn_b. As described above, ISMA will iteratively add every node in C_a to the partial mapping and continue mapping the rest of the nodes. Every time a new node from C_a is examined, it is removed from C_a before the search is continued. When the candidates for sgn_b are determined, ISMA will use C_a as a constraint to force nodes in C_b to be in C_a. This will avoid generating the symmetric counterparts of the subgraph instances. If $[X, a, Y, b, Z]$ was previously examined, its counterpart $[X, b, Y, a, Z]$ will not be examined as, at that time, a will no longer be in C_a and therefore not in C_b.

The same basic principle is also applied to cyclic rotations. A subgraph contains a *cyclic rotation symmetry* if and only if it has a sequence of nodes that can be shifted without changing the subgraph configuration. An example can be seen in the "XX00XX" subgraph in Fig. B.1.

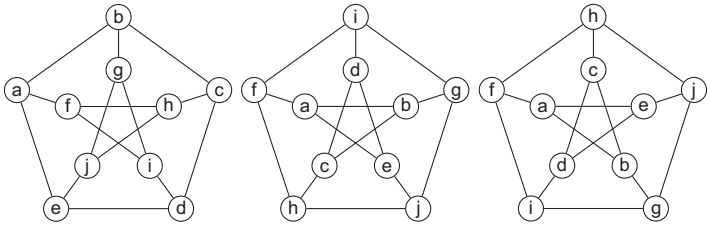
Figure B.1 Subgraph examples. In "XXXXXX", every node is symmetric to every other node as, in a clique, all nodes can be swapped. The "XX00XX" graph is symmetric as it has rotation symmetry (all nodes can be shifted in the ring) and reflection symmetry (top two nodes can be switched with bottom two nodes for example). The "XZ00ZY" graph is also symmetric as the same configuration is obtained when node 1 is switched with node 2 and node 3 with node 4. The tetrahedron and the Petersen graph [15] have more complex symmetric structures.



B.2.3 Symmetry in ISMAGS

While the symmetry handling approach in ISMA performs well for small subgraphs with small reflection or rotational symmetries, it cannot efficiently tackle larger subgraphs with more elaborate symmetric structures. ISMA was only optimised for simple symmetric structures that can be easily detected like reflection symmetries between 2 nodes (= 2 nodes that can be switched) or ring structures (= rotation symmetries). It eliminates similar subgraph instances induced by these symmetries but does not handle larger symmetric structures (consisting of multiple nodes being switched) or more complex symmetric structures (in which nodes can be part of multiple symmetric structures and the symmetric properties are less easily detected). As the symmetry analysis in ISMA was limited and did not extract complex multi-node symmetries, these were not taken into account which means that multiple similar subgraph instances induced by these symmetries will be returned. Fig. B.2 shows valid subgraph instances that differ by only a permutation of the subgraph nodes. As the degree of symmetry increases, listing all possible permutations becomes very time-consuming. A more efficient approach would be to avoid finding the permuted instances by reducing the search space.

Figure B.2 Some permuted instances of the Petersen graph.



This section deals with the modified approach in ISMAGS to successfully handle all symmetric structures in a subgraph. The main idea of ISMAGS is to detect the symmetric properties in the subgraphs and convert them to pruning rules for the search space. Detecting the symmetries is detailed in the next paragraph, followed by the symmetry-breaking approach and a description of the data structures used. In general, *breaking symmetry* means that the information of the symmetries is used to develop rules or constraints to simplify the search and speed up calculations. By fully breaking the symmetry, the set of subgraph instances returned by ISMAGS is minimal as a single subgraph instance will only be exported once while similar subgraph instances induced by the subgraph symmetry are omitted.

B.2.3.1 Symmetry detection

The first step in ISMAGS is to determine the symmetric properties of the subgraph under examination. While the subgraph isomorphism problem is NP-complete,

several basic techniques have been developed to minimise the required work. These techniques and how they are used in ISMAGS to develop a custom symmetry-breaking approach are explained next.

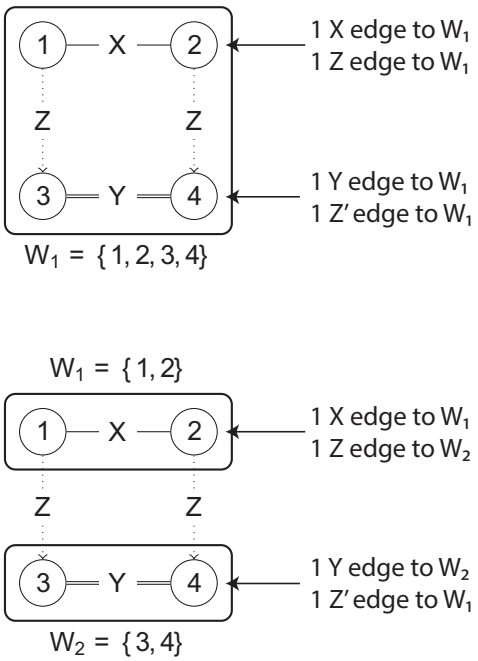
Subgraph partitioning. The basis for the symmetry detection is derived from Nauty [5]. The basic mechanisms are explained here but the reader is referred to [5, 16] for more details. The analysis starts by grouping the subgraph nodes sgn_i based on their incoming and outgoing edges as only nodes with similar properties could be symmetric to each other. The nodes are first grouped into an ordered partition $\pi = [W_1|W_2|...|W_p]$ with every node in a cell W_i having the same number of outgoing/incoming edges to/from each of the other cells.

Fig. B.3 illustrates how these partitions are formed. In the initial partition, all nodes are put in the same cell W_1. Every node is then analysed by determining the source/target cells of its edges. Nodes 1 and 2 both have 1 X-edge to a node in W_1 and 1 outgoing Z-edge to another node in W_1. However, the top 2 nodes have different edge properties compared to the bottom 2 nodes. If nodes within one cell do not have the same properties, the partition needs to be *refined*. Cells containing nodes with different properties are split to ensure partition validity. The top nodes of the subgraph are separated from the bottom nodes by introducing a new cell. This leads to the partition in the bottom of Fig. B.3. The nodes are then again analysed by their edges as the introduction of new cells can require splitting up other cells. This process is repeated until all nodes in the same cell have the same properties.

Ordered partition pair. The actual subgraph symmetry analysis uses 2 partitions $\pi_t = [T_1|T_2|...|T_t]$ and $\pi_b = [B_1|B_2|...|B_b]$ to analyse the subgraph. An example of the analysis can be found in Fig. B.4. The 2 partitions together from an Ordered Partition Pair (OPP). This OPP will be used to investigate symmetric structures by simultaneously refining both partitions. If a subgraph has symmetric structures, the refinement will have multiple branching points which lead to the symmetries in the subgraph. By exploring all branches during the analysis, all symmetries can be found.

Coupling. The 2 partitions in the initial OPP are identical and follow from the initial partitioning of the subgraph. Fig. B.4 shows the symmetry-breaking algorithm for the "XX00XX" subgraph. The partitioning in the initial OPP at the top of the search tree consists of a single cell as all nodes have 2 edges to other nodes. The different branches in the OPP search tree are then separately investigated by selecting one of the subgraph nodes in one of the cells of π_t and mapping it to all subgraph nodes in the corresponding cell in π_b. In the remainder of this work, each such mapping is referred to as a *coupling*. When coupling, the nodes are selected in the order of increasing node ID. The node with the smallest ID among all unmapped nodes gets selected first, both in the top partition (of which

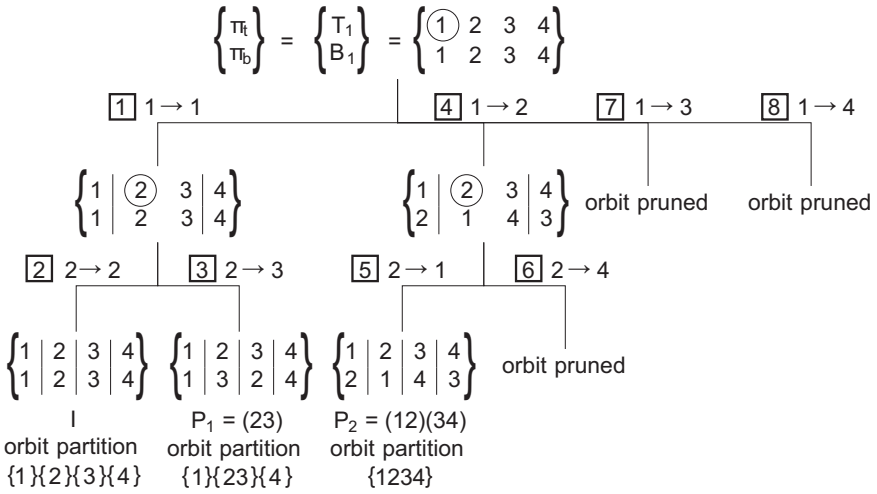
Figure B.3 Example of subgraph refinement. The top figure shows the initial partitioning in the "XZ00ZY" graph in which all nodes are in the same cell. However, nodes 1 and 2 have an outgoing Z edge while nodes 3 and 4 do not. This indicates the partition needs to be refined. Nodes 3 and 4 are put in a separate cell as shown in the bottom figure.



only 1 node is chosen) as in the bottom partition (for which coupling iterates over all nodes in order of increasing ID).

Recursive refinement. A coupling operation thus maps a node (sgn_t) from a top partition cell to a node (sgn_b) in the corresponding bottom partition cell. This is done in the partitions by putting sgn_t and sgn_b in a newly created cell in their respective partitions. The coupling is followed by a *refinement* operation on each of the partitions (top and bottom). As mentioned above, each partition groups nodes with identical properties. By introducing a new cell (for sgn_t or sgn_b), the grouping might no longer be accurate. Nodes with edges to/from sgn_t now have edges to/from the new cell while previously those edges were to/from the original cell of sgn_t (analogously for the bottom partition with sgn_b). This change in properties needs to be incorporated in the partition, possibly yielding more cells to ensure that all nodes in one cell have the same properties. The partition is refined until it is completely valid again. Note that while the refinements of π_t and π_b are done separately, each top partition cell will correspond to a bottom cell. The

Figure B.4 Subgraph symmetry analysis of "XX00XX". The branches are denoted with the applied coupling. The boxed numbers indicate the order of tree traversal, with a depth-first exploration, according to the *smallest node first* coupling. The initial partition has all nodes in the same cells T_1 and B_1. The first coupling $1 \rightarrow 1$ splits up the cells in both partitions in 3 cells (separation of cells denoted by $|$). When a permutation is found, the orbit partition is updated as shown. Orbit pruning is used to reduce the required computations as explained in text.



coupling and refinement operations are done recursively until all nodes have their own cell. At that point, there is a one-on-one correspondence between nodes in the top partition and nodes in the bottom partition, yielding a valid permutation.

In the example of Fig. B.4, subgraph node 1 from T_1 is coupled to the 4 possibilities in B_1. This splits up T_1 and B_1 as node 1 gets its own cell because of the coupling. It also leads to the further refinement of the initial OPP as in both partitions for nodes 2 and 3, one of the incoming edges is coming from the new cell while for node 4 that is not the case. As shown in Fig. B.4, the recursive refinement leads to the full subgraph analysis.

The key strength of using the OPPs in the analysis is that the discovered symmetry relations can be used to prune and speed up the analysis. When the coupling of a node in π_t to a node in π_b leads to 2 partitions with a different number of cells, the configuration can be discarded as no symmetries can be found. Even more extensive pruning is used in the original Nauty algorithm and its successors but is simplified here as the analysis in ISMAGS is combined with symmetry-breaking constraints (see next section) that modify the search process.

Orbit pruning. Orbit pruning [5] is a group theoretical optimisation technique to narrow down the search space. During the analysis of the subgraph, a set of generating permutations is built for the automorphism group A of the subgraph. In the example of Fig. B.4, the set of generating permutations consists of P_1 and P_2. The automorphism group A is a permutation group and will act on the set S of all possible permutations on the subgraph nodes. The effect of A can now be analysed in 2 ways.

First, the effect of permutations $P_k \in A$ on individual subgraph nodes $sgn_i \in V$ is considered. Starting from an element $e \in S$, the permutation permutes the individual nodes to different positions. In general, the element e is transformed in $P_k(e)$ by permuting the individual subgraph nodes in the element.

$$\begin{aligned} e = [sgn_a, sgn_b, ..sgn_s] \rightarrow P_k(e) = [sgn_x, sgn_y, ...sgn_z], \\ \{a, b, ..., s\} = \{x, y, ..., z\} \end{aligned} \tag{B.1}$$

The image $P_k(sgn_i)$ of sgn_i under P_k can then be used to define the *orbit partition*. This is a partition of V into disjoint cells C_j for which holds that

$$V = \{C_1|C_2|...|C_z\} \tag{B.2}$$

$$\forall sgn_i \in C_j, \forall P_k \in A : P_k(sgn_i) \in C_j \tag{B.3}$$

More intuitively, when an automorphism is applied, a node can only be mapped to itself or another node in its orbit partition cell. Every time a permutation is found during the symmetry detection, the orbit partition is updated by merging the orbit partition cells of nodes that can be mapped to each other. In Fig. B.4, $P_1 = (23)$ is found as an automorphism. The image $P_1(2)$ of node 2 is node 3 (and vice versa) and so they will share the same cell in the orbit partition.

For the second analysis of the effect of A, consider the relations between the elements of S. Given an element e of S, its *orbit* is the set of all elements of S that can be found by combining all permutations in A to e. The orbits themselves form a partition of the set S of all possible permutations on the subgraph nodes.

$$S = \{O_1|O_2|...|O_z\} \tag{B.4}$$

$$\forall e \in O_i, \forall P_k \in A : P_k(e) \in O_i \tag{B.5}$$

Orbit pruning relies on this partitioning to prune parts of the search space. As one element suffices for generating the entire orbit (by applying the generating permutations), the analysis can omit parts of the search space that would result in redundant generators. The orbit partition is used to detect these cases and to abort the search. A more detailed application of orbit pruning can be found in [17].

Without orbit pruning, the subgraph symmetry analysis described above continues finding all permutations after the necessary set of permutations is already

exported. As only that set is needed to generate all permutations, orbit pruning is used to reduce the search space. In the example of Fig. B.4, the orbit partition gets updated when permutation $P_1 = (23)$ is found and again when $P_2 = (12)(34)$ is exported. For P_1, node 2 and node 3 are put in the same cell in the orbit partition while P_2 merges the cell of node 2 with the cell of node 1 and the cell of node 3 with the cell of node 4. This results in all nodes being in a single cell. When the subgraph analysis is continued after finding P_2, node 2 in the top partition would be coupled to node 4 in the bottom partition but as node 2 and 4 share the same cell in the orbit partition, the coupling can be pruned. Backtracking further in the analysis would lead to top node 1 being recoupled to bottom nodes 2, 3 and 4 but this can be omitted analogously.

B.2.3.2 Symmetry breaking

The symmetry detection in ISMAGS results in a set of permutations of the subgraph nodes and an orbit partition. The permutations can be applied on any valid subgraph instance to produce other valid instances. To avoid generating and exporting instances that can be obtained through permuting previously found instances, the symmetry needs to be broken. To explain the symmetry breaking in ISMAGS, some group theoretical concepts first need to be introduced.

Stabilisers. In general, permutations in a permutation group G act on sequences of nodes by switching/swapping nodes to different positions in the sequence. However, some permutations do not change all nodes and leave some nodes on their original position. These permutations can be used to define *stabilisers*. For every node i, the stabiliser i_G is defined as

$$i_G = \{P_k | P_k(i) = i, P_k \in G\} \tag{B.6}$$

For the example of the "XX00XX" subgraph with generating permutations $\{(23), (12)(34)\}$, the stabiliser 1_G of node sgn_1 is $\{(1)(2)(3)(4), (1)(23)(4)\}$ as these permutations all map node sgn_1 to itself.

Stabiliser chains. Using the concept of stabilisers, a *stabiliser chain* of (sub-) groups $G_0, G_1, ..., G_s$ is defined for a permutation group G, acting here on the set of subgraph node permutations. Each group in the chain is a subgroup of the previous ($G_i \subset G_{i-1}$) starting with $G_0 = G$. Formally, the groups are defined as

$$\forall i = 1..s : G_i = i_{G_{i-1}} \tag{B.7}$$

More intuitively, the permutations in G_1 are those permutations that do not change node sgn_1 while the permutations in G_2 are those permutations in G_1 that do not change node sgn_2 and thus leave 2 nodes unchanged.

Coset representatives. The *coset representative set* C_i of a subgraph node sgn_i is defined as the set of subgraph nodes sgn_x to which sgn_i can be mapped in G_{i-1}.

$$\forall i = 1..s : C_i = \{sgn_x | P(sgn_i) = sgn_x, P \in G_{i-1}\} \tag{B.8}$$

Given G_1 in subgraph "XX00XX", $C_2 = \{sgn_2, sgn_3\}$ as sgn_2 can be mapped to any subgraph node in C_2 by the permutations in G_1.

Given the above definitions, the subgroup G_i consists of those permutations that leave the first i nodes unchanged. As shown in [18], stabiliser chains can be converted to symmetry-breaking constraints that fully break the symmetry of the subgraph. However, contrary to [18], the G_i groups are not fully generated in ISMAGS. To generate the constraints, only the coset representatives for each subgraph node are needed. These coset representatives are generated during the symmetry analysis phase.

Determining coset representatives. Recall that during the symmetry analysis, nodes are coupled according to increasing ID. In the search tree, constructed by the couplings, the first leaf node to be reached is the identity permutation. When backtracking starts, the couplings are undone by replacing the last coupling, say $sgn_k \rightarrow sgn_k$, with a new coupling, say $sgn_k \rightarrow sgn_l$. Undoing the mapping can be interpreted as looking for permutations that leave the first $k-1$ nodes in place but permute the remainder of the subgraph nodes. If such a permutation is found, it belongs to the G_{k-1} group. In Fig. B.4, when $P_1 = (23)$ is found, it is a permutation leaving the first node unchanged and thus belongs to the G_1 group. As detailed above, when a permutation is found, the orbit partition is updated. After P_1 is found, the orbit partition cell O_2 of node sgn_2 will be merged with the cell of node sgn_3.

Subsequently, the algorithm continues by backtracking and uncouples $sgn_2 \rightarrow sgn_3$. As all possible couplings for sgn_2 are tested, ISMAGS further backtracks and uncouples $sgn_1 \rightarrow sgn_1$. However, at this point in the analysis, the coset representative set C_2 can be determined. Orbit partition cell O_2 only contains subgraph nodes sgn_l that can be mapped to sgn_2 without remapping the first node as the first node has thus far remained mapped to itself due to the coupling. Cell O_2 thus corresponds to the set of coset representatives of sgn_2.

Following the example above, the coset representative set C_i is found for every node as an intermediate orbit partition cell. Coset representative set C_i is set equal to its orbit partition cell when all possible mappings $sgn_i \rightarrow sgn_j$ are evaluated and $sgn_{i-1} \rightarrow sgn_{i-1}$ is to be undone next. While the orbit partition is updated every time a permutation is found, the C_i sets are not.

Note that not all nodes will have coset representatives generated. During the creation of the initial chain of couplings that results in the identity permutation, some nodes are mapped to themselves by the couplings while other nodes are mapped to themselves by the refinement procedure. For the latter nodes, no coset

representatives will be generated as, with the first nodes fixed, they can only be mapped to themselves.

For the example in Fig. B.4, C_2 is set as $\{2, 3\}$ after P_1 is found, right before $sgn_1 \rightarrow sgn_1$ is undone. Set $C_1 = \{1, 2, 3, 4\}$ is found similarly at the end of the subgraph symmetry analysis, after $sgn_1 \rightarrow sgn_4$ is investigated. Sets C_3 and C_4 can be omitted as they can only be mapped to themselves if sgn_1 and sgn_2 need to remain fixed.

Generating symmetry-breaking constraints. To break the symmetry in the subgraph, the coset representatives are converted into constraints on the IDs of the graph nodes mapped to the subgraph nodes, as shown in [18]. For every subgraph node sgn_j in the coset representative set C_i of subgraph node sgn_i, a constraint is introduced to force the ID of the graph node mapped to sgn_j to be higher than the ID of the graph node mapped to sgn_i. More formally, the following constraints are introduced to be used during the node mapping in ISMAGS.

$$\forall i, j : sgn_j \in C_i, i \neq j : ID_i < ID_j \tag{B.9}$$

While the constraint generation is done as in [18], ISMAGS does not require explicitly generating the stabiliser chains (introduced by [19]) as only the coset representatives are necessary and found during subgraph symmetry analysis.

The approach to breaking symmetry in ISMAGS begins with a subgraph symmetry analysis derived from Nauty [17]. In general, Nauty generates a set of generating permutations for the automorphism group of the subgraph. However, the set of generating permutations generated is not necessarily unique. For the "XXXXXX" subgraph in Fig. B.1, a generating set of permutations could be $Q_1 = \{(12), (123), (1234)\}$ while $Q_2 = \{(12), (23), (34)\}$, generated by ISMAGS, is equally as valid for generating the full set of permutations. By tuning the order of the node coupling, ISMAGS embeds the stabiliser chains into the search process. The permutations found are generators for as many subgroups G_i as possible while the coset representatives can readily be found. This eliminates the need for explicitly generating all possible permutations, the stabiliser chains and coset representatives as in [18].

B.2.4 Integrating symmetry detection and symmetry breaking with ISMA

With the symmetry-breaking constraints described in the equation above, a full description of ISMAGS can now be given. While ISMAGS reuses the basic principles of ISMA to limit the search space, it goes much further in the subgraph analysis and pruning. A pseudocode description of the different steps is given in Algorithm B.1. The actual subgraph instances are found and exported in the mapNodes function.

Algorithm B.1: findSubgraphInstances(Graph g, Subgraph sg)

```
1:  Set<Constraint>constraints ← analyseSubgraph(sg);
2:  NodeListHandler[] candidates ← candidate node lists (to be intersected) for
    each subgraph node;
3:  for SubgraphNode sgn in sg do
4:     for Edge e leaving/arriving in sgn do
5:        t ← type of edge e;
6:        S ← list of graph nodes in g that have an edge of type t;
7:        add S to candidates[sgn];
8:     end for
9:  end for
10: SubgraphNode sgn ← subgraph node with the smallest candidate node sublist;

11: NodeList snl ← calculate candidate node list of sgn;
12: mapNodes(sgn, candidates, constraints);
```

The search for all instances starts with the analysis of the subgraph specification for symmetric properties. This analysis is detailed in the previous sections and results in a set of constraints between the IDs of the graph nodes in a mapping. These constraints are stored for each subgraph node and used for determining the candidates for a specific subgraph node. After subgraph analysis, the search for subgraph instances, similar to ISMA, begins on line 3 in Algorithm B.1 by creating the first candidate node lists based on the subgraph configuration. As in ISMA, the candidates for a subgraph node are determined by intersecting collections of nodes. However, the collections in ISMAGS are lists of ordered nodes based on node ID. Sorting of the lists of nodes and neighbours only needs to be done once for every network as the networks can be stored with the lists sorted. Using ordered lists accommodates quick subset selection (detailed further below) during node mapping. The candidates for a subgraph node are determined based on the edges in the subgraph specification. For each of its edges, the corresponding list of graph nodes is added to a set of lists. Once all lists are known, their intersection gives all graph nodes that have edges of the required types. Note that, as in ISMA, the intersection of the starting sets is delayed until after the subgraph node is selected to avoid calculating intersections that are not used.

Once the initial subgraph node is determined, the candidate node list is generated (see line 11 in Algorithm B.1). This candidate node list is calculated with a linear sweep over the different (ordered) lists while all other intersections in ISMAGS are calculated by checking node membership to all lists (as in ISMA). As node lists are large at this point, a linear sweep is more efficient than node-by-node set membership tests.

Algorithm B.2: mapNodes(SubgraphNode sgn, NodeListHandler[] nodelists, Set<Constraint>constraints)

```
1:  List<Node>snl ← getCandidates(sgn, nodelists);
2:  for Node n in snl do
3:     map n to sgn;
4:     if subgraph instance is complete then
5:        export instance;
6:     else
7:        for Edge e arriving/leaving sgn do
8:           t ← type of e;
9:           q ← origin/destination of e;
10:          neighbours ← neighbours of n by type t;
11:          addNeighbourList(nodelists[q], neighbours);
12:       end for
13:       nextSubgraphNode ← determineNextSubgraphNodeToProcess(
          nodelists, constraints);
14:       mapNodes(nextSubgraphNode, nodelists, constraints);
15:    end if
16:    unmap n to sgn;
17: end for
```

The mapNodes function is very similar to the approach in ISMA and recursively maps graph nodes to subgraph nodes to find all subgraph instances. Every time a graph node is mapped to a subgraph node, the neighbours of the graph nodes are taken into account as new constraints for the remaining unmapped nodes. If graph node a is mapped to subgraph node sgn_i and sgn_i has an edge of type e to subgraph node sgn_j, the candidate graph node for sgn_j needs to have an edge of type e from a. Note that the direction of the edges is taken into account in the edge type. Once the lists of neighbouring nodes are added to the constraint sets, the next node to investigate is determined.

The next subgraph node to examine is determined on line 13 heuristically as in ISMA. The intersection of lists is delayed to avoid unnecessary work. Instead, the lists that need to be intersected are stored separately per subgraph node. The size of the intersection is estimated by the size of the smallest list of that subgraph node. This is an upper bound on the actual size and trades off a slightly larger search space for less list intersections. To further optimise the calculation of the candidate list, ISMAGS disregards the lists introduced during initialisation (see Algorithm B.1). These lists are generally very large (they list all nodes with an edge of the correct type) and contain little useful information, as the lists only indicates the presence of an edge which is verified each time a node is mapped. Omitting the lists results in shorter computation times during intersection with only a limited increase in search space size.

The key difference with ISMA is that when a subgraph node is selected for examination, the actual intersection is calculated using the symmetry-breaking constraints. The constraints are combined with the partial instance constructed so far to determine boundaries for the node ID of the graph nodes to be mapped on the subgraph node. If for the selected node sgn_i a constraint $ID_i < ID_j$ was generated and node sgn_j is already mapped, this gives an upper bound on the ID of a candidate for sgn_i. A lower limit is found analogously by considering the set of subgraph nodes which should have a smaller ID. The boundaries can then be used during intersection, allowing to skip the nodes with IDs outside the allowed range. As the lists to be intersected are sorted, binary search can be used to quickly find the start and ending point of the sublist of valid IDs.

While the symmetry breaking is the main source of the performance gain, additional speed-up could be gained by maintaining extra state. As explained above, the candidate sets are stored in memory as ordered lists to allow quick retrieval of nodes within a specific ID range. When lists are intersected, the intersection is determined by iterating over all nodes (in that valid ID range) and checking for membership of the other lists. In ISMAGS, this checking is done by using binary search on the sorted lists. To speed up this operation, a copy of the list could be maintained in a hash-based set. This would allow to check membership in overall constant time ($O(1)$), whereas binary search requires logarithmic time ($O(log(n))$, n = number of entries in the list). Experiments show that an additional 5% speed-up could be gained by using this optimisation at the cost of almost doubling memory requirements. As memory is often a bottleneck in biological networks, the optimisation was not included in the presented version of ISMAGS.

The basic pseudocode of the subgraph analysis (to generate the constraints) is given in Algorithm B.3. The initial OPP is constructed based on an initial partitioning of the input subgraph as detailed in the symmetry detection section above. The OPP is then recursively refined to find all symmetries (and constraints), as described in the two previous sections. As mentioned above, ISMAGS tunes the order in which the nodes are coupled in the OPPs by always selecting the node with the lowest ID first as shown in lines 5 and 9 of Algorithm B.4. The constraints are generated on line 17 in Algorithm B.4.

B.3 Results

To illustrate the performance of ISMAGS, the algorithm is benchmarked against previously published results and algorithms. After a description of the algorithms and network data use, ISMAGS is compared against its predecessor ISMA to show the effects of the added symmetry breaking and related optimisations. Next, ISMAGS is compared against the VF2 algorithm and the subgraph enumeration algorithm of Grochow-Kellis [7], denoted by GK.

Algorithm B.3: analyseSubgraph(Subgraph sg)

```
1: Set<Permutation>permutations;      // create new set to store permutations
2: Set<Constraint>constraints;        // create new set to store constraints
3: Set<Set<SubgraphNode>>orbits;      // create the initial orbit partition
4: for SubgraphNode sgn_i ∈ sg do
5:    add set {sgn_i} to orbits;
6: end for
7: Partition π ← create initial partition of subgraph sg;
8: OPP opp ← create initial OPP from π;
9: processOPP(opp, permutations, constraints, orbits);
10: return constraints;
```

B.3.1 Algorithms

The ISMA and ISMAGS algorithms were implemented in Java (version 1.6.0_26) while for the VF2 experiments, the VFLibrary (*http://mivia.unisa.it/datasets/graph-database/vflib/*) was used. To perform the GK experiments, the authors of [7] provided the original (Java) code for their algorithm from which the code for subgraph enumeration was extracted. This was necessary as the GK algorithm is a network motif finding algorithm while we present a subgraph enumeration algorithm. An implementation of the ISMAGS algorithm is freely available at https://github.com/mhoubraken/ISMAGS.

The experiments were performed on a single core of an Intel Core 2 Duo P8400 processor clocked at 2.26 GHz with 4 GB of RAM under a 64-bit Windows installation. To remove the influence of memory operations, the reported times exclude the reading of the networks and writing to memory of the subgraph instances found. The times reported in the results thus only pertain to the time needed to look for all possible matches in the search space and, if applicable, the time needed to analyse the subgraph for symmetries. Most of the results were averaged over 1,000 runs. However, some test instances were limited to fewer runs as the long calculation times were prohibitive for more elaborate testing. The number of runs used for averaging is shown along with the results. When fewer runs were used, this is denoted with an asterisk or circle, depending on the number of runs.

B.3.2 Network data

The input networks for the experiments are similar to the networks from [14]. They are denoted as biological, Slashdot and SNAP (based on the source of the data) and their properties can be found in Table B.1. Note that while only results are shown for these 3 types of networks, ISMAGS can be used for subgraph matching in any graph with multiple edge types.

Table B.1: Network properties. For each network, the number of nodes and edges is given. If multiple edge types are present in the network, separate counts are given for each edge type, denoting the number of edges of the specific type as well the number of nodes having an edge of that type. The XYZ-network and the ABZ-network have the same node and edge count as the A- and B-edges are the directed versions of the X- and Y-edges. Similarly, the reduced PGS-network has the same node and edge count as the PGS-network.

Network	#Nodes	#Edges
PGS (reduced)	1,255	6,454
P	887	1,844
G	469	4,051
S	404	659
XYZ/ABZ	15,617	80,405
X/A	4,847	36,391
Y/B	9,599	40,626
Z	5,496	3,388
Slashdot	79,120	469,768
E	37,412	118,755
F	69,998	351,013
Wiki-Vote	7,115	100,762
p2p-Gnutella08	6,301	20,777
p2p-Gnutella30	36,682	88,328
CA-CondMat	23,133	93,439
CA-HepTh	9,875	25,973

Algorithm B.4: processOPP(OPP opp, Set <Permutation>permutations, Set<Constraint>constraints, Set<Set<SubgraphNode>>orbits)

if all nodes are mapped **then**
  add current mapping to *permutations*;
  update *orbits*;
else
  $sgn_i \leftarrow$ subgraph node with the lowest *ID* among the unmapped nodes;
  $T \leftarrow$ cell in top partition which contains sgn_i;
  $B \leftarrow$ cell in bottom partition corresponding to T;
  sort B by increasing *ID* : [l_1,l_2,...,l_b];
  for $j = 1...b$ **do**
    couple sgn_i to l_j;
    $opp_{new} \leftarrow$ refine *opp*;
    processOPP(opp_{new}, *permutations*, *constraints*, *orbits*);
  end for
  if *opp* maps $sgn_k \rightarrow sgn_k, \forall k < i$ **then**
    $C_i \leftarrow$ orbit of sgn_i;
    for $sgn_t \in C_i, i \neq t$ **do**
      add $ID_i < ID_t$ to *constraints*;
    end for
  end if
end if

The biological networks comprise two networks with multiple edge types. The first network pertains to physical (P, undirected), genetic (G, undirected) and signalling (S, directed) interactions between kinases and phosphatases in yeast [20, 21]. The second network consists of protein-protein interactions in yeast (X, undirected, obtained from the BioGRID [22] database), protein-protein interactions in humans (Y, undirected, obtained from the BioGRID and STRING [23] databases), and orthology relations between human and yeast proteins (Z, bipartite, from the InParanoid database [24]). Additionally, both networks were modified to obtain additional test networks. The reduced PGS-network is constructed by interpreting all edges in the PGS-network to be undirected and of the same edge type. The ABZ-network is derived from the input files of the XYZ-network. The X- and Y-edges, specified as "$node_1$ $node_2$" on individual lines in their respective edge file, are interpreted as directed A- and B-edges going from $node_1$ to $node_2$.

The Slashdot network [25] represents "friend" and "foe" relations between users of the technology-centred Slashdot community. A group of friends in which everyone is a friend of each other can be represented as an F-clique subgraph while two friends with a mutual enemy can be represented by a "FEE" subgraph. Note that the edges are assumed to be undirected. Finding subgraphs in this social network is an example of how ISMAGS can be used to mine social data.

The third set of networks consists of some of the networks available in the SNAP database (found at *http://snap.stanford.edu/data/*). The Wiki-Vote network represents votes cast by users during Wikipedia admin elections, the p2p-Gnutella08 and p2p-Gnutella30 are 2 snapshots of the Gnutella peer-to-peer network and the CA-CondMat and CA-HepTh networks are collaboration networks based on co-authorship of papers published in the arXiv repository in the Condense Matter and the High Energy Physics - Theory category, respectively. For these networks, the 3- and 4-node cliques are searched as well as the instances of the subgraph "XxXXXX" (tetrahedron). Note that, during the search for the instances of the cliques, the edges in the networks are considered to be undirected while they are considered to be directed during the search for the instances of the tetrahedron. This difference in interpretation allows to show the performance of the symmetry handling of the algorithms on both directed and undirected networks. The tetrahedron subgraph was selected to be searched for as it contains a relatively simple symmetric structure that was not taken into account in ISMA.

Note that the number of nodes and edges reported in Table B.1 can differ from the counts of the original data source. This is a result of network preprocessing. Aside from removing unconnected nodes and (anti-)parallel edges, the preprocessing also ensured that only 1 edge is present between any pair of nodes. While ISMAGS correctly deals with these issues, the VF2 and GK implementations did not support them. The preprocessed networks are included in the source code of ISMAGS (available at https://github.com/mhoubraken/ISMAGS).

B.3.3 ISMA versus ISMAGS

To show the advantages of incorporating the symmetric information in the search, Table B.2 compares the performance of ISMA to ISMAGS on the biological networks. The *SPRF* varies depending on the subgraph.

For the subgraphs with $SPRF > 1$, the reduction can be attributed to various factors. For the subgraphs with limited symmetry ("SsS","SsG","PGSPGS"), the reduction is due to a quicker termination of uninteresting paths. When the mapping of a node would result in empty candidate lists for an unmapped node, this is detected in ISMAGS before the node is mapped and can quickly be terminated. For the subgraphs with large symmetric structures (Petersen graph of Fig. B.1, XYZ subgraphs), the reduction comes from the symmetry-handling which was not fully exploited in ISMA, showing the benefits of the improved symmetry-breaking approach.

*Table B.2: Comparison between ISMA and ISMAGS on the biological networks. The reported #instances is the number of subgraph instances as exported by ISMAGS. The search space size is the #nodes visited during the search process. SPRF is the search space reduction factor and is calculated as the ratio of the search space size for ISMA to the search space size for ISMAGS. The speed-up factor is defined analogously for the calculation time. All reported timings are averaged over 1,000 runs unless denoted by an asterisk *, in which case only one test was performed as the long computation times were prohibitive for more elaborate testing.*

		Search space (#nodes)			Calculation time (ms)		
	#instances	ISMA	ISMAGS	SPRF	ISMA	ISMAGS	SF
PGS-network							
GGG	9,008	4,520	4,520	1.00	25.91	5.80	4.46
SSS	78	763	761	1.00	1.61	0.24	6.65
SsS	0	190	90	2.11	0.62	0.06	10.24
GPS	47	462	454	1.02	0.99	0.28	3.56
SSG	103	763	761	1.00	1.76	0.44	3.97
SsG	25	190	131	1.45	0.49	0.13	3.79
GGS	294	462	454	1.02	1.29	0.48	2.69
GGP	418	8,571	8,571	1.00	14.96	3.83	3.90
ssG	112	372	419	0.89	0.91	0.25	3.69
PGSPGS	0	391	232	1.69	0.97	0.14	7.02
P0P	24,452	4,575	4,575	1.00	19.59	2.87	6.82
P0P00P	221,290	57,167	53,479	1.07	192.67	30.67	6.28
P0P00P000P	2,570,154	551,837	496,059	1.11	2,142.74	302.55	7.08
Petersen	9,430	1,131,600	616,418	1.84	330,882*	733.25	451

Table B.2: Comparison between ISMA and ISMAGS on the biological networks (continued).

	#instances	Search space (#nodes) ISMA	ISMAGS	SPRF	Calculation time (ms) ISMA	ISMAGS	SF
Reduced PGS-network							
3-clique	10,614	7,709	7,709	1.00	39.50	8.81	4.48
4-clique	11,150	18,323	18,323	1.00	118.69	27.60	4.30
5-clique	7,669	29,473	29,473	1.00	225.49	48.36	4.66
6-clique	3,616	37,142	37,142	1.00	320.71	64.90	4.94
7-clique	1,158	40,758	40,758	1.00	379.00	76.63	4.95
8-clique	226	41,916	41,916	1.00	412.69	84.97	4.86
9-clique	24	42,142	42,142	1.00	431.79	92.21	4.68
10-clique	1	42,166	42,166	1.00	451.08	99.08	4.55
XYZ-network							
XZ00ZY	2,554	23,164	19,095	1.21	36.66	9.58	3.83
XXXZ000Z0Y00ZYY	4,727	73,647	39,572	1.86	152.77	27.54	5.55
ABZ-Network							
AZ00ZB	1,337	11,588	10,859	1.07	20.46	6.02	3.40
AAAZ000Z0B00ZBB	837	15,186	14,190	1.07	36.52	9.73	3.75

For the “ssG” subgraph, a small increase in search space is present. This is due to the omission of list membership tests of the large initial lists of candidates. While omitting these tests allows to map candidates faster, it increases the search space slightly as some graph nodes get examined while they do not have all required edges. However, the speed gain of omitting these tests still outweighs the slight increase.

The SPRF factors around 1 indicate that ISMA and ISMAGS follow the same path through the search space. This is primarily the case when the symmetry in the subgraphs is incorporated in ISMA (e.g.“GGG”, “SSS”, cliques). Interestingly, ISMAGS still reduces query times for these subgraphs due to the list-based implementation of its symmetry breaking. When calculating candidate sets for the subgraph nodes involved in the symmetric structures, ISMA removes nodes from constraint sets before calculating the intersections of its sets. This ensures that no nodes get mapped in symmetric configurations. However, to calculate the intersection, ISMA still needs to intersect the different sets. ISMAGS uses the ID-based constraints derived during symmetry analysis to avoid most work during list intersection. Using the constructed partial mapping and the constraints, ISMA can quickly find the interesting range of nodes in the to-be-intersected lists and ignore the remainder. This reduces the time needed for candidate set generation and improves execution time results. Additionally, as explained above, ISMAGS omits the large initial lists of candidate graph nodes when calculating candidate set lists once the initial node is determined. This reduces calculation time as less list membership needs to be checked.

The speed-up factor in Table B.2 shows the ratio of the calculation time of ISMA to the calculation time of ISMAGS. For most investigated subgraphs, the speed-up factor is larger than 3, indicating that ISMAGS only needs a third of the calculation time of ISMA. While ISMAGS gives a speed-up for the subgraphs previously published [14] due to the more optimised list-based implementation, it was designed to handle more complex symmetries in the subgraphs. This can be seen in the results of the Petersen graph and the line graphs. The Petersen graph [15] is inherently very symmetric as one instance can be permuted in 120 other instances. The line graphs (“P0P”,“P0P00P” and “P0P00P000P”) consist of n nodes, connected by $n-1$ edges, with node n_i connected to n_{i+1} for $i = 1..n-1$. As such, the line graphs correspond to chains of nodes. The symmetry in these graphs is limited to reversing the chains, as every subgraph instances can be read left-to-right and right-to-left. While this is a simple symmetry, it involves multiple nodes being remapped, which was not fully incorporated in ISMA.

An analogous analysis was done on the Slashdot and SNAP networks and can be found in Table B.3. Again, the test instances with $SPRF = 1$ have similar search spaces between the 2 algorithms. The results for the tetrahedron show a $SPRF > 1$, indicating that the symmetric structure in the tetrahedron is taken into account by ISMAGS but not by ISMA. The speedup factors are slightly lower for the Slashdot test instances than for the biological and SNAP networks. This is mostly due to the fact that ISMA already incorporates most of the symmetric information in the subgraphs and thus leaves less room for improvement for ISMAGS. However, ISMAGS still speeds up the searches.

B.3.4 Full comparison

Besides ISMA, the ISMAGS algorithm was compared to 2 similar algorithms, viz VF2 and the GK algorithm. Table B.4 shows the timing results of the VF2 algorithm and the subgraph enumeration algorithm from [7] for some of the subgraphs from the previous section along with the results of ISMAGS. Note that not all test instances from the previous table are repeated as the reference algorithms implementations, in contrast to ISMA and ISMAGS, were not designed to support multiple edges between a pair of nodes.

Table B.4 shows the results for the test instances on the biological networks. Compared to the VF2 algorithm, the GK algorithm reduces query times by exploiting the symmetry in the cliques. As the cliques have more nodes, the symmetry breaking increasingly prunes the search space (e.g. for the 10-clique, only 1 match out of 10! permutations is retained). ISMAGS however further reduces computation times by the optimised matching processes and symmetry breaking described in the previous sections. For most instances, ISMAGS reduces query times by 1-2 orders of magnitude.

The results for the instances on the SNAP networks are shown in Table B.4. Compared to VF2 and GK, ISMAGS again significantly reduces query times by 1-2 orders of magnitude. This allows to process larger networks in reasonable time frames and opens up possibilities for research. Interestingly, the GK results for the tetrahedron instance are worse than the VF2 algorithm. This can be explained by the fact that the GK algorithm focuses on reducing the search space while VF2 focuses on search space traversal. GK reduces the search space by a factor of 3 (by exploiting the symmetry of the tetrahedron) but the VF2 more efficiently processes its (larger) search space. For larger subgraphs, GK becomes better as the symmetry breaking can prune more search space.

Table B.3: Comparison between ISMA and ISMAGS on the Slashdot and SNAP networks. The reported parameters are analogously defined as in Table B.2. Note that for the clique subgraphs, the SNAP networks are assumed to be undirected, while for the tetrahedron, they are assumed to be directed.

		Search space (#nodes)			Calculation time (ms)		
	#instances	ISMA	ISMAGS	SPRF	ISMA	ISMAGS	SF
Slashdot							
3-clique E	12,176	156,167	156,167	1.00	844.37	283.44	2.98
4-clique E	223	168,343	168,343	1.00	1,075.89	364.76	2.95
5-clique E	3	168,566	168,566	1.00	1,253.54	392.47	3.19
3-clique F	392,556	421,011	421,011	1.00	4,145.04	1,600.07	2.59
4-clique F	1,779,701	813,567	813,567	1.00	14,137.33	4,727.03	2.99
FEF	67,594	148,084	139,542	1.06	1,018.42	890.77	1.14
FEE	75,100	178,997	173,435	1.03	1,145.29	618.22	1.85
FE00EF	755,899	7,544,696	1,577,058	4.78	35,769.79	5,388.92	6.64
Wiki-Vote							
3-clique	608,389	107,877	107,877	1.00	1,794.22	410.19	4.37
4-clique	2,077,903	716,266	716,266	1.00	14,137.27	5,156.31	2.74
tetrahedron	84,787	115,205	49,836	2.31	1,025.65	320.72	3.20

Table B.3: Comparison between ISMA and ISMAGS on the Slashdot and SNAP networks (continued).

		Search space (#nodes)			Calculation time (ms)		
	#instances	ISMA	ISMAGS	SPRF	ISMA	ISMAGS	SF
p2p-Gnutella08							
3-clique	2,383	27,078	27,078	1.00	84.96	21.23	4.00
4-clique	175	29,461	29,461	1.00	107.87	28.69	3.76
tetrahedron	2	9,565	6,043	1.58	21.22	4.92	4.32
p2p-Gnutella30							
3-clique	1,590	125,010	125,010	1.00	405.26	113.69	3.56
4-clique	13	126,600	126,600	1.00	464.50	139.06	3.34
tetrahedron	2	41,147	27,365	1.50	101.60	28.03	3.62
CA-CondMat							
3-clique	173,361	116,572	116,572	1.00	548.19	128.96	4.25
4-clique	294,008	289,933	289,933	1.00	1,518.38	357.29	4.25
tetrahedron	0	67,688	39,758	1.70	163.95	35.08	4.67
CA-HepTh							
3-clique	28,339	35,850	35,848	1.00	118.27	30.45	3.88
4-clique	65,592	64,187	64,187	1.00	290.38	64.56	4.50
tetrahedron	0	18,258	11,434	1.60	37.38	9.16	4.08

Table B.4: Comparison between ISMAGS, VF2 and GK on the biological networks. The top row denotes, for each algorithm, the number of runs averaged to obtain the reported timing results. However, for results denoted with an asterisk, only 1 run was performed.

	#instances	Calculation time (ms) VF2	GK	ISMAGS
#runs		1,000	1,000	1,000
PGS-network				
GGG	9,008	885.56	319.27	5.80
SSS	78	24.12	22.90	0.24
SsS	0	22.50	17.40	0.06
P0P	24,452	268.76	103.81	2.87
P0P00P	221,290	4,252.12	585.77	30.67
P0P00P000P	2,570,154	35,303*	5,705.64	302.55
Petersen	9,430	6,854,418*	53,608*	733.25
Reduced PGS-network				
3-clique	10,614	1,170*	511.33	8.81
4-clique	11,150	7,300*	1,177.12	27.60
5-clique	7,669	32,527*	2,160.31	48.36
6-clique	3,616	115,409*	2,968.93	64.90
7-clique	1,158	329,521*	3,513.20	76.63
8-clique	226	754,671*	3,700.21	84.97
9-clique	24	1,337,881*	3,813.57	92.21
10-clique	1	1,848,315*	4,181.80	99.08

Table B.4: Comparison between ISMAGS, VF2 and GK on the SNAP networks. Similar to Table B.4, the top row denotes, for each algorithm, the number of runs averaged to obtain the reported timing results. However, the result denoted with a circle was averaged over 10 runs.

		Calculation time (ms)		
	#instances	VF2	GK	ISMAGS
#runs		100	100	1,000
Wiki-Vote				
3-clique	608,389	187,191.52	27,940.99	410.19
4-clique	2,077,903	3,410,302°	189,357.30	5,156.31
tetrahedron	84,787	15,260.17	106,367.64	320.72
p2p-Gnutella08				
3-clique	2,383	816.04	1,163.03	21.23
4-clique	175	1,659.69	1,359.35	28.69
tetrahedron	2	114.66	1,151.98	4.92
p2p-Gnutella30				
3-clique	1,590	6,259.23	5,681.83	113.69
4-clique	13	5,867.19	5,527.54	139.06
tetrahedron	2	1,991.82	5,793.46	28.03
CA-CondMat				
3-clique	173,361	37,742.10	7,196.78	128.96
4-clique	294,008	232,134.17	15,558.19	357.29
tetrahedron	0	6,547.67	11,779.51	35.08
CA-HepTh				
3-clique	28,339	4,441.81	1,416.95	30.45
4-clique	65,592	21,374.00	2,361.03	64.56
tetrahedron	0	539.45	1,790.64	9.16

B.4 Conclusion

In this paper, we present the Index-Based Subgraph Matching Algorithm with General Symmetries (ISMAGS), an improved version of the Index-Based Subgraph Matching Algorithm (ISMA). The improved version takes into account all symmetric structures in a subgraph. Whereas ISMA minimises the search space exploration by optimising the order in which the nodes of the query subgraph are investigated, it only takes into account the basic symmetries (e.g. single node symmetry and rotation). ISMAGS removes this restriction by introducing symmetry-breaking constraints in the search tree traversal using a customised symmetry analysis. This analysis yields symmetry-breaking constraints which were incorporated in a list-based implementation of the algorithm. Experiments show that the optimised implementation of list operations and the symmetry-breaking constraints significantly reduce calculation times. On average, a speed-up factor (compared to ISMA) of 3 to 4 was present for the subgraphs in the experiments. However, depending on the degree and complexity of symmetry in the subgraph, the speed-up factor varied between 1.14 (for simple symmetric structures already incorporated in ISMA) to 451 (for complex symmetries). Compared to other algorithms available in literature, ISMAGS also reduces query times by 1 to 2 orders of magnitude. While ISMAGS was initially developed to speed up finding subgraph instances in biological networks with multiple edge types, the algorithm can also be used in non-biological networks like social networks to speed up network analysis (e.g. mining for social structures).

Acknowledgements

Maarten Houbraken is supported by a PhD fellowship grant by the Research Foundation - Flanders (FWO-Vlaanderen). The authors would like to thank Dr. Grochow and Prof. Kellis for providing the source code of their algorithm. The authors would further like to acknowledge the support of the Ghent University Multidisciplinary Research Partnership “Bioinformatics: from nucleotides to networks”.

References

[1] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. *Network motifs: simple building blocks of complex networks.* Science (New York, N.Y.), 298(5594):824–827, October 2002.

[2] N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. *Mfinder tool guide*. Technical report, Department of Molecular Cell Biology and Computer Science and Applied Mathematics, Weizman Institute of Science, Israel, 2002.

[3] S. Wernicke and F. Rasche. *FANMOD: a tool for fast network motif detection.* Bioinformatics (Oxford, England), 22(9):1152–3, May 2006. Available from: http://www.ncbi.nlm.nih.gov/pubmed/16455747.

[4] S. Wernicke. *A faster algorithm for detecting network motifs*. In Algorithms in Bioinformatics, pages 165–177. Springer, 2005.

[5] B. D. McKay. *Practical graph isomorphism*. Department of Computer Science, Vanderbilt University, 1981.

[6] P. Ribeiro and F. Silva. *G-tries: an efficient data structure for discovering network motifs*. In Proceedings of the 2010 ACM Symposium on Applied Computing, pages 1559–1566. ACM, 2010.

[7] J. Grochow and M. Kellis. *Network motif discovery using subgraph enumeration and symmetry-breaking*. In Research in Computational Molecular Biology, pages 92–106, 2007.

[8] N. Pržulj. *Biological network comparison using graphlet degree distribution*. Bioinformatics, 23(2):e177–e183, 2007.

[9] S. A. Cook. *The complexity of theorem-proving procedures*. In Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158. ACM, 1971.

[10] A. Lubiw. *Some NP-complete problems similar to graph isomorphism*. SIAM Journal on Computing, 10(1):11–21, 1981.

[11] J. R. Ullmann. *An Algorithm for Subgraph Isomorphism*. Journal of the ACM, 23(1):31–42, January 1976.

[12] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. *Performance evaluation of the VF graph matching algorithm*. In Proceedings of the International Conference on Image Analysis and Processing, pages 1172–1177, 1999.

[13] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. *An improved algorithm for matching large graphs*. In 3rd IAPR-TC15 workshop on graph-based representations in pattern recognition, pages 149–159, 2001.

[14] S. Demeyer, T. Michoel, J. Fostier, P. Audenaert, M. Pickavet, and P. Demeester. *The index-based subgraph matching algorithm (ISMA): fast subgraph enumeration in large networks using optimized search trees.* PloS one, 8(4):e61183, January 2013.

[15] J. Petersen. *Sur le théorème de Tait*. L'Intermédiaire des Mathématiciens, 5:225–227, 1898.

[16] P. T. Darga, M. H. Liffiton, K. a. Sakallah, and I. L. Markov. *Exploiting structure in symmetry detection for CNF*. In Proceedings of the 41st annual conference on Design automation - DAC '04, pages 530–534, 2004.

[17] H. Katebi, K. A. Sakallah, and I. L. Markov. *Graph symmetry detection and canonical labeling: Differences and synergies*. In Turing-100, EPIC vol. 10, pages 181–195, 2012.

[18] J.-F. Puget. *Breaking symmetries in all different problems*. In IJCAI, pages 272–277, 2005.

[19] C. C. Sims. *Computation with Permutation Groups*. In Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation, SYMSAC '71, pages 23–28, 1971.

[20] A. Breitkreutz, H. Choi, J. R. Sharom, L. Boucher, V. Neduva, B. Larsen, Z.-Y. Lin, B.-J. Breitkreutz, C. Stark, G. Liu, J. Ahn, D. Dewar-Darch, T. Reguly, X. Tang, R. Almeida, Z. S. Qin, T. Pawson, A.-C. Gingras, A. I. Nesvizhskii, and M. Tyers. *A global protein kinase and phosphatase interaction network in yeast.* Science (New York, N.Y.), 328(5981):1043–1046, May 2010.

[21] D. Fiedler, H. Braberg, M. Mehta, G. Chechik, G. Cagney, P. Mukherjee, A. C. Silva, M. Shales, S. R. Collins, S. van Wageningen, P. Kemmeren, F. C. P. Holstege, J. S. Weissman, M.-C. Keogh, D. Koller, K. M. Shokat, and N. J. Krogan. *Functional organization of the S. cerevisiae phosphorylation network.* Cell, 136(5):952–963, March 2009.

[22] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. *BioGRID: a general repository for interaction datasets.* Nucleic acids research, 34(Database issue):D535–D539, January 2006.

[23] L. J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering. *STRING 8–a global view on proteins and their functional interactions in 630 organisms.* Nucleic acids research, 37(Database issue):D412–D416, January 2009.

[24] A.-C. Berglund, E. Sjölund, G. Ostlund, and E. L. L. Sonnhammer. *InParanoid 6: eukaryotic ortholog clusters with inparalogs.* Nucleic acids research, 36(Database issue):D263–D266, January 2008.

[25] J. Kunegis, A. Lommatzsch, and C. Bauckhage. *The Slashdot Zoo: Mining a Social Network with Negative Edges*. In Proceedings of the 18th international conference on World wide web, pages 741–750, 2009.