DATA-DRIVEN TRANSIT SIMULATION

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

PIETER JACOBUS FOURIE

Master of Engineering, University of Pretoria,
South Africa

born on 11.09.1974

citizen of
the Republic of South Africa

accepted on the recommendation of

Prof. Dr. Kay Werner Axhausen
Prof. Dr. Kai Nagel
Dr. Alexander Erath

2017
Data-driven transit simulation

Pieter J. Fourie

PhD-Thesis 23948
2017
## Contents

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>3</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Data-driven transport modelling</td>
<td>5</td>
</tr>
<tr>
<td>1.2.1 Application to transit</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Research methodology</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Thesis structure</td>
<td>8</td>
</tr>
<tr>
<td><strong>2 System design</strong></td>
<td>10</td>
</tr>
<tr>
<td>2.1 Overview</td>
<td>10</td>
</tr>
<tr>
<td>2.1.1 Assumptions driving modelling focus</td>
<td>11</td>
</tr>
<tr>
<td>2.2 MATSim</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Data elements</td>
<td>13</td>
</tr>
<tr>
<td>2.3.1 The Contactless EPurse Application System (CEPAS)</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 Road network description</td>
<td>16</td>
</tr>
<tr>
<td>2.3.3 Transit schedule</td>
<td>16</td>
</tr>
<tr>
<td>2.3.4 Vehicle fleet description</td>
<td>17</td>
</tr>
<tr>
<td>2.4 Process elements</td>
<td>18</td>
</tr>
<tr>
<td>2.4.1 Data synthesis and demand generation</td>
<td>18</td>
</tr>
<tr>
<td>2.4.2 Dwell time modelling</td>
<td>19</td>
</tr>
</tbody>
</table>
2.4.3 Network modification .................................................. 19
2.4.4 Trajectory reconstruction ............................................. 21
2.4.5 Speed regression modelling and modified transit simulation . . 22
2.4.6 Distributed re-planning and pseudo-simulation ..................... 22
2.4.7 Post-processing and analysis .......................................... 23

3 Trajectory reconstruction .................................................. 25
3.1 Introduction ....................................................................... 25
3.2 Challenges ....................................................................... 26
3.3 Method ............................................................................. 28
3.4 Applications ..................................................................... 32
   3.4.1 Average waiting times by time of day ......................... 32
   3.4.2 Waiting times related to vehicle trajectories for a given bus service . . 35
3.5 Conclusion ...................................................................... 35

4 Localised prediction of bus speeds from reconstructed trajectories 36
4.1 Overview ......................................................................... 36
4.2 Deriving topological and dynamic variables .......................... 37
   4.2.1 Variable listing .......................................................... 38
4.3 Data preparation ............................................................. 40
   4.3.1 Multivariate outlier detection ...................................... 40
   4.3.2 Centring and scaling .................................................. 41
4.4 Correlation and multicollinearity ......................................... 43
   4.4.1 Trading off Akaike Information Criterion (AIC) vs Variance Inflation Factor (VIF) .......................... 45
4.5 Comparison of instantaneous vs. hourly, speed vs. travel time prediction . . 48
4.6 Interactions .................................................................... 51
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7 Random forest regression</td>
<td>52</td>
</tr>
<tr>
<td>4.7.1 Training</td>
<td>53</td>
</tr>
<tr>
<td>4.7.2 Results</td>
<td>53</td>
</tr>
<tr>
<td>4.8 Modelling stochasticity</td>
<td>55</td>
</tr>
<tr>
<td>4.8.1 OLS regression results</td>
<td>55</td>
</tr>
<tr>
<td>4.8.2 RF regression results</td>
<td>55</td>
</tr>
<tr>
<td>4.8.3 Comparison of simulated results versus actual</td>
<td>57</td>
</tr>
<tr>
<td>4.9 Temporal autocorrelation</td>
<td>60</td>
</tr>
<tr>
<td>4.10 Spatial autocorrelation</td>
<td>61</td>
</tr>
<tr>
<td>4.10.1 Background</td>
<td>61</td>
</tr>
<tr>
<td>4.10.2 Results</td>
<td>67</td>
</tr>
<tr>
<td>4.11 Conclusion</td>
<td>70</td>
</tr>
<tr>
<td>5 Transit simulation</td>
<td>72</td>
</tr>
<tr>
<td>5.1 Key elements of the data-driven transit simulation</td>
<td>72</td>
</tr>
<tr>
<td>5.1.1 Demand generation</td>
<td>72</td>
</tr>
<tr>
<td>5.1.2 Link dynamics</td>
<td>73</td>
</tr>
<tr>
<td>5.1.3 Dwell time model</td>
<td>75</td>
</tr>
<tr>
<td>5.2 Validation and performance</td>
<td>75</td>
</tr>
<tr>
<td>5.2.1 Headways, dwell times and bus bunching</td>
<td>75</td>
</tr>
<tr>
<td>5.2.2 Passenger travel time measures</td>
<td>79</td>
</tr>
<tr>
<td>5.2.3 Computation times of simplified simulation</td>
<td>81</td>
</tr>
<tr>
<td>5.3 Application</td>
<td>82</td>
</tr>
<tr>
<td>5.3.1 Impact on bus bunching</td>
<td>83</td>
</tr>
<tr>
<td>5.3.2 Excess waiting times</td>
<td>83</td>
</tr>
<tr>
<td>5.4 Conclusion and future work</td>
<td>85</td>
</tr>
</tbody>
</table>
6 Improving simulation performance

6.1 Development context

6.1.1 Transit

6.2 Motivation

6.2.1 Harnessing parallelisation

6.2.2 The expanding scope of MATSim applications

6.3 Background

6.3.1 Mutation approaches

6.3.2 Best-response vs. random-response replanning

6.3.3 Simulation-based optimisation using surrogate models

6.3.4 Feedback and learning

6.4 Design

6.4.1 MATSim events

6.4.2 Pseudo-Simulation (PSim) operation

6.5 Experimental setup

6.5.1 Simulation scenario

6.6 Results

6.6.1 Characterizing solution state

6.6.2 Varying QSim:PSim ratio

6.6.3 Performance test

6.6.4 Solution state

6.7 Conclusion

6.7.1 Performance

6.7.2 Solution state

6.7.3 Application to data-driven simulation
# List of Figures

1.1 Schema of design science research cycles. ........................................... 7

2.1 Schema of the system design elements. .............................................. 11

2.2 MATSim default operation. ................................................................. 13

2.3 Boarding/alighting flows and number of on-board passengers .................. 19

2.4 Original navigation network and reduced network. .................................. 20

3.1 Problems in identifying bus dwell operations from smart-card transactions .... 26

3.2 Overview of the trajectory reconstruction process. ................................... 28

3.3 Filtering of GPS errors using the 'speed' between consecutive transactions .... 30

3.4 Interpolation of missing dwell operations ............................................. 31

3.5 Average waiting time distributions by time of day .................................. 33

3.6 Bus vehicle space-time graphs with rolling average wait time per stop ............ 34

4.1 Distribution of speeds of valid data, compared with multivariate outliers .......... 41

4.2 Leverage before and after MVO elimination .......................................... 43

4.3 Correlation plot of all variables generated for the speed regression study ........ 44

4.4 Efficient frontier of VIF and AIC, compared with coefficient ratios ............... 47

4.5 Tree model, recursively partitioning the speed observations ....................... 52

4.6 Random Forest (RF) variable importance ............................................. 54

4.7 Random forest variable importance for standard deviation model ............... 57
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8</td>
<td>Comparison of OLS vs RF speed profiles for shopping district.</td>
<td>58</td>
</tr>
<tr>
<td>4.9</td>
<td>Comparison of OLS vs RF speed profiles for industrial area.</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>Comparison of RF point prediction vs stochastic density vs source data.</td>
<td>60</td>
</tr>
<tr>
<td>4.11</td>
<td>Temporal autocorrelation for a single stop-to-stop combination.</td>
<td>62</td>
</tr>
<tr>
<td>4.12</td>
<td>Temporal autocorrelation at a stop, actual vs. simulated</td>
<td>63</td>
</tr>
<tr>
<td>4.13</td>
<td>Temporal autocorrelation distribution</td>
<td>64</td>
</tr>
<tr>
<td>4.14</td>
<td>Maps of OLS, GWR and RF residuals.</td>
<td>65</td>
</tr>
<tr>
<td>4.15</td>
<td>Maps of difference between GWR and OLS coefficients.</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>Comparison of bus speed between stops</td>
<td>76</td>
</tr>
<tr>
<td>5.2</td>
<td>Comparison of bus headway distributions</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison of bus headway variability</td>
<td>77</td>
</tr>
<tr>
<td>5.4</td>
<td>Bus dwell time histogram and density comparison of non-trivial cases.</td>
<td>78</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of space-time graphs of CEPAS versus simulation</td>
<td>80</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of simulated versus actual trip travel times</td>
<td>81</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of simulated versus actual bus stage-in vehicle time.</td>
<td>81</td>
</tr>
<tr>
<td>5.8</td>
<td>Comparison of simulated versus actual transfer time.</td>
<td>82</td>
</tr>
<tr>
<td>5.9</td>
<td>Space-time diagram of a bus line before and after the line has been split into two.</td>
<td>83</td>
</tr>
<tr>
<td>5.10</td>
<td>Comparison of the excess waiting time before and after a long bus line has been split into two separate routes.</td>
<td>84</td>
</tr>
<tr>
<td>6.1</td>
<td>Number of mutations vs score of executed plan</td>
<td>89</td>
</tr>
<tr>
<td>6.2</td>
<td>Schematic illustration of the modified MATSim simulation process with pseudo-simulation.</td>
<td>93</td>
</tr>
<tr>
<td>6.3</td>
<td>Example MATSim XML event stream for a single agent.</td>
<td>94</td>
</tr>
<tr>
<td>6.4</td>
<td>Average executed score versus QSim iterations</td>
<td>99</td>
</tr>
<tr>
<td>6.5</td>
<td>Score evolution vs time for large-scale scenario</td>
<td>100</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.6</td>
<td>Score evolution vs time for large-scale scenario</td>
<td>101</td>
</tr>
<tr>
<td>6.7</td>
<td>Departure profile RMSD and car mode share comparison</td>
<td>102</td>
</tr>
<tr>
<td>7.1</td>
<td>Master/slave configuration for distributed simulation</td>
<td>109</td>
</tr>
<tr>
<td>7.2</td>
<td>Master/slave task distribution</td>
<td>110</td>
</tr>
<tr>
<td>7.3</td>
<td>Example comparing two Kriging models with the true response of an unknown function</td>
<td>116</td>
</tr>
<tr>
<td>7.4</td>
<td>Experimental design for deriving Kriging model of distributed simulation parameters</td>
<td>118</td>
</tr>
<tr>
<td>7.5</td>
<td>Parameters from from DEO solution and surrounding solution space in Kriging model</td>
<td>120</td>
</tr>
<tr>
<td>8.1</td>
<td>Pairwise histograms of smart card data and its principal components</td>
<td>125</td>
</tr>
<tr>
<td>8.2</td>
<td>Generalisation-based anonymisation: anonymisation step</td>
<td>131</td>
</tr>
<tr>
<td>8.3</td>
<td>Generalisation-based anonymisation: reconstruction step</td>
<td>132</td>
</tr>
<tr>
<td>8.4</td>
<td>Illustration of the process of histogram matching</td>
<td>133</td>
</tr>
<tr>
<td>8.5</td>
<td>Iterative multiple histogram matching</td>
<td>135</td>
</tr>
<tr>
<td>8.6</td>
<td>Comparison of spatial distribution of trip origins</td>
<td>141</td>
</tr>
<tr>
<td>8.7</td>
<td>Comparison of trip Euclidean speed against source data before IHM and after 50 iterations</td>
<td>142</td>
</tr>
<tr>
<td>8.8</td>
<td>Comparison of trip destination counts and travel time in smart card data and synthesis</td>
<td>143</td>
</tr>
<tr>
<td>8.9</td>
<td>Percentile “sinogram” of trip origin coordinates</td>
<td>145</td>
</tr>
</tbody>
</table>
## List of Tables

4.1 Input data summary ................................................. 42
4.2 VIF vs. AIC efficient frontier .................................. 49
4.3 Comparison of instantaneous vs. hourly model ............... 50
4.4 Model of standard deviation of speeds per hour ............... 56
4.5 GWR model summary ............................................. 66
4.6 Comparison of spatial autocorrelation models with GWR, OLS, RF .... 69
6.1 Summary of solution state measures ............................. 103
7.1 Kriging model estimation results ................................. 119
List of Acronyms

AIC  Akaike Information Criterion
API  Application Programmer’s Interface
AWT  Actual Waiting Time
CEPAS  Contactless EPurse Application System
DEO  Differential Evolutionary Optimisation
EWT  Excess Waiting Time
FSM  Four-Step Model
GTFS  Generalised Transit Feed Specification
GPS  Global Positioning System
GWR  Geographically Weighted Regression
ICT  Information and Communication Technology
IHM  Iterative Histogram Matching
ITS  Intelligent Traffic Systems
KNN  k-Nearest Neighbour
LAGSAR  spatial simultaneous autoregressive lag
LBS  Location-Based Services
LTA  Land Transport Authority
MRT  Mass Rapid Transit
MSE  Mean Squared Error
MVO  Multivariate Outlier
**OLS**  Ordinary Least Squares

**PCA**  Principal Component Analysis

**PSim**  Pseudo-Simulation

**QSim**  Queue Simulation

**RAM**  Random Access Memory

**RF**  Random Forest

**RLD**  bus Registration number, Line and Direction

**RSS**  Residual Sum of Squares

**SACSAR**  spatial simultaneous autoregressive lag and error

**SMP**  Symmetric Multiprocessor System

**SWT**  Scheduled Waiting Time

**VIF**  Variance Inflation Factor
Data-driven transit simulation

Pieter J. Fourie
Future Cities Laboratory
Singapore ETH Centre
1 CREATE Way, 138602
Singapore
phone: +65-9116-8440
fax:
fourie@ivt.baug.ethz.ch

2017

Abstract

This thesis details the design of a modified MATSim agent-based simulation system that is driven by transit smart card data. A complete set of bus vehicle trajectories are reconstructed from the smart card transactions. The reconstructed trajectories are then used to estimate a number of regression models of the speed at which buses travel between stops. These models drive the simulation dynamics, along with a dwell time model which takes account of bus ridership, in a MATSim simulation where smart card transaction records have been transformed into an agent-based travel demand.

Besides an exposition of the various elements of the data-driven system, as well as validation of the simulation output and a test case scenario, the thesis also covers a number of related topics, including a technique to improve simulation times for MATSim simulations in general, a method to distribute MATSim simulations across multiple computer nodes, a useful approach to perform parameter optimisation in MATSim using a Kriging meta-model, and ends with a technique to synthesise surrogate data that could be used to drive the simulation, while eliminating any associated privacy concerns.

Keywords

Transit, public transport, trajectory reconstruction, speed regression modelling, transit smart cards, data-driven modelling, surrogate data synthesis, simulation performance, agent-based simulation, MATSim, Singapore

Preferred citation style

Zusammenfassung

Diese Arbeit beschreibt die Gestaltung eines modifizierten agenten-basierten Simulationssystems MATSim, welches die im öffentlichen Verkehr (ÖV) durch Smart-Card Transaktionen gesammelten Daten verwendet.


Schlüsselwörter

Agenten-basierten Simulation, öffentlichen Verkehr, MATSim, Surrogatdaten, Geschwindigkeitsmodellierung, Singapur

Bevorzugter Zitierstil

Acknowledgments

During my time at the Future Cities Laboratory I have crossed paths with some truly exceptional people, many of whom still form an important part of my life. I would like to thank a few of them here and beg the forgiveness of those who I might have left out.

First and foremost, Prof. Dr. Kay Axhausen, who has not only guided my studies over the past six years, but also serves as an incomparable role model and inspiration. I appreciate the freedom he afforded me in my explorations, while always having been available for guidance and to reel me back in when I spun out on a wild tangent. The young and energetic Dr. Alex Erath has led the efforts of our team here in Singapore and created opportunities with various authorities. Without his tireless efforts I would not have had much to research or write about in the first place. Prof Kai Nagel from VSP, TU Berlin, who was the original inspiration for me to have embarked on my adventures in agent-based simulation back in 2008. The open source software platform, MATSim, that is his brainchild, has enabled so many fascinating new avenues of research across the world. It is deeply satisfying to continue to be part of the MATSim journey.

Singapore has been a welcoming host country, it’s people kind, and I am grateful for the opportunities it has afforded me. My studies and employment here was funded by the National Research Fund. This work would not be possible if not for the continuing collaboration with the Land Transport Authority, who provided access to their transit smart card data for the development of the simulation system.

For Chapter 6, I would like to thank the developers of the Zurich scenario for making the data available; they include Michael Balmer and Marcel Rieser from Senozon A.G., Zurich, and the team of Kay Axhausen at the Institute of Transport Planning and Systems, ETH Zurich: Francesco Ciari, Andreas Horni, Konrad Meister, Basil Vitins and Rashid Waraich. Gunnar Flötteröd from the Royal Institute of Technology, Stockholm, for insightful feedback, and Dominik Grether from VSP, TU Berlin for his technical assistance during development of the PSim module.

I would also like to thank Kirill Müller from IVT, ETH Zurich, for his assistance and insight in the histogram matching work of Chapter 8 as well as having been a collaborator on many
fronts across the years.

On a personal note, I would like to single out a few of my mobility research colleagues at FCL who have sustained me over the duration of this effort. My dear friend Artem who I have shared many adventures with across Asia and Europe. Sergio, who ensures that never a dull day passes at the office, and who is always ready to explore a new idea. Michael, thoughtful and never rushing to judgement, for all the enlightening chats over coffee and advice on fatherhood.

It gives me great pleasure to dedicate the work of this thesis to my family; my parents Piet and Cora, who have made tremendous sacrifices for me and my siblings over the years and have always believed in me even when I did not. My brother Johan for his good heart and taking care of our parents in my absence. My sister Anélia who never gives up and brother-in-law Rafal who is always there for her.

To my wife Mihiro, for her sense of humour, pushing me to work on the thesis when I got lazy, and entering into a whole new adventure together. And to the apple of my eye, little Linka, who continues to bring us more joy (and less sleep) every day.
Chapter 1

Introduction

This thesis presents a modified activity/agent-based simulation system, based on MATSim \cite{Horni:2016} where transport demand, supply and simulation dynamics are driven by transit smart card data. It has been implemented for Singapore smart card data, where the technology has a very high penetration rate and records both trip stage origin and destination at high spatial and temporal resolution. Smart card data is remarkably congruent with a disaggregate model such as MATSim, where individual agents making trips are the mode of analysis. The smart card records can be easily transformed into a very accurate agent-based travel demand description. If the records are properly geo-located, one need only group trip stages together and identify the initial board and final alighting locations of transit users, in order to produce a MATSim agent-based plans file.

However, using the data to inform the supply and simulation dynamics of the simulation requires more specialised processing. The process starts with the reconstruction of bus vehicle trajectories using the smart card transactions, published transit schedules, a road network and spatial coordinates of the bus stops\footnote{In this study, it was assumed that trains largely operate according to schedule, thus not requiring trajectory reconstruction. The point is argued in more detail in the following chapter.}. The reconstructed trajectories are then used to estimate regression models of the speed, and its variability, at which buses travel between stops during the course of the day. The regression models are used to drive the simulation dynamics, obviating the need for detailed network topology as well as the simulation of other modes of transportation in the system, as bus speeds between stops predicted by the regression models already incorporate their effects. The data was used in another study \cite{Sun:2014} to produce a dwell time model which takes account of bus ridership. Transport supply is modified from the published Generalised Transit Feed Specification (GTFS) schedule using the reconstructed bus trajectories, in order to have realistic departure frequencies.

Besides an exposition of the various elements of the data-driven system, as well as validation
of the simulation output and a test case scenario, the thesis also covers a number of related topics, including a technique to improve simulation times for MATSim simulations in general, a method to distribute MATSim simulations across multiple computer nodes, a potentially useful approach to perform parameter optimisation in MATSim using a Kriging meta-model, and ends with a technique to synthesise surrogate data that could be used to drive the simulation, while eliminating any associated privacy concerns.

This chapter attempts to motivate the research, and define what is meant by 'data-driven' transport modelling.

### 1.1 Motivation

Activity-based travel demand modelling is gaining acceptance in the transport planning domain, but comes with a new set of challenges compared to traditional aggregate models (e.g. Vovsha et al. 2005; Davidson et al. 2007). The disaggregate approach has been criticised for its complexity and data-intensiveness. Furthermore, unlike the aggregate Four-Step Model (FSM) — which relies on relating directly measurable quantities, such as zonal building capacities and traffic counts, to trip flows and travel times — activity/agent-based methods rely on these larger-scale phenomena to emerge from individual behaviour and interaction from the bottom up. It therefore provides no direct handle to adjust the larger-scale observations, as is typically done during FSM calibration.

But perhaps the biggest challenge in disaggregate demand modelling has been the reliance on a few observations of (usually) reported human behaviour, recorded in travel diary surveys, to be generalised to the population at large. Whatever uncertainties or errors that might be encountered along the way in this process of generalisation, and the degree to which these inaccuracies interact and compound, affect the reliability of model predictions. Surveys are very expensive to conduct, take long to encode and release, and are only performed periodically, further compounding problems of generalisation with the added dimension of time.

However, the advent of big data, from sources like transit smart cards and cellular phones, have the potential to revolutionise disaggregate travel demand modelling by overcoming many of these problems. These data sources are congruent with the granularity and mode of analysis of the disaggregate modelling approach, providing direct observation of individual travel behaviour between activity locations, updated on a daily basis. An activity/agent-based simulation model, such as MATSim (Horni et al. 2016), can be fed with these observations, and if the transport supply is accurately represented, general transport mode choice parameters are correctly specified, and simulation dynamics are realistic, the model should produce a reasonable likeness of the trip flows and travel times observed in reality. This thesis describes a first effort
to develop such a data-driven agent-based simulation, for the specific case of transit smart card data.

It should be noted from the outset that no effort is made to identify or understand the underlying activity patterns in the data, or the individual attributes and contexts generating these patterns. Consequently, the planning horizon for the approach discussed here should probably be limited to the short term and a limited scope of expected behavioural responses to policy interventions. Many new strands of research are focused on learning from and fusing data sources to impute individual attributes and classify activity purposes, with the ultimate goal of turning big data sources into an ongoing travel survey. However, there are already many advantages to be gained from getting on with modelling with big data before we have reached this level of maturity, as this first attempt hopes to illustrate.

A brief discussion is warranted to define what is meant by a data-driven model, as the term appears to be redundant; aren’t all models informed in some way by data?

### 1.2 Data-driven transport modelling

The predictive accuracy of a transport model generally depends on (a) an accurate representation of transport supply and policy; (b) a realistic description of demand; and (c) how well the model captures the dynamics and interactions of the demand with the transport supply, governed by policy. For a model to be said to be truly data-driven, this thesis argues that all of these components need to be grounded in case-specific, contemporaneous observations, preferably of the entire population. It is worth examining these elements in turn, to briefly contrast how they are addressed in existing frameworks versus the data-driven approach proposed in the thesis.

**Transport supply and policy** are usually inventoried, but it is possible that, in a multi-stakeholder environment, complete knowledge of infrastructures, services and policies are not available to the transport modeller. Some supply information, such as published schedules, might also not always be realised due to operational contingencies. In a data-driven approach, implementation of published supply information and official policy would be modified by observation of actual events. For example, in this thesis, transit smart card data is processed to determine the actual departures of buses during the course of the day, and published GTFS schedules are modified to reflect these changes.

**Transport demand** is, arguably, the most challenging component to compile in an activity-based model, as it requires one to extract essential characteristics from observations of reported behaviour, e.g. time use or travel diary surveys, and then generalise these to the rest of the population. A data-driven approach would, instead, be derived from direct observation of the full population’s actual behaviour, such as transit smart card or cellular phone data.
**Model dynamics** can range from simple, monotonically increasing volume-delay functions that attempt to capture how link travel times increase with increasing volume, to finely-grained, agent-based interactions between commuters, vehicles and transportation infrastructure. In a standard modelling system, model dynamics would be determined by some approximation of the physics of the system, informed by known attributes of components, such as typical capacities of different road grades and vehicle configurations. In the data-driven approach, the physics engine would either be replaced or modified by finely-grained observations; the degree of generalisation would therefore be dramatically reduced.

For instance, in the system described in the thesis, the speed at which buses travel between bus stops does not emerge from interactions with other traffic, as it does in a standard MATSim simulation. Instead, it is derived from the processed smart card data, as the influence of surrounding traffic by time of day is captured in the implicit timings of vehicle movement. There is generalisation to unobserved road segments through a regression model that takes account of a number of topographical and dynamic variables, providing much better estimates of the actual behaviour of the system than what the broad generalisations driving a standard simulation would produce.

In a recent literature survey (Anda et al., 2017), a number of developments in the area of data-driven transport modelling are discussed, further highlighting the differences between the new approach and that followed in traditional models.

1.2.1 Application to transit

Pelletier et al. (2011) provide an excellent review of how transit smart card data is employed in planning at strategic, tactical and operational levels. Up until the time of their review, transit smart card data use in predictive modelling had been limited to future trend estimation to create origin-destination matrices. None of the authors they reviewed had been using the data to drive disaggregate models. In fact, the authors had identified new modelling approaches (specifically the ‘Totally Disaggregate Approach’) as a major avenue for new research in their conclusions.

Since then, a special interest group of transport researchers engaged in working with transit smart card data, convened in Gifu, Japan, in April 2014, at the First International Workshop on Utilising Transit Smart Card Data for Service Planning. The first iteration of the system explained in this thesis had been presented at this occasion, and subsequently made its way into a book on the topic (Fourie et al., 2016). From several searches of the literature, it appears to be the first disaggregate simulation model of its kind, where all components of the model, as set out above, have been informed by the same data source. While there have been other studies using MATSim as a simulation platform for a travel demand derived from smart card data (e.g. Ali et al., 2016), modifying the simulation dynamics and published transit schedule
Figure 1.1: Schema of design science research cycles.

![Design Science Research Cycles Diagram](image)

Source: Hevner (2007)

are critically important elements of the system presented in this thesis. These modifications increase the accuracy of results while reducing simulation time and complexity, as there is no need to simulate private transport in order to produce realistic bus travel times.

1.3 Research methodology

Leonardo Da Vinci is attributed with saying ‘Art is never finished, only abandoned’. The same can be said of the work presented in this thesis. It was written up following the development and testing of several iterations of the system design, only to find that, while the simulation manages to produce reasonable results, there is still substantial room for improvement.

While the result is therefore not as elegant and universal as \( E = mc^2 \), a considerable amount of firm knowledge was still produced. Studies of this kind can be said to follow a design research methodology, as suggested by, e.g., Hevner (2007). In a schematic representation of the method, in Fig. 1.1 it can be seen that the three suggested cycles of relevance, design and rigour can lead to worthwhile expansion of the knowledge base, with the implicit acknowledgement that the process never finishes, and that future cycles might be warranted.

While not strictly adhering to this method in particular, it can be said, in retrospect, that many of the elements identified in Fig. 1.1 have formed a part of the development process. Problems and opportunities have been identified in the domain of activity/agent-based transport demand planning, in the light of the emergence of a new technical system (transit smart card systems).
From knowledge of the organisational requirements of transit system operators and authorities, the design science research method of building and evaluating artefacts and processes (e.g. regression models, modified simulation subsystems) were presented to the transport research community in reports, book chapters and conference presentations. From this process of peer review (the rigour cycle), the research was deemed 'good enough' to qualify as a worthwhile addition to the knowledge base.

It is therefore within this context that the thesis is presented, with the expectation of the reader’s implicit understanding that, whenever shortcomings in the system are identified, they are intended to be addressed in future cycles of the design science research methodology.

1.4 Thesis structure

System design. The following chapter will cover the system design of the data-driven transit simulation, highlighting the author’s own contributions in what has been a collaborative effort over a number of years, and ends with a description of the input data and the Singapore transit smart card system.

Trajectory reconstruction. In Chapter 3, the first essential element to the data-driven transit simulation is discussed. Bus trajectories are reconstructed from transit smart card transaction records.

Localised bus speed prediction. Chapter 4 explains how the output of the trajectory reconstruction process, as well as transport network topographical data and the original smart card records, are used to model the speed of buses travelling between transit stops.

Transit simulation. Chapter 5 covers modifications made to the MATSim simulation, the demand generation process, and presents validation and performance results, as well as an application of the system in a simple fictitious case study.

Improving simulation performance. While the simulation system, as presented, can be run within a few hours on appropriate hardware, it is easy to imagine how users will run into performance limitations as bigger scenarios are explored with more possible behavioural responses to policy interventions. To this end, Chapter 6 proposes a method for speeding up MATSim simulations in general, while Chapter 7 extends the method to make use of distributed computer architectures and cloud computing services.

Data privacy through surrogate data synthesis. While the results of the data-driven transit simulation described in this thesis uses actual transit smart card records as its input, the author recognises that practical application would require guarantees of maintaining data privacy.
The system design chapter to follow shows how the supply-side components of the simulation require no exposure of individual observations, only deriving operational parameters from the data. The bigger challenge is to produce a disaggregate demand that does the same. The final chapter represents a first attempt to do just that, in order for the entire model to be employed in practice as a *machine-eyes only* approach, with absolute guarantees of individual privacy.
Chapter 2

System design

Many concepts presented in this thesis have been developed and refined over almost four years as a collaborative effort by the transportation and mobility modelling team at the Future Cities Laboratory in Singapore, under the leadership of Dr Alex Erath, who also came up with the original idea of a smart card data-driven transit simulation system. The team published a book chapter in a recent volume (Fourie et al., 2016) summarising the system design, operations and results.

As the development of this system has been a collaborative project, this chapter aims to not only give an overview of the various design elements presented in Fig. 2.1, but also point out the author’s contributions, which will form the subject matter of the rest of the thesis.

2.1 Overview

Figure 2.1 provides an overview of the system design of the data-driven transit simulation. It largely relies on one set of input data, the transit smart card data, and uses it to drive most components of the simulation.

Transport demand is assumed to be completely described by the transit smart card records, assuming the demand to be the initial boarding and final alighting locations of passengers. Instead of relying on published transit schedules, the actual schedule of departures is taken from the data, by reconstructing the trajectories of buses. Simulation dynamics, such as the dwell time at stops, and the time that it takes a bus to travel between stops, is determined by analysis of the data. Regression models derived from the dwell time and speed observations makes the simulation capable of predicting how the transit system will respond if its configuration is altered. No attempt is made to simulate private vehicles, as it is assumed that speed predictions from the regression model capture the effects of interactions between transit and private vehicles. The
only other inputs to the model are a network description, and a transit route description, that is
geo-coded for transit stops to be associated with links in the network.

2.1.1 Assumptions driving modelling focus

Trains run to schedule. The core assumption that is implicit in all of the modelling steps
presented in this chapter, is that trains operate largely according to schedule, while buses are
subject to far more sources of variation. Therefore, no attempt is made to reconstruct train
trajectories in order to derive the actual departure frequencies of different lines; instead, they
run according to the published schedule. Neither was a regression model formulated to model
train speeds between stops as a function of system load and network topology.

The motivation is straightforward:

- Trains do not have any interaction with other modes of transportation and the congestion
  that they cause;
- Trains have many more exits per vehicle for fast boarding and alighting, therefore train
  bunching due to long dwell times is more the exception than the rule;
- It is possible to exercise a far higher level of control over the operation of the train system
than what would be possible for buses, with dedicated and intelligent signalling systems that constantly adapt to operational circumstances. Therefore, the system can recover more easily from disruptions and return to scheduled operations;

- The number of points of interaction between train services on different lines, i.e. transfer stations, are far fewer than those in the bus network.

These assumptions were borne out by the results ultimately produced by the simulation, in terms of the travel time distributions produced across all modes at the level of individual agents.

**Captive demand.** Another implicit assumption is that changes in configuration or service frequency can only affect the relative mode shares of train, bus and walking, and that the entire demand is captive to these modes. The current design does not allow for passengers to switch away from transit modes, or to move to transit from other modes.

This assumption holds well for Singapore, where car ownership is highly regulated, and the share of public transport is high (more than 66% of trips during peak hours are made by transit). The topic will have to be revisited for scenarios where the commuter population have more options available to them.

**No induced demand.** Finally, the observed trip-making of the commuter population is not expected to change in response to any changes in service frequency, network configuration or pricing. Neither are changes external to the model system able to affect demand, e.g. changes in car ownership, income, or increased non-motorised accessibility. These assumptions are contrary to literature on factors influencing transit demand, e.g. [Paulley et al. (2006)].

In future work, the capability to predict induced or lost demand could be addressed by additional models. These could range from a simple elasticity-based approach to complete classification of smart card records against a travel survey using machine learning, in order to synthesise additional demand or trim it back.

## 2.2 MATSim

The data-driven transit simulation system is built on the MATSim framework. MATSim simulates the traffic produced in a transportation network by agents pursuing daily schedules of activities (plans) separated in time and space. Its principle of operation is shown in Fig. 2.2.

The system is fed with an initial demand of agent plans that are repeatedly executed in a Queue Simulation (QSim) network loading. After each QSim run, plan performance is evaluated using a utility-based scoring function. Then, agent plans are mutated along a number of choice
dimensions, such as activity start times and durations, route choice, trip transport mode, activity location choice, etc., to produce new plans for execution in the following QSim iteration. With increasing iterations, the number of plans in each agent’s memory grows up to a limiting number, following which poorly performing plans are discarded. Consequently, the average score of plans improves with increasing iterations, until a ‘relaxed’ state is reached where plan mutations do not further improve the score of any agent’s day plan.

This approach is analogous to that of evolution by natural selection, where a genotype, or plan, is expressed as a phenotype in the physical environment, i.e. an agent in traffic (e.g. Goldberg 1989; Hraber et al. 1994). The success of the phenotype determines the longevity of genes in the genotype; for MATSim plans, genes can be understood to refer to combinations of plan elements, such as mode choice, activity timing and location, that become more-or-less stable features across generations.

MATSim is well documented, and the interested reader is referred to Horni et al. (2016).

2.3 Data elements

As this is a data-driven simulation, data forms an integral input to the design, and will determine the system’s range of capabilities.

This section provides an overview of the Singapore transit smart card payment system, which forms the basis for both the demand modelling process, as well as several derived models that will determine simulation dynamics. Supporting data includes a network description that is derived from a Global Positioning System (GPS) navigation network, and transit schedule/route profile, giving the locations of bus and train stops, and the order in which these are visited for each transit route. Finally, the simulation is supplied with information on the vehicle fleet serving the transit routes, as these determine both the system’s capacity and the dynamics of passenger boarding and alighting operations at stops.

1Strictly speaking, alleles, not genes, persist over generations. But an explanation of the relationship between the two concepts detracts from the essential idea of the analogy presented here.
Some of the data sources required some preliminary processing and preparation in order for them to be in a suitable format for a standard MATSim run. A lot of the preparation was done long before the system described in this thesis was designed, as part of a project to establish the first large-scale agent-based simulation of all motorised passenger transportation in Singapore. Details of this initial implementation and the preparatory steps can be found in Erath et al. (2012).

2.3.1 The Contactless EPurse Application System (CEPAS)

The CEPAS system operates with a contactless smart card that keeps track of a user’s pre-loaded cash balance. When a user taps the card on a system sensor, a transit transaction is recorded, relevant information is calculated and the user’s balance adjusted.

The system operates differently depending on transportation mode (rail or bus). If they need to use the mass rapid transit (MRT) train service, users tap in and out at station entrances. The exact service and vehicle they use, as well as their boarding/alighting time are therefore not directly recorded.

In the case of bus services, users have to tap in and out when they enter and leave the bus. For each transaction, the system records an array of information, including the vehicle’s registration number, the user’s card identifier, the service and direction of the bus, the boarding or alighting time, as well as the vehicle stop locations as recorded by its GPS tracker. Post-processing by the data provider associates transactions with a bus stop identification number based on the reported GPS location.

Furthermore, concessions are provided to students and senior citizens, so the system will also record the type of card used during the transaction, identifying the user as a child/student, senior citizen or adult user. There are also special concessions for persons with disabilities, as well as for people who require financial support from the government.

2.3.1.1 Pricing and data completeness

The Singapore transit system uses distance-based pricing, providing an incentive for users to tap out in order to avoid excess charges. Consequently, most boarding transactions have a corresponding alighting transaction for the same card identification number, and transactions

---

Sun et al. (2012) have shown how it is possible to determine whether passengers in the MRT system are in trains or waiting at stations, using a regression model based on the observations from the fastest passengers for each origin-destination pair. The model provides the spatiotemporal density of passengers, allowing for the identifications of trains’ trajectories and the assignment of passengers to single trains, according to their estimated location.
can be grouped into trip stages for each card.

For a typical weekday in 2011, fewer than 150,000 of the more than 5 million boarding transactions lack a corresponding alighting transaction. Transactions at train stations are generally complete, as the electronically operated gates physically bar the user from exiting the station if they do not tap out.

Missing alighting transactions are therefore mainly associated with bus trips. A simple analysis of the data shows that the majority of users that do not tap out of buses are students or scholars; approximately 11% of these users do not have a corresponding alighting transaction for each boarding transaction. The reason for this behaviour is probably due to the distance-based fee for students and senior citizens turning into a flat fee for all trips longer than 7.2 km, providing no further incentive for tapping out.

The distance-based charging system works on a full-trip basis, and users are therefore not charged the minimum amount for starting a trip stage when they make a transfer; instead charging continues if the next trip stage starts within a 45 minute allowance, and the entire trip is completed within two hours. It is therefore relatively straightforward to reconstruct the entire trip trajectory of a particular card user from initial boarding to final alighting location. In other cities, where the rule of tapping out of a bus is not imposed, alighting locations have to be imputed, by analysing multiple consecutive days’ worth of data (e.g. Munizaga and Palma 2012).

Electronic payments by CEPAS card account for a total of 96% of all transit transactions, as cash payments are discouraged through higher fares (Prakash, 2008). Smart cards are durable and can be used for years, offering transport analysts a longitudinal view of user behaviour in response to external effects, as well as to the introduction of new transportation infrastructure and policies. A comparison of a week’s worth of data in 2011 and 2013 shows that approximately 42% of all card IDs in use in 2011 were still being used in 2013.

For incidental travellers, a temporary travel card can be purchased that is valid for the MRT and LRT system. There is also “tourist pass” card that allows tourists to travel anywhere in the transit system at S$8 per day (LTA and STB, 2016).

The combination of card durability, distance-based charging and the incentive for using transit smart cards have resulted in Singapore having one of the most detailed and accurate descriptions of transit demand possible today.

Furthermore, the fact that data is recorded at per second accuracy, makes it possible to perform highly detailed temporal analyses, such as determining the time it takes to travel between bus stops, as well as the time that it takes for people to board and alight from a bus given its occupancy. The same kind of information can be determined for trains using the methodology
The current implementation of the simulation system described in this thesis is driven by data taken from a Wednesday in April 2011, which can be assumed to be a ‘typical’ day in Singapore. This means that the data contains both commuters travelling to work and back, as well as most scholars, students and visitors.

### 2.3.2 Road network description

The original road network was supplied in the form of an ESRI line shape file, and is assumed to be an accurate description of the detailed topography of road centre lines in Singapore in 2008. This network was developed by the company, NavTeq, for the purposes of GPS navigation. As the simulation is driven by smart card data from 2011, the assumption is that the road network plied by the buses has not changed significantly between 2008 and 2011.

Each line shape in the file contains information on the road type, the number of lanes and the allowed speed of each section of road. The GPS network was then transformed into a MATSim directed network graph of capacitated links. Our data provider, the Singapore Land Transport Authority (LTA), further provided us with a description of where bus lanes may be found in Singapore. These were encoded in the network description, by splitting the appropriate links into a car link, and a parallel bus-only link. After all this processing, the resulting network contains 82,742 links connecting 44,529 nodes. Sergio Ordóñez was responsible for the coding of the network and bus lanes, used in the original Singapore MATSim implementation.

While the network was prepared in the MATSim format, it is only used during the preparatory stages to running the simulation system described in this thesis. The simulation itself runs on a highly simplified network that is derived from the original network, the construction of which will be described in Section 2.4.3. However, it remains important that all information on the original network be as accurate as possible, if the system is to be used to predict the performance of any changes to the bus network. This is because the regression model of bus speed between stops strongly relies on network topology as one of its predictors. This requirement might preclude the use of networks traditionally used in transportation planning, as these tend to be highly simplified, and exclude roads below a certain grade.

### 2.3.3 Transit schedule

In MATSim, the transit schedule is a compendium of information containing the location of transit stops, and all the transit lines that make up the transit service. A transit line might contain one or more transit routes, which, in turn contain
• the order in which transit stops are visited;
• the sequence of links to traverse between stops;
• the frequency of departures during the course of the day;
• supplementary information such as the vehicle type that will operate on these routes;
• and whether the transit stop blocks other traffic on the link (for the case of transit modes that interact with other modes, e.g. bus and tram).

For the case of Singapore, much of this information is available in the form of a Generalised Transit Feed Specification (GTFS). However, as the creators of the transit feed do not have any relation to the company that provided the navigational network described in Section 2.3.2, the task of specifying the sequence of links between transit stops remained. The GTFS for Singapore happened to contain supplementary information, in the form of GPS traces. Ordoñez Medina and Erath (2011) developed a map matching procedure to come up with the sequence of links in the network between all bus stops for the more than 300 bus lines in Singapore, using the supplementary GPS coordinates to decide on the most likely links to make up the bus’s trajectory.

2.3.4 Vehicle fleet description

For the Mass Rapid Transit (MRT) system, dedicated Wikipedia pages are available describing the capacities of the rolling stock. For buses, a dedicated webpage happen upon that provides extensive details on the stock of buses that operate in Singapore, for every bus route in the transit system.

Pertinent content includes, amongst others, manufacturer, model, length, year of use, entrance type, exact capacity, including both seated and standing passengers, engine model, and emission standard.

This information proved useful in the estimation of regression models for the time spent at bus stops allowing passengers to board and alight from the vehicle, as described in Section 2.4.2.

Sergio Ordoñez created a procedure to automate the encoding of the various vehicle types operating on different routes.

2.4 Process elements

2.4.1 Data synthesis and demand generation

Data privacy is a growing concern in all disciplines dealing with big data sources, but the concern for mobility data is even more severe, as knowledge of people’s movement can directly compromise their personal security. De Montjoye et al. (2013) have shown that you only need four distinct space-time stamps in a day to identify a person uniquely in a given dataset.

From the outset, it should be made clear that the prototype work presented in this thesis uses actual smart card records for travel demand. Co-author of Fourie et al. (2016), Artem Chakirov, created a procedure that transform the transit smart card data into an agent-based demand description suitable for execution in MATSim. This procedure analyses the transit smart card records on a trip basis, determining the initial boarding and final alighting locations. It creates an agent for each trip departing from the initial boarding stop, destined for the final alighting stop. It then determines the arrival times of the buses at the stop, and randomly assigns the departure time for each trip from the time of the last departure at that stop until the current record’s tap-on time.

However, following several presentations of this work to authorities, it has become clear that using data directly is a violation of privacy statutes for many cities that could potentially implement similar systems, and therefore the issue needs to be addressed. In the thesis’s final chapter, a surrogate data synthesis process is proposed that aims to reproduce the full joint distribution of the original dataset from aggregates. This development makes it possible to produce an endless stream of fake transit smart card trajectories that reproduce the same trip flows as the original data.

As all other processes driving the simulation are to estimate models or larger-scale behaviours, such as bus arrivals and departures, they do not produce output containing information on any given passenger. With the data synthesis process it would be possible to generate a set of scripts that pass the data through “machine eyes only”. The only output generated by the scripts to drive the simulation would be:

- a machine that can generate fake, but realistic, transit smart card transactions;
- a realistic transit schedule;
- predicted bus speeds between all transit stops during the course of the simulated day; and
- parameters for the dwell time model.
Figure 2.3: Boarding/alighting flows and number of on-board passengers during dwell operation.

Source: Sun et al. (2014). The plot shows that a bus’s occupancy needs to be below a critical level for boarding to proceed.

2.4.2 Dwell time modelling

A significant cause of bus bunching is the time spent at bus stops waiting for passengers to board and alight from vehicles. In Singapore, users are only allowed to board from the front door of the bus, and alight from the rear. When alighting, they have to tap out at one of the two sensors located at the doors. The transit smart card data contains information on the dynamics of the dwell process that can be derived by analysing the timing of boarding and alighting transactions.

Sun et al. (2014) used this information to derive a model that describes how long it would take passengers to board and alight from the different bus types employed in the Singapore transit network. A typical result from their analysis is shown in Fig. 2.3, where it can be seen that the bus needs to reach a critical occupancy before it is possible for passengers to board.

In Fourie et al. (2016), Sergio Ordóñez developed a Java class that implements the dwell time model with parameters appropriate to the bus vehicle type used for each bus line.

2.4.3 Network modification

In MATSim, simulation time is both dependent on network topology, i.e. the configuration of links and nodes, as well as the number of agents in the system.

Experience has shown that MATSim can only produce realistic results for transit simulations
when the sample size is greater or equal to 25% of the full population. This places a lower limit on the number of agents that need to be simulated. Furthermore, the number of departures for each transit service has to remain fixed, regardless of sample size. The number of transit vehicles that need to be simulated, therefore, also represents a fixed constraint determining performance.
Consequently, the only viable intervention to reduce simulation time would be to reduce the number of nodes and links in the network. We realised that we can eliminate the links between two consecutive stops and replace these with a single link with modified link dynamics in the simulation. The only interaction between buses would be in a short queueing link before the stop, where buses have to wait for earlier arrivals to finish passenger boarding/alighting procedures.

Sergio Ordóñez developed a Java class to produce the simplified network. The program reads in the existing network and the transit schedule, and then traverses the links between consecutive pairs of stops in the schedule to determine the distance between them. A new link of this length is created in the alternative network, as well as a queueing link before the stop.

The procedure dramatically reduces the number of links on the network, from 82,742 down to 11,389 links, while the number of nodes are reduced from 44,529 to only 9,640 nodes. The original GPS navigation network compared with the reduced network is shown in Fig. 2.4.

### 2.4.4 Trajectory reconstruction

As the work of Sun et al. (2014) has clearly shown, the transit smart card data contains a lot more information implicitly than only the boarding and alighting transactions of passengers. While their work focuses on the very short intervals between consecutive transactions, there are also the larger intervals, between transactions at consecutive stops serviced by the same vehicle, that can be used to reconstruct the trajectory of the vehicle, if the stop locations and the path between the stops are known. As all of this information is already encoded in the electronic transit schedule, described in Section 2.3.3, trajectory reconstruction should prove to be a straightforward task.

Chapter 3 details the refutation of this assumption of simplicity, as well as the processes that were established to deal with the various complexities that emerged, in order to arrive at a set of trajectories for nearly every bus service recorded in the smart card data.

It should be noted that the same was not attempted for train services, as associated transactions firstly take place not when entering the vehicle, but instead when transit users enter or leave train stations. The route that train users take through the transit network, as well as the vehicles that they board and alight from, cannot be directly observed in the data, and require special inference processes in order to arrive at approximate train trajectories (see Sun et al., 2012).
2.4.5 Speed regression modelling and modified transit simulation

In reality, the variability in the time taken to travel between stops is due to interaction of buses with other vehicle traffic. Simulating private vehicles is expensive, therefore in order to speed up the simulation, the bus is assigned a travel time between the stops, sampled from an appropriate distribution.

In the initial version of the system, all buses had their speeds sampled from a normal distribution with parameters derived from the full-scale MATSim implementation. Following the development of the trajectory reconstruction process, we realised that it would be possible to instead sample from actual bus speeds. One could therefore take the distribution of bus speeds from the trajectory reconstruction process for all the combinations of stops throughout the course of the day, and sample from these values for each stop-to-stop traversal.

However, while the current performance of the network could thus be modelled quite accurately, one would not be able to predict bus speeds if any changes were made to the network. Furthermore, the observed speeds in the trajectory reconstruction process are probably influenced by the number of services operating between two stops, as well as the frequency of the departures between those stops. Therefore, there would be no guarantee that one would still be able to predict with accuracy if the frequency of services were changed instead of the transit network itself.

Consequently, in order to make predictions of alternative states of the network, the speed of buses between stops have to be predicted based on a set of independent variables, that include information on the local network topology and surrounding levels of activity encountered along the way between stops.

The author developed a number of regression models for this purpose. A comparison between these models can be found in Chapter 4.

The link dynamics in the MATSim queue simulation can be easily modified, allowing one to assign arbitrary link travel times to individual vehicles. The initial Java class to assign stochastic travel times was written by Sergio Ordóñez, and then later modified by the author to work with travel times generated by exogenous models. The entire transit simulation process is discussed in more detail in Chapter 5.

2.4.6 Distributed re-planning and pseudo-simulation

Agent plans in a MATSim simulation are subjected to repeated mutations (“replanning”), during which aspects of the plan are changed, such as the network route, activity timing, mode choice, or secondary activity locations.
The time taken by running these mutations on plans can vary, but nearly all of them require a new set of routes through the transport system to connect each agent’s activities.

Because MATSim relies on a detailed network description with a lot of links, and transit networks can have even more links, the routing of plans can take a very long time. Experiments with the Singapore scenario show that even on the latest hardware, with up to 24 dedicated computer cores per server, the initial routing of plans through the network in preparation for the simulation can take at least two hours to complete for a 25% sample of agents, while the re-planning step can take twice as long as the mobility simulation for a reasonable mutation rate.

There is therefore a strong incentive to reduce the number of iterations, the time per iteration, as well as initial routing time, in order to make MATSim a viable transport planning tool for practice. Progress on the first objective was achieved by introducing a so-called pseudo-simulation step, whereby plans are first evaluated in a simplified simulation that runs at two orders of magnitude faster than the full queue simulation. In this way, plans can be subjected to multiple iterations of mutation and evaluation, and badly performing plans can be eliminated, obviating the need for them to be run in a costly queue simulation.

The second objective of reducing the time spent per iteration can be achieved in three ways: by designing improved replanning strategies that run a lot faster; increasing the number of cores in a Symmetric Multiprocessor System (SMP); or by distributing tasks across multiple machines. This final alternative was implemented by developing a framework with a so-called master-slave design pattern; the master node’s main responsibility being to run the queue simulation, and an arbitrary number of slave nodes providing it with agent plans to execute. Each slave node receives updated travel time information from the master, in order to run its own pseudo-simulation and rapidly evaluate newly generated plans.

The concept of pseudo-simulation is discussed in Chapter 6 while the distributed framework is described in Chapter 7.

2.4.7 Post-processing and analysis

The transit simulation system produces the usual set of MATSim results, which include the final set of plans in each agent’s memory, as well as events generated during the simulation that were written to disk. These simulation outputs tend to be massive (an uncompressed events file can be larger than 10 GB for the Singapore scenario) and can generally only be processed by Java classes that make use of the MATSim Application Programmer’s Interface (API).

The sheer volume of data produced by MATSim has given rise to specialised software packages
capable of handling it, such as senozon Via\(^4\). This software package processes MATSim events to produce a visualisation of vehicle movements, as well as provide analysis capabilities offered by most commercial transport planning suites, such as evaluating transit system performance indicators like ridership, or comparing simulated traffic volumes against those recorded at count stations in reality.

The author created a set of Java classes that will allow simulation results to be analysed using off-the-shelf business analytics software, such as Tableau\(^5\). This data processes MATSim events streams to compile a set of travel diaries for each agent, documenting their trips and composing stages and transfers, as well as the activities that the trips connect. Many of the analyses presented in Chapter 5 rely on output produced by these classes. The Java classes and their output are documented in the MATSim book (Erath and Fourie, 2016).

In the next chapter, the first critical component for realistic simulation dynamics will be discussed; how the transit smart card records were transformed into bus vehicle trajectories for nearly all services operating across Singapore on a Wednesday in April, 2011.

\(^4\)See [http://via.senozon.com](http://via.senozon.com).

\(^5\)See [http://www.tableausoftware.com](http://www.tableausoftware.com).
Chapter 3

Trajectory reconstruction

This chapter initially appeared as a working paper (Fourie, 2014a) and was compiled into a presentation at the first International Workshop on Utilizing Transit Smart Card Data for Service Planning, in Gifu, Japan (Fourie, 2014b).

The trajectory reconstruction process produces the input essential to the speed regression modelling presented in Chapter 4, as well as to validate simulation output.

3.1 Introduction

Trajectory reconstruction from transit smart card data includes constructing passenger trajectories or flows (e.g., Itoh et al., 2013; Yang et al., 2013; Yuan et al., 2013) or train trajectory reconstruction through regression analysis (Sun et al., 2012). So far, no attempts appear to have been made to date to reconstruct the detailed trajectories of bus vehicles from these data.

The reconstruction of bus trajectories could facilitate the analysis of transit performance measures on a per-vehicle basis. One could also identify causes of phenomena like bus bunching and interactions between different lines holding each other up at bus stops, and estimate models of bus speed and headway variability if other supporting information is available.

In this chapter, bus vehicle trajectories are reconstructed from a full day of smart card transactions in Singapore, from a mid-April Wednesday, 2011. The trajectories were recorded in the MATSim event stream format, using the specifications developed by Rieser (2010).

This standard makes visualisation and analysis possible in compliant software, such as Senozon Via (Senozon AG, 2015). Beside the application of estimating models of bus speed between stops, the output from the trajectory reconstruction process can also be used to route agents using realistic travel times in a dynamic transit router (Ordóñez Medina and Erath, 2013).
Figure 3.1: Problems in identifying bus dwell operations from smart-card transactions.

Bus ridership is inferred from transactions by time of day. Transactions are coloured by bus stop ID. The red text highlights possible problems in identifying the start and end of a dwell operation.

### 3.2 Challenges

Intuitively, in order to reconstruct bus trajectories from smart card records, one would simply

1. isolate the transactions for a single vehicle;
2. go through the list of transactions at each stop;
3. identify when the first user enters and the last user leaves the bus; and
4. assume that the bus arrived a few seconds before and departed a few seconds after these two respective events.

If these conditions are satisfied for each stop in the bus’s route, then you would have the travel time between each and every bus stop along the route. You could then reconstruct the trajectory between the stops using the description from the electronic transit schedule, by assigning a series of reasonable link travel times contingent on maximum speed and link length, for all links between each pair of stops.
Of course the actual problem is much more nuanced and complicated, as can be seen in Fig. [3.1]. This figure shows the transactions for a single bus between 6:40 and 7 o’clock in the morning. The number of passengers on the bus is inferred by keeping track of the number of tap-in and tap-out transactions. Each transaction is coloured to denote a unique bus stop identifier. A number of typical errors are highlighted.

**GPS errors and technical problems** may cause bus stop identifiers to be incorrectly identified. These errors might be due to stops being very close to each other, high buildings in the city centre confusing GPS signals, inclement weather conditions interfering with GPS, and so on.

**Early tap-outs and late tap-ins.** These are transactions that occur before or after larger clusters of transactions at the stop. A likely reason for early tap-outs is when a bus is prevented from entering a bus bay due to it being blocked by other traffic. As the bus approaches the stop, its GPS sensor activates the contactless smart card sensor and people can start to tap out even before the bus has come to a full stop. Sometimes the bus has to wait in a queue before it can enter the bus bay and safely open its doors, so a substantial amount of time can pass before the first tap-out is registered and passengers can actually start exiting the vehicle.

Late tap-ins, on the other hand, could be due to a user entering the bus, its door closes and it pulls away while the user frantically searches for their smart card in order to perform the transaction. Another possibility is that, especially in long bus bays that serve a number of lines, the bus takes on passengers at one end of the bus stop and then reopens its doors when it reaches the front of the bus bay in order to take on late passengers. Of course, it is also possible the bus driver simply waits at any given bus stop for somebody that he can see running to make the bus. The exact causes and how they manifest in the transit smart card data would require closer investigation. For the moment, trajectory reconstruction would have to rely on a number of global assumptions to deal with these behaviours.

**No transactions at stops.** The final significant problem is the case where buses pass by bus stops that have no passengers waiting and where no passengers on the bus need to get off. Of course, a bus that doesn’t register any transactions during its circuit run is completely invisible to the reconstruction procedure. If the bus doesn’t register any transactions for a number of stops at the beginning or end of its run, an extrapolation of its rate of movement will be pure conjecture. However if, as in the case of the bus in Fig. [3.1] only a number of intermediate stops were passed by without any transactions, one could make a more reasonable interpolation of its rate of movement from stop to stop, based on the maximum allowed speed and distances of the links between the stops.
Loop routes. Another subtle problem that does not appear in the figure is the case where a bus route is circular or loops in on itself, i.e. bus routes where one or more stops are visited more than once during a circuit run. These routes can make it difficult to identify when a circuit run has only started or whether it is at its end. The reconstruction procedure therefore has to check ahead to the next stop being visited if the route continues after a repeated stop; if the repeated stop is the first in the sequence of stops; or if the repeated stop observation is due to a GPS error.

3.3 Method

An overview of the trajectory reconstruction process appears in Fig. 3.2.

Initialisation The process starts with a number of preparatory operations, beginning with loading the supporting data structures, i.e. the network and electronic transit schedule. A number of look-up data structures are prepared in order to associate a bus’s set of transactions with a particular route, based on the line direction and bus stops it visited.

Identifying bus routes A connection is established with the smart card transaction database and transactions are drawn for every combination of bus Registration number, Line and Direction (RLD). This information is not enough to uniquely identify the route that the bus has
operated on for an entire day, because the same bus might be travelling in the same direction on the same transit line a number of times, but in one circuit run it might be operating the full route and in another it might be operating an express route or a shortened contingency service.

Bus lines not appearing in the transit schedule are ignored; this can happen due to simple oversight, or when transactions are recorded for a later date than the date at which the electronic transit schedule was compiled.

For a given RLD a set of possible routes are identified, and routes are ranked according to the number of times transactions at the set of stops in the route have been visited by the bus. The bus route with the highest number of transactions registered at its stops is assumed to be the route traversed by the bus and will form the basis of all further analysis for this particular set of transactions.

Correcting GPS errors In Fig. 3.3 the process for eliminating GPS errors is illustrated. It shows the ridership on a bus inferred from counting the number of boardings and alightings over time. Each transaction in Fig. 3.3(a) is coloured by a speed value, with slower transactions shifted towards blue and faster transactions shifted towards the red end of the spectrum. These speed values are calculated as the network distance between the bus stops associated with two consecutive transactions, divided by the time difference between the transactions.

An iterative process removes transactions with high speed values (arbitrarily taken to be more than 80 km/h) one at a time, and speeds are recalculated for all transactions after each removal. The number and timing of errant transactions is recorded, in order for the correct stop IDs to be assigned to these transactions in a later procedure that associates them with a dwell operation that overlaps with the errant transaction’s boarding or alighting time.

Clustering into dwell operations For each RLD combination, transactions are then processed by stop ID, and closely timed transactions are clustered to produce a vehicle dwell operation (arrival at and departure from a bus stop). For each dwell operation, the vehicle arrival and departure time is then taken from the largest sub-cluster of closely timed transactions, in order to eliminate false dwell operation timing from early tap-outs and late tap-ins.

If a boarding transaction is found before the assigned arrival time, or an alighting transaction is found after the assigned departure time, the dwell operation timing is adjusted. Departure and arrival is assumed to take place three seconds before and after the first and last transaction used to identify the dwell operation duration. The duration of dwell operations identified in this way is adjusted to a minimum of six seconds, for dwell operations where very few transactions occur.
Figure 3.3: Filtering of GPS errors using the ‘speed’ between consecutive transactions.

(a) Bus ridership coloured by transaction speed, red denotes high speed

(b) Transactions coloured by stop ID before filtering

(c) Transactions coloured by stop ID after filtering.

High transaction speeds in Fig. 3.3(a) correspond to sudden changes in colour (stop ID) in Fig. 3.3(b).

Transactions associated with the dwell operation, but that occur outside the dwell operation time window (due to tapping out early or tapping in late), have their timings adjusted to fit inside the window.
Figure 3.4: Interpolation of missing dwell operations, and removal of erroneously identified dwell operations.

The two graphs show dwell operations for a bus before and after missing operations have been identified. When an erroneous dwell operation had been identified (usually due to very late tap-ins or GPS errors where a single transaction occurs at a neighbouring stop, and that transaction is associated with the preceding stop), comparison against the route description identifies these and they are removed.

**Grouping dwell operations into circuit runs** In order for the output from the trajectory reconstruction process to be converted into a valid MATSim event stream, each individual circuit run, from first to last stop, needs to be identified for each RLD combination’s set of inferred dwell operations.

To this end, dwell operation order is evaluated against the sequence of stops visited in the associated route description, and each dwell operation is added to the circuit run if its bus stop follows that of the preceding dwell operation. If the dwell operation’s bus stop is not in order, it might indicate the start of a new circuit run or an incorrect bus stop ID associated with the dwell operation. This latter case is possible for bus stops that are relatively close to each other, and dwell operations with a single transaction that has a GPS error but was not removed in the preceding steps. If the dwell operations following this one continue the sequence of the preceding dwell operations, then the errant dwell operation is removed as a mis-classification, and its associated transactions are ignored; otherwise the dwell operation is associated with a new circuit run, and the process repeats.
Dwell event interpolation  Missing dwell operations (stops driven past with no transactions) are created in order to satisfy the requirements for the MATSim transit specification. The results of this process is illustrated in Fig. 3.4.

These dwell operations have a zero duration, and they are spaced in time by dividing the interval between the first and last observed dwell operations into a set of intervals that is proportional to the expected travel time between the set of bus stops that were passed by. These expected travel times between stops are currently based on the free speed travel time of each link, but this can be adjusted using the output from the regression models described in Chapter 4.

Constructing vehicle trajectories between dwell operations  A vehicle trajectory is written out in a separate file for each RLD combination in the MATSim XML format. The file records each dwell operation and its associated transactions, and the sequence of links the vehicle traverses between consecutive dwell operations, taken from the transit schedule. Given the time between consecutive dwell operations, the time spent on each link is taken as the time interval between stops, multiplied by the free-speed travel time of the link (not exceeding 80 km/h, assumed to be the maximum speed of a bus), divided by the total free-speed travel time of all links between stops. While the procedure does not account for congestion effects on the individual links between two stops, it does capture the total effect of congestion between two stops, in the total travel time.

Finally, the separate vehicle XML files are combined chronologically into a single output file using a merge-sort operation.

3.4 Applications

In this section a number of possible applications of the output from the procedure is illustrated, mainly in the form of descriptive analyses.

3.4.1 Average waiting times by time of day

In Fig. 3.5 we have two views of the Actual Waiting Time (AWT) distribution across the entire system in 15 minute intervals. AWT is calculated based on a definition by the London transport authorities (e.g. Schill, 2012) in Eq. (3.1), using a rolling window operating on the observed headways $H_i$ for the last $k$ dwell operations recorded for a particular line and direction at a
Figure 3.5: Average waiting time (AWT) distributions by time of day (TOD).

(a) AWT by TOD by dwell operation count

(b) AWT by TOD by passengers boarding.

The colours in the plots indicate the number of dwell operations and number of passengers affected. AWT is calculated using a rolling window operating on the last nine dwell operations recorded for a particular line and direction at a particular stop. Waiting time resolution is 15 seconds, while that of the x-axis is 15 minute time intervals.
Figure 3.6: Bus vehicle space-time graphs with rolling average wait time per stop.

In this plot, wait times are calculated for a rolling window of the last nine dwell operations at each stop, and plotted in 15 minute intervals as coloured background tiles. Overlaid lines denote individual vehicle space-time plots, and dot sizes denote the number of transactions recorded for the dwell operation.

In Fig. 3.5(a), for every 15 minute interval, the number of bus stops with AWTs within the appropriate interval are counted for each of the 15 second waiting time bins.

In the second plot, Fig. 3.5(b), the number of passengers boarding during the time of day- and waiting time intervals is shown. From this figure can be seen that most passengers experience waiting times in the order of six minutes in the morning peak, with a relatively larger spread than that observed during the evening peak. During intermediate hours waiting times are longer as bus frequencies drop outside peak hours.

\[
\text{AWT} = \frac{\sum_{i=k}^{i-k} H_i^2}{2 \sum_{i=k}^{i-k} H_i} 
\]  

(3.1)
3.4.2 Waiting times related to vehicle trajectories for a given bus service

Fig. 3.6 summarises the performance of bus services operating on a relatively long bus line in Singapore. The coloured background shows AWT at each stop, plotted at 15 minute intervals for a rolling window of nine dwell operations within and preceding each interval. Overlaid black lines are the time space plots that show the movement of each individual vehicle along the route. Circular dots of varying sizes indicate the number of passengers boarding at each stop for each vehicle.

The plot clearly illustrates how bus bunching leads to increases in AWT, especially at stops that are located towards the end of the line. Note that, around 4 o’ clock in the afternoon at bus stop 51, a short line appears, that does not extend back to the first stop. This line highlights a shortcoming in the current procedure; this particular circuit run probably only had its first transaction recorded at stop 51. Dwell operations and vehicle trajectories before and after the first and last recorded transaction for a circuit run are not extrapolated. However, the plot also reveals that a reasonable remedy to this shortcoming might be to assume that the bus had traversed the preceding stops at a rate intermediate to the buses before and after it.

3.5 Conclusion

The procedure, as it stands, is only a first attempt, but it does establish a methodology that can inform a machine learning approach, which can be the subject of further research.

The procedure is clearly useful for evaluating bus system performance using a variety of measures, and its output can be used in a set of interactive dashboards for daily operations analysis. Furthermore, it is useful to be able to see vehicle movements in compliant software such as senozone via, where dynamic vehicle interaction effects can be directly observed.

The process is available in the open source MATSim project as a set of Java classes that can be downloaded with the MATSim nightly build and be used with a compliant smartcard dataset.
Chapter 4

Localised prediction of bus speeds from reconstructed trajectories

As shown by [Sarlas and Axhausen (2015)](Sarlas and Axhausen (2015)), the speed of vehicles in a network link are related not only to the level of demand on the link, but also to the network topology, the presence of signalling systems and surrounding urban density and activity level. While their study modelled daily average speed for the entire Swiss road network of private and public transport, this investigation focuses on determining observed bus speeds at any given time of day as a function of network topology and indicators of the level of activity and demand that may be derived from the transit smart card data.

Such a regression model should make it possible to predict how fast buses will travel if the service frequency is changed, or possibly even how fast buses will travel between stops if the entire system were redesigned. For the current implementation of the data driven transit simulation, the travel times of buses in the simulation is derived from regression models that predict speed and its variability for all bus stop-to-stop combinations during the course of the simulated day.

4.1 Overview

The output from the trajectory reconstruction process forms the primary input to the speed prediction models that were tested in this chapter. As the trajectory reconstruction process generates MATSim events, these events are loaded into a Java program and processed using the MATSim [API](MATSim [API]). The Java program produces, for each combination of stops, and every service recorded in the trajectory reconstruction process, a travel time observation, along with a set of associated attributes. Attributes from the network topology, as well as dynamic quantities
associated with the bus traversal between transit stops, were compiled for the time of each
observation.

These data are then fed into the R statistical analysis package, where it is filtered for multi-
variate outliers. This filtering is required, as the stop-to-stop travel times from the trajectory
reconstruction process, as well as the variables calculated in the Java program, are subject to
various sources of error.

Only then is the data ready for regression analysis. The purpose of the analysis is to predict the
average and standard deviation of bus speeds between stops, in 15 minute intervals, throughout
the course of the day. These predictions will be fed into the modified transit simulation, to pro-
duce a distribution of speeds for every link connecting two stops, for every 15 minute interval
during the course of the day.

Besides ordinary least squares regression analysis, geographically weighted as well as spatially
auto-correlated regressions were also run, to investigate the influence of any localised effects.
A random forest regression analysis was also performed, to see if this technique produces a
marked improvement over the linear approaches.

4.2 Deriving topological and dynamic variables

A Java program was created using the MATSim API to process the event stream produced
during the trajectory reconstruction step. Besides the event stream, the program is fed the
original MATSim network, the transit schedule, vehicles file, as well as a list of links that were
found to be within 50 m of a traffic signal.¹

The Java program takes as arguments:

- A radius for the circle that determines the area of influence around each bus in the dataset,
as it travels along its trajectory. At the end of each link in the bus’s trajectory, features
that are within the circle around the bus are used in aggregate calculations. For this study,
the radius was set at $\sqrt{\frac{1}{\pi}}$ km, such that the circle will have an area of 1 km².

- A time value that serves as the bin size for dynamic variables. The average and standard
deviation of dynamic variables, associated with nodes in the network, are calculated for
each of the time bins of the specified size during the course of the day. A value of 15
minutes was used in this study.

- The number of threads to use in the analysis. A large number of calculations need to be

¹LTA kindly provided us with a shape file recording the location of all traffic signals in Singapore.
performed in order to construct the variables. A multithreaded approach speeds things up.

The program begins by streaming the events generated during the trajectory reconstruction, storing the travel time recorded for all stop-to-stop combinations, as well as the smart card transaction activity counts at every stop, for every 15 minute time bin. Then, for every combination of consecutive stops in the transit schedule, and every time bin, the links between the two stops are traversed in sequence. At the end of each link, the nodes within the pre-specified radius are processed for various attributes, and the average, or weighted average of these values for all nodes between the two stops are recorded. The distance between the two stops are also recorded as the sum of the length of all the links that connect them. Other attributes that are recorded include the length of the path between the two stops that has a dedicated bus lane, the average number of lanes and flow capacity along the path.

For each stop-to-stop combination recorded for every bus, the program then writes to disk a text file containing the origin and destination stop IDs, the bus’s departure and arrival time, and all the variables that were calculated for the time of day during which the bus had traversed the path between the two stops. For prediction purposes, these values are written to disk for every stop-to-stop combination in the transit schedule and every 15 minute time bin.

### 4.2.1 Variable listing

Table 4.1 provides a summary of all the variables that were compiled. While some variables are self-explanatory, others are described in more detail below. Not all of these variables made it into the final model, but their exclusion might also be of interest. The distribution of each of the variables produced by the Java program were examined in turn, as those with a large variance tend to have increased leverage on estimation results. The regression plane will tend to pass close to these isolated points, and have an inordinately large influence on coefficient estimates and, consequently, predictive power.

For variables that range over several orders of magnitude, the natural logarithm was tested in an initial Ordinary Least Squares (OLS) multiple regression, and the $R^2$ of the model was compared with a reference model where all variables were left untransformed. The better performing version of the variable was used in further estimations. Where applicable, Table 4.1 summarises both the variable and its natural logarithm. Some variables were found to be more significant if a weighted average value was used; where applicable those weighted averages were calculated as $\sum x^2/\sum x$.

**intersections_per_km** The number of nodes along the path of the bus between two stops that have more than one ingoing and one outgoing link or two ingoing and two outgoing links
(nodes denoting changes in direction for one-way or bi-directional roads, respectively, therefore not intersections), divided by the length of the path in kilometres.

**fraction_of_path_excl_buslane** A number of road segments in Singapore allow only buses in the leftmost lane; either exclusively for the entire day or during peak hours. This variable denotes the fraction of the path that has such a bus lane. The process does not take account of exclusivity by time of day, so this is a static variable.

**transit_system_accumulation** This variable tracks the number of passengers tapped in across the entire transit system, and is therefore an indication of overall system load by time of day.

**intersection_density** For each node in the path between two stops, draw a circle with a 1 km$^2$ area and count the number of intersections within that area according to the definition stated previously, then divide the sum by the number of nodes in the path.

**weighted_avg_intersection_complexity** For each intersection along the path, take the number of links that meet in the node as an indication of its relative complexity, as the more links that meet at an intersection affects the signalling times. The weighted average was found to be more significant.

**approx_activity_level_in_radius_along_path** For each node in the path between two stops, draw a circle with a 1 km$^2$ area around the node and retrieve all smart card transactions that have been recorded at transit stops within the circle. Assign each boarding transaction a value of -1, and each alighting transaction a value of +1, and find the running sum of the values by time of day. Subtract the minimum of the running sum from all its values, and use the resulting set of values as an indication of the number of activities that take place within the circle. For a given time of day the value of the running sum at each node is read from a table, and divided by the number of nodes in the path to arrive at an average value that is used in the regression.

**smartcard_transaction_rate_in_radius_along_path** For each node in the path between two stops at a given time of day, calculate the 15 minute moving average of the total number of transactions taking place per second within a 1 km$^2$ circle around the node, dividing the sum of these values by the number of nodes in the path. This value is used as an indication of the general level of traffic that the bus encounters along its path between stops.

**RMS_radians_turned** The angle between consecutive links in the path is measured in radians, and the root-mean-squared value is recorded. This value, along with path_length_over_euclidean_distance, is an indication of how winding or tortuous the path is.
minimum_capacity The minimum link flow capacity, in vehicles per hour, encountered along the length of the path.

squeeze_capacity For each intersection in the path, find the maximum difference between inbound and outbound flow capacity, in vehicles per hour; the hypothesis being that intersections with squeeze_capacity could be choke points and slow vehicles down as they approach and merge.

4.3 Data preparation

Both the trajectory reconstruction process, as well as the list of variables produced by the Java program, are subject to various sources of error. The most obvious are erroneous data points are those that have excessively fast speeds. So, as an initial filtering step, all data points with speed values above $90 \text{ km h}^{-1}$ were removed. The same was done for other variables in the data that were found to be obvious outliers. This initial filtering step reduced the number of stop stop observations from 1,512,415 to 1,473,515.

However, it is possible for an observation to not be an outlier in any of its attributes; instead the combination of attribute values might be highly unlikely, making it a Multivariate Outlier (MVO). It is important to remove these from the dataset, as MVOs can bias model coefficient estimates, even though their influence on average values for attributes might be negligible.

4.3.1 Multivariate outlier detection

Multivariate Outliers (MVOs) are a general problem in high-dimensional data analysis and can distort regression estimation results. In Filzmoser et al. (2008), the authors developed a method for identifying high-dimensional MVOs, that has been implemented in the R mvoutlier package (Filzmoser and Gschwandtner 2015). Data is scaled, centred, and PCA is performed. Distances between observations are calculated in principal component space, and used to calculate weights for location and scatter outliers. Combined weights, normalised to fall between 0 and 1, are used for outlier identification, making an observation more or less likely to be an MVO.

The algorithm identified 353,807 of the observations to be multivariate outliers. This left a total of 1,119,708 observations assumed to be valid for modelling purposes. Fig. 4.1 shows that the MVO detection algorithm tends to remove observations that lie towards the right of the speed distribution. This agrees with the intuition that, as the speed between bus stops is an average value, it is highly unlikely to regularly go over $50 \text{ km h}^{-1}$.
Figure 4.1: Distribution of speeds of valid data, compared with those classified as multivariate outliers.

Simple **OLS** regression using all the variables before and after the **MVO** elimination shows the leverage of multivariate outliers, and how upon their removal, the residual tends towards a normal distribution with little heteroscedasticity (Fig. 4.2). Table 4.1 summarises the input data after **MVOs** have been removed.

### 4.3.2 Centring and scaling

Following **MVO** the data is centred and scaled again. This is required to compensate for changes in the distribution of all variables. Consequently, following this step, all variables will have a mean of zero and a standard deviation of one. Naturally, the scaling and centring vectors are recorded in memory in order to reverse the transformation for predicted values.

By centring and scaling, each independent variable’s coefficient denotes by how many standard deviations the dependent variable changes for one standard deviation change in that particular independent variable. Consequently it is easier to compare the relative influence of variables, and is especially helpful for interpreting the influence of nonlinear (e.g. logarithmic or weighted average) variables, or variables that are difficult to interpret in terms of their absolute values.
<table>
<thead>
<tr>
<th>Variable</th>
<th>Min.</th>
<th>1st Quantile</th>
<th>Mean</th>
<th>Median</th>
<th>3rd Quantile</th>
<th>Max.</th>
<th>Std.Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed (m/s)</td>
<td>0.5069</td>
<td>4.301</td>
<td>6.583</td>
<td>5.973</td>
<td>8.034</td>
<td>24.98</td>
<td>3.282</td>
</tr>
<tr>
<td>ln_speed</td>
<td>-0.6795</td>
<td>1.459</td>
<td>1.773</td>
<td>1.787</td>
<td>2.084</td>
<td>3.218</td>
<td>0.4736</td>
</tr>
<tr>
<td>departure (hours)</td>
<td>5.148</td>
<td>10.12</td>
<td>14.78</td>
<td>14.91</td>
<td>19.05</td>
<td>25.01</td>
<td>5.143</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>11</td>
<td>30240</td>
<td>44240</td>
<td>36930</td>
<td>58090</td>
<td>92130</td>
<td>20990</td>
</tr>
<tr>
<td>approx_activity_level</td>
<td>8</td>
<td>619.1</td>
<td>1624</td>
<td>1143</td>
<td>2019</td>
<td>54850</td>
<td>1769</td>
</tr>
<tr>
<td>ln_approx_activity_level</td>
<td>2.079</td>
<td>6.428</td>
<td>6.991</td>
<td>7.042</td>
<td>7.611</td>
<td>10.91</td>
<td>0.9287</td>
</tr>
<tr>
<td>smartcard_transaction_rate</td>
<td>16</td>
<td>531.7</td>
<td>1311</td>
<td>899.9</td>
<td>1472</td>
<td>28100</td>
<td>1609</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate</td>
<td>2.773</td>
<td>6.276</td>
<td>6.768</td>
<td>6.802</td>
<td>7.295</td>
<td>10.24</td>
<td>0.9072</td>
</tr>
<tr>
<td>freespeed (m/s)</td>
<td>6.566</td>
<td>13.44</td>
<td>14.9</td>
<td>15.18</td>
<td>16.67</td>
<td>25</td>
<td>2.37</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>1.882</td>
<td>2.598</td>
<td>2.688</td>
<td>2.72</td>
<td>2.813</td>
<td>3.219</td>
<td>0.1681</td>
</tr>
<tr>
<td>path_length (m)</td>
<td>105.7</td>
<td>305.3</td>
<td>401</td>
<td>376.1</td>
<td>469.3</td>
<td>1814</td>
<td>143.4</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>4.661</td>
<td>5.721</td>
<td>5.938</td>
<td>5.93</td>
<td>6.151</td>
<td>7.503</td>
<td>0.3312</td>
</tr>
<tr>
<td>path_length_over_euclidean_distance</td>
<td>0.943</td>
<td>1.002</td>
<td>1.082</td>
<td>1.017</td>
<td>1.068</td>
<td>2.369</td>
<td>0.1498</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.05864</td>
<td>0.00191</td>
<td>0.07084</td>
<td>0.01696</td>
<td>0.06577</td>
<td>0.8627</td>
<td>0.1223</td>
</tr>
<tr>
<td>RMS_radians_turned</td>
<td>0.00031</td>
<td>0.08958</td>
<td>0.2057</td>
<td>0.1676</td>
<td>0.2838</td>
<td>1.571</td>
<td>0.1558</td>
</tr>
<tr>
<td>ln_RMS_radians_turned</td>
<td>-8.067</td>
<td>-2.413</td>
<td>-1.912</td>
<td>-1.786</td>
<td>-1.259</td>
<td>0.4516</td>
<td>0.9233</td>
</tr>
<tr>
<td>dwell_count_at_stop</td>
<td>20</td>
<td>367</td>
<td>627.3</td>
<td>587</td>
<td>840</td>
<td>2901</td>
<td>352.3</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>2.996</td>
<td>5.905</td>
<td>6.248</td>
<td>6.375</td>
<td>6.733</td>
<td>7.973</td>
<td>0.6844</td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>0</td>
<td>0</td>
<td>0.2829</td>
<td>0</td>
<td>0.732</td>
<td>1</td>
<td>0.4269</td>
</tr>
<tr>
<td>node_count_in_path</td>
<td>1</td>
<td>3</td>
<td>5.754</td>
<td>5</td>
<td>7</td>
<td>24</td>
<td>2.972</td>
</tr>
<tr>
<td>ln_node_count_in_path</td>
<td>0</td>
<td>1.099</td>
<td>1.621</td>
<td>1.609</td>
<td>1.946</td>
<td>3.178</td>
<td>0.5154</td>
</tr>
<tr>
<td>intersection_density</td>
<td>8</td>
<td>118</td>
<td>156.4</td>
<td>153.6</td>
<td>189</td>
<td>448.3</td>
<td>62.34</td>
</tr>
<tr>
<td>intersections_per_km</td>
<td>0</td>
<td>4.376</td>
<td>7.559</td>
<td>7.12</td>
<td>10.21</td>
<td>25.64</td>
<td>4.209</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>0</td>
<td>0</td>
<td>0.027</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.1621</td>
</tr>
<tr>
<td>right_turns_made_at_intersections</td>
<td>0</td>
<td>0</td>
<td>0.07863</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.2692</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>0</td>
<td>1</td>
<td>1.3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>0.9994</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>0</td>
<td>0</td>
<td>1.077</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.9065</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>0</td>
<td>1</td>
<td>2.485</td>
<td>2</td>
<td>4</td>
<td>11</td>
<td>2.05</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>0</td>
<td>1.8</td>
<td>1.977</td>
<td>2</td>
<td>2.273</td>
<td>4</td>
<td>0.5887</td>
</tr>
<tr>
<td>minimum_capacity (veh/hr)</td>
<td>396</td>
<td>3000</td>
<td>4192</td>
<td>3600</td>
<td>5400</td>
<td>13200</td>
<td>1907</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>0</td>
<td>0</td>
<td>1156</td>
<td>400</td>
<td>2000</td>
<td>8200</td>
<td>1528</td>
</tr>
<tr>
<td>weighted_avg_capacity</td>
<td>1758</td>
<td>3600</td>
<td>5305</td>
<td>5082</td>
<td>6256</td>
<td>13270</td>
<td>1880</td>
</tr>
<tr>
<td>weighted_avg_lane_count</td>
<td>1</td>
<td>1</td>
<td>1.634</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0.4805</td>
</tr>
</tbody>
</table>
Figure 4.2: Leverage before and after multivariate outlier elimination.

In the top figure, high leverage residuals are coloured darker than low leverage ones to make them easier to see.

4.4 Correlation and multicollinearity

Fig. 4.3 shows a correlation plot for all variables derived for the study. The variables with correlation greater than 0.5 or less than -0.5 appear in grey. Many variables show a very high degree of correlation. Consequently the model is expected to suffer from multicollinearity effects.
These include an increase in variance of coefficient estimates, as well as for coefficient estimates to be dependent on what other variables are in the model. Therefore, no firm conclusions can be drawn on the strength of effects of dependent variables in a model where there exists a high degree of multicollinearity.

To avoid this outcome, the modeller would have to make a selection of the available variables that would provide a satisfying trade-off between the predictive power of the model versus the degree of multicollinearity. The problem can be formulated as a multi-objective optimisation, and an efficient frontier of the two measures may be derived. An efficient frontier is a line in the plane of the two objective function variables that gives the optimal value of one variable
when the other is set as a constraint, for all feasible values of the second variable.

4.4.1 Trading off AIC vs VIF

If predictor variables are strongly correlated, then one might ask the question, what selection of variables will maximise predictive power while minimising the extent of autocorrelation in the model? If we consider that both predictive power and the degree of autocorrelation increase with increasing number of predictor variables, the question becomes instead, what selection of dependent variables would provide a good trade-off between the two quantities?

Variance Inflation Factor (VIF). The VIF is a well established measure of the degree of multicollinearity in a model. When variables in a model are correlated, their coefficient estimates are sensitive to the number of available observations. Removing or adding an observation can cause a substantial change in correlated coefficient estimates. The degree to which the variance of coefficient estimates are inflated by their collinearity compared to non-related variables is given by the VIF, which is defined as

$$VIF_k = \frac{1}{1 - R^2_k}$$  \hspace{1cm} (4.1)

where $R^2_k$ is the $R^2$ of a regression of the $k^{th}$ variable on the remaining predictor variables.

Acceptable values for the VIF vary across sources, but none recommend a value greater than 5. As a model containing all of the dependent variables has a VIF of less than 5, it would be acceptable by most standards. However, it is an interesting exercise to still investigate the trade-off, as it illustrates how much stock we should put in our interpretation of the variable coefficient estimates.

Akaike Information Criterion (AIC). The AIC of the model is taken as a measure of its predictive power.

For $k$ model parameters and maximum likelihood value $L$, $AIC = 2k - 2 \ln L$. AIC is a relative measure, allowing one to compare models with each other. In model selection, the preferred model would be the one with minimum AIC, thus the one that minimises the number of parameters (avoiding over-fitting) while maximising predictive power (maximum likelihood).

Determining the minimum AIC as a function of VIF. Both AIC and VIF are nonlinear quantities dependent on the outcome of an entire OLS regression, which is itself a non-linear minimisation. The selection of variables is binary. The number of combinations of variables
precludes the possibility of an enumeration of all possible solutions. It was intuited that a meta-heuristic should be employed to provide a satisfactory set of solutions that can serve as an approximate efficient frontier.

**Optimisation method.** Differential Evolutionary Optimisation (DEO) ([Price et al., 2006](#)) is a robust, general purpose optimisation meta-heuristic that works well with general non-linear functions that need not have an analytical form. It initialises with a population of candidate solutions in the search space, then uses simple mathematical techniques to combine these candidate solutions into a new generation of solutions that are more likely to minimise the objective function. It is implemented in the R package, [DEOptim](#) ([Mullen et al., 2011](#)). DEO has relatively few parameters; generally increasing the population size and number of iterations improves solution quality at the cost of speed, which can be counteracted by increasing the number of computational cores used to evaluate the function in parallel.

**Objective function.** A number of values for VIF was selected, and an optimal AIC was found for each VIF in turn. An algorithmic function was defined which takes a vector of binary values as its input; the vector having the same length as the total number of possible variables to include in the model. The binary values each determine whether the variable be included in an OLS regression. The VIF of the model is calculated, as well as its AIC. If the VIF is more than the current value under consideration, the function returns a large value, otherwise it returns the AIC. Minimisation in DEoptim will therefore produce the set of variables with the most negative AIC for the VIF under consideration, or something reasonably close to it.

**Efficient frontier and coefficient estimates.** The results of the optimisation are shown in Fig. 4.4, along with a graphical illustration of the variation in coefficient values for the eight most recurring variables. A complete set of parameter estimates for a selection of VIF values also appear in Table 4.2. This table also includes the AIC values for each model for comparison.

From the efficient frontier plot it appears that AIC decreases substantially up to a VIF of 2.01, after which it does not change significantly with increasing VIF (and number of variables). The optimal model for a VIF of 2.01 requires only 15 of the 23 variables under consideration. It was decided, therefore, that this model be used in all further investigations.

Fig. 4.4(b) shows the ratio between the coefficient for the multiple regression and its value in a simple regression, i.e. where it’s the only predictor in the model. While none of the variables change in sign, there is substantial variation in their magnitude depending on what other variables appear with them in a multiple regression, especially for smaller values of VIF. This plot drives home how relative the interpretation of variable coefficient values can be in a regression study. From this perspective, there is perhaps not much merit in coefficient interpretations,
Figure 4.4: Efficient frontier of VIF and AIC, compared with coefficient ratios of efficient models.

(a) Efficient frontier of VIF vs. AIC for predicting $\ln(speed)$. Subscripts denote number of variables in each model.

(b) Model coefficient ratios for 8 most recurring variables, for the models shown in the efficient frontier plot.

The coefficient ratio is taken as the coefficient value for each model, divided by its value in a simple linear regression; i.e. a model where it is the only dependent variable.
at least for this particular application. Rather, it might be more important to find out which variables provide the model with more predictive power than others.

### 4.5 Comparison of instantaneous vs. hourly, speed vs. travel time prediction

All of the parameter estimates shown in Table 4.2 was for hourly averages. $R^2$ values do not compare well with those found by Sarlas and Axhausen (2015), which used daily averages. This section therefore shortly investigates the influence of the time horizon in the quality of predictions.

For comparison, OLS was run on the individual stop-to-stop observations (called, for convenience, the ‘instantaneous model’). Predicted values for each stop-to-stop combination were averaged across the day, and the hourly vs instantaneous models’ averages compared against the true daily averages, and expressed as a ‘quasi-$R^2$', which is the $R^2$ of a simple regression where the predicted values become the only independent variable.

The results are shown in Table 4.3. The predictive power of the model for daily averages approaches 0.5, which is still less than the values found for the Swiss study, but substantially better than the hourly value. The hourly model, in turn, performs better than the instantaneous one.

It is therefore safe to conclude that increasing the time horizon increases the predictive accuracy of regression models, as the variance in instantaneous predictions that is not explained by the model parameters is subsumed by the process of averaging.

An interesting finding is that, when travel time is calculated from the predicted speed value and compared with the actual value, it has a substantially larger quasi-$R^2$ than the speed model that produced it, as can be seen in the bottom of Table 4.3. This increase in $R^2$ is due to the error in predicted travel time being inversely proportional to the error in predicted speed.

By definition, for a data set of values $[y_1 \ldots y_i \ldots y_n]$, with a mean $\bar{y}$, and a set of predicted values $[f_1 \ldots f_i \ldots f_n]$, the coefficient of determination is defined as $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$, where the sum of squares of the residuals, $SS_{res} = \sum (y_i - f_i)^2 = \sum \epsilon_i^2$ and the total sum of squares, $SS_{tot} = \sum (y_i - \bar{y})^2$.

As the total sum of squares is constant, $R^2$ is determined by the residual sum of squares.

We therefore need an indication of how the residual of travel time depends on predicted speed.

---

2Summary of the Wikipedia page for “Coefficient of determination”.
<table>
<thead>
<tr>
<th></th>
<th>1.02</th>
<th>1.11</th>
<th>1.21</th>
<th>1.51</th>
<th>2.01</th>
<th>5.01</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VIF</strong></td>
<td>1.02</td>
<td>1.11</td>
<td>1.21</td>
<td>1.51</td>
<td>2.01</td>
<td>5.01</td>
</tr>
<tr>
<td><strong>AIC</strong></td>
<td>3.431×10^4</td>
<td>2.975×10^4</td>
<td>2.894×10^4</td>
<td>2.833×10^4</td>
<td>2.749×10^4</td>
<td>2.732×10^4</td>
</tr>
<tr>
<td>Adjusted R^2</td>
<td>0.05586</td>
<td>0.2973</td>
<td>0.3331</td>
<td>0.359</td>
<td>0.3931</td>
<td>0.4</td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.0154</td>
<td>–0.02947</td>
</tr>
<tr>
<td>Intercept</td>
<td>0.04946</td>
<td>0.02067</td>
<td>–0.02522</td>
<td>–0.02105</td>
<td>–0.006757</td>
<td>–0.003368</td>
</tr>
<tr>
<td>intersection_density</td>
<td>–</td>
<td>–0.2722</td>
<td>–</td>
<td>–</td>
<td>–0.0873</td>
<td>–0.06742</td>
</tr>
<tr>
<td>intersections_per_km</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.01979</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>–0.05293</td>
<td>–</td>
<td>–0.03267</td>
<td>–0.03065</td>
<td>–0.03497</td>
<td>–0.06107</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>–</td>
<td>–</td>
<td>–0.1161</td>
<td>–0.05701</td>
<td>–0.06107</td>
<td>–0.06107</td>
</tr>
<tr>
<td>ln_approx_activity_level_in_radius_along_path</td>
<td>–</td>
<td>–</td>
<td>–0.06931</td>
<td>–0.06465</td>
<td>–0.0459</td>
<td>–0.03498</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.05374</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.07361</td>
</tr>
<tr>
<td>ln_node_count_in_path</td>
<td>–0.1324</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–0.05499</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>–</td>
<td>0.1787</td>
<td>0.2605</td>
<td>0.2868</td>
<td>0.283</td>
<td>0.3194</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>–</td>
<td>–0.1423</td>
<td>–0.1228</td>
<td>–0.1196</td>
<td>–0.09031</td>
<td>–0.07092</td>
</tr>
<tr>
<td>ln_RMS_radians_turned</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>–</td>
<td>–</td>
<td>–0.2328</td>
<td>–0.2044</td>
<td>–0.1475</td>
<td>–0.1225</td>
</tr>
<tr>
<td>minimum_capacity</td>
<td>–</td>
<td>–</td>
<td>0.06848</td>
<td>0.06866</td>
<td>–</td>
<td>0.04388</td>
</tr>
<tr>
<td>right_turns_made_at_intersections</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–0.03075</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>–</td>
<td>–</td>
<td>–0.1297</td>
<td>–0.1009</td>
<td>–0.04572</td>
<td>–0.05033</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>–</td>
<td>–</td>
<td>0.00184</td>
<td>0.00291</td>
<td>–0.02083</td>
<td>–0.009349</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.1581</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>–</td>
<td>–0.1313</td>
<td>–</td>
<td>–0.03904</td>
<td>–0.06452</td>
<td>–0.07776</td>
</tr>
<tr>
<td>weighted_avg_capacity</td>
<td>0.09445</td>
<td>0.002586</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>–</td>
<td>–0.1141</td>
<td>–</td>
<td>–0.02126</td>
<td>–0.04098</td>
<td>–0.03779</td>
</tr>
<tr>
<td>weighted_avg_lane_count</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–0.04766</td>
<td>–</td>
</tr>
<tr>
<td>Variable</td>
<td>Hourly Estimate</td>
<td>Std. Error</td>
<td>t value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intercept</td>
<td>0.0000</td>
<td>0.001</td>
<td>0.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>0.02734</td>
<td>0.00101</td>
<td>0.003198</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>intersection_density</td>
<td>-0.09277</td>
<td>0.001079</td>
<td>0.002836</td>
<td>-85.97 -30.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>-0.0209</td>
<td>0.0009174</td>
<td>0.00165</td>
<td>-22.78 -18.86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>-0.04941</td>
<td>0.001038</td>
<td>0.002684</td>
<td>-47.6 -21.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>-0.06883</td>
<td>0.001034</td>
<td>0.002194</td>
<td>-66.54 -21.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.05393</td>
<td>0.0009443</td>
<td>0.002143</td>
<td>57.11 23.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_path_length</td>
<td>0.2776</td>
<td>0.001005</td>
<td>0.002484</td>
<td>276.2 116.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.1151</td>
<td>0.0009744</td>
<td>0.002062</td>
<td>-118.1 -42.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>-0.1472</td>
<td>0.001088</td>
<td>0.002632</td>
<td>-135.2 -54.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>-0.04198</td>
<td>0.001052</td>
<td>0.002609</td>
<td>-39.9 -19.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>-0.01311</td>
<td>0.000898</td>
<td>0.00219</td>
<td>-14.61 -10.53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>-0.1473</td>
<td>0.001139</td>
<td>0.002911</td>
<td>-129.3 -54.06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>-0.06253</td>
<td>0.0009365</td>
<td>0.002453</td>
<td>-66.77 -27.23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>-0.05977</td>
<td>0.000999</td>
<td>0.002608</td>
<td>-59.83 -16.85</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Residual standard error: 0.5844 0.8938
Adjusted $R^2$: 0.3910 0.2011

Predictions per stop-to-stop combination, averaged over the entire day

<table>
<thead>
<tr>
<th>Variable</th>
<th>Hourly Estimate</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>0.0000</td>
<td>0.001</td>
<td>0.0000</td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>0.02734</td>
<td>0.00101</td>
<td>0.003198</td>
</tr>
<tr>
<td>intersection_density</td>
<td>-0.09277</td>
<td>0.001079</td>
<td>0.002836</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>-0.0209</td>
<td>0.0009174</td>
<td>0.00165</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>-0.04941</td>
<td>0.001038</td>
<td>0.002684</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>-0.06883</td>
<td>0.001034</td>
<td>0.002194</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.05393</td>
<td>0.0009443</td>
<td>0.002143</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>0.2776</td>
<td>0.001005</td>
<td>0.002484</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.1151</td>
<td>0.0009744</td>
<td>0.002062</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>-0.1472</td>
<td>0.001088</td>
<td>0.002632</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>-0.04198</td>
<td>0.001052</td>
<td>0.002609</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>-0.01311</td>
<td>0.000898</td>
<td>0.00219</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>-0.1473</td>
<td>0.001139</td>
<td>0.002911</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>-0.06253</td>
<td>0.0009365</td>
<td>0.002453</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>-0.05977</td>
<td>0.000999</td>
<td>0.002608</td>
</tr>
</tbody>
</table>

Residual standard error: 0.4803 0.485
Quasi-$R^2$: 0.475 0.473

Instantaneous and hourly **travel time** predictions per stop-to-stop combination

<table>
<thead>
<tr>
<th>Variable</th>
<th>Hourly Estimate</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>0.0000</td>
<td>0.001</td>
<td>0.0000</td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>0.02734</td>
<td>0.00101</td>
<td>0.003198</td>
</tr>
<tr>
<td>intersection_density</td>
<td>-0.09277</td>
<td>0.001079</td>
<td>0.002836</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>-0.0209</td>
<td>0.0009174</td>
<td>0.00165</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>-0.04941</td>
<td>0.001038</td>
<td>0.002684</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>-0.06883</td>
<td>0.001034</td>
<td>0.002194</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.05393</td>
<td>0.0009443</td>
<td>0.002143</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>0.2776</td>
<td>0.001005</td>
<td>0.002484</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.1151</td>
<td>0.0009744</td>
<td>0.002062</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>-0.1472</td>
<td>0.001088</td>
<td>0.002632</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>-0.04198</td>
<td>0.001052</td>
<td>0.002609</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>-0.01311</td>
<td>0.000898</td>
<td>0.00219</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>-0.1473</td>
<td>0.001139</td>
<td>0.002911</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>-0.06253</td>
<td>0.0009365</td>
<td>0.002453</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>-0.05977</td>
<td>0.000999</td>
<td>0.002608</td>
</tr>
</tbody>
</table>

Residual standard error (seconds): 21.7 33.3
Quasi-$R^2$: 0.579 0.343
values. The vector of speed values in the data set can be stated as

\[ v = f_v + \epsilon_v \]  \hspace{1cm} (4.2)

with \( f_v \) the predicted value and \( \epsilon_v \) the residual. Similarly, travel time has a component predicted by the speed model as well as a residual, or

\[ t = \frac{s}{f_v} + \epsilon_t \]  \hspace{1cm} (4.3)

where \( s \) denotes distance. Then, if

\[ t = \frac{s}{v} \]  \hspace{1cm} (4.4)

substitution of (4.2) and (4.3) yields:

\[ \frac{s}{f_v} + \epsilon_t = \frac{s}{f_v} + \epsilon_v \]  \hspace{1cm} (4.5)

Eq. (4.5) can be solved for \( \epsilon_t \):

\[ \epsilon_t = \frac{-s\epsilon_v}{f_v^2 + \epsilon_v f_v} \]  \hspace{1cm} (4.6)

Assuming that residuals from the speed model (\( \epsilon_v \)) are small compared to predicted values (\( f_v \)), the residual for travel time will drop off dramatically with increasing predicted speed value and, consequently, quasi-\( R^2 \) for travel time will be improved due to the inverse-square relationship with predicted speed.

4.6 Interactions

A number of interactions were investigated to test some basic hypotheses formed from observation. For instance, when a bus is in a dedicated bus lane, but has to cross an intersection with a left turn, then other vehicles are allowed to enter the bus lane and turn left, slowing the bus down. One would therefore expect a negative coefficient for the product of fraction_of_path_excl_buslane and left_turns_passed_at_intersections. This was found to be the case, but the effect on \( R^2 \) was negligible, as it was with the other intuitively important interactions that were investigated.

A way of finding the most significant interactions in a regression model is to estimate a tree model of the data, which partitions the dataset along its dependent variables in such a way that
The plot shows that, e.g. if the smart card transaction rate along the path is less than -0.293 (so left at first branch), and intersection density is more than -0.8165 (right at second branch), the mean value of the dependent variable (standardised $\ln(speed)$) is 0.1653 (the second-from-left leaf of the tree). This partitioning of the dataset exposes maximum conditional variance, and so forms a good basis for defining interaction variables in a linear model.

From this model, a set of dummy variables were created, and their interactions tested, which improved the $R^2$ of the instantaneous model by 0.032. It was concluded that, rather than examine interactions to improve predictive power, one might rather look towards machine learning techniques. The tree model structure inspired the selection of the random forest approach as a method of interest.

### 4.7 Random forest regression

A number of machine learning methods were tested for comparison against the OLS regression results, as well as the other methods discussed in this chapter. RF regression was selected from a pragmatic point of view, as the method

- is fast to estimate compared to many of the other machine learning methods tested;
- has few parameters to optimise;
• is intuitive to interpret;

• allows the ranking of variables in terms of their importance;

• has a good implementation in R, in the \texttt{randomForest} package (Liaw and Wiener, 2002).

Explained quite simply, a RF is a set of a pre-determined number of regression trees, and its result is an averaging of the results produced by those trees. A regression tree is similar to the tree model in Fig. 4.5. For each of the nodes in the tree, up to a depth of \( n \) nodes, from a selection of \( m \) variables in the data, a value of each independent variable is selected such that the averages of the the dependent variable on either side of that value minimises the sum of squares difference with the actual values of the data points.

For a given observation, a regression tree’s predicted value can be found by tracing a path down its branches to the values found at its leaf nodes. A RF simply produces the average predicted values from a predetermined number of trees. The exact number of trees, the depth of the individual trees \( n \), and the number of variables selected at each node \( m \), can be determined through hyper-parameter optimisation.

4.7.1 Training

The model was trained against a 20\% sample of the stop-to-stop records, for 200 iterations. A simple hyper-parameter optimisation procedure that maximises the quasi-\( R^2 \) was run to find a reasonable set of values for the most important RF parameters, including:

• the number of variables randomly sampled as candidates at each split;

• the minimum size of terminal nodes, so how much of the share of dependent variable values end up at each terminal node, regulating the size of trees that are grown;

• sample size to be drawn for bagged sampling.

4.7.2 Results

The RF approach produced a radical improvement over the OLS regression. The quasi-\( R^2 \) values for both instantaneous as well as hourly predictions came to nearly double that of OLS. RFs do not produce coefficient estimates, as the result is a compound object composed of many trees, specifying the covariance structure of the data, for arbitrary combinations of variables.
However, it is not a complete black box, so there are ways for us to directly obtain at least an understanding of the importance of variables in producing the result. Indeed, as was argued in Section 4.4.1, this level of understanding might be more useful than a coefficient estimate, as coefficients will vary substantially when a model exhibits even a slight degree of multicollinearity.

Fig. 4.6 is an illustration of two such importance measures. The height of the bars show the influence of random permutation of variable input values on the Mean Squared Error (MSE); so replacing all values of transit_system_accumulation by randomly sampled values from its distribution will increase the MSE by more than 90%. The colour of the bar indicates the reduction in the Residual Sum of Squares (RSS) is between 10,000 and 15,000 when splitting on this variable. From this plot, it is clear that the dynamic variables are very important to the model’s predictive power, as two of the top three variables for both measures of variable importance are dynamic.
4.8 Modelling stochasticity

It was always a governing assumption of the modelling approach that bus speeds observed in reality exhibit stochasticity, and that the simulation model should reflect that. The output of the model would therefore have to be a set of parameters to produce realistic distributions of bus speeds during the course of the day. As the logarithm of speeds in the data appears to be normally distributed for most stop-to-stop combinations, the simulation samples for each combination from a normal distribution with given mean and standard deviation for that time of day (currently the resolution is 15 minute intervals).

It is therefore not enough to only predict average speed values; an additional model has to formulated to predict the standard deviation of the speed values by time of day.

4.8.1 OLS regression results

This section compares results from an OLS and RF regression model of standard deviation of $ln(speed)$, with the same set of variables that drives the main model. The results of the OLS regression of hourly standard deviation by stop-to-stop combination are shown in Table 4.4. The variables remain centred and scaled, as with all the previously estimated models.

The estimation results show that the $R^2$ value for this model is about half of the model for $ln(speed)$. As we have many observations, significance of variables tends to increase with the magnitude of the coefficient estimate.

As with the model of $ln(speed)$, we should be careful about attaching too much value to our interpretation of the relative strength of coefficient estimates, as the model suffers from the same multicollinearity effects. However, at the very least the signs of the biggest coefficient estimates are in line with expectations, e.g. increasing path length decreases variability, as does its most strongly correlated counterpart, the number of nodes in the path.

What is somewhat surprising is that $ln(speed)$ is insignificant, and its coefficient is nearly 0. However, this finding is borne out by the fact that the model of $ln(speed)$ displays very little heteroscedasticity, and the spread of the residual is approximately constant across all values of the dependent variable.

4.8.2 RF regression results

The RF variable importance plot for speed standard deviation is shown in Fig. 4.7. The RF was grown using a 20% sample size of all hourly standard deviations for all stop-to-stop combina-
Table 4.4: Model of standard deviation of speeds per hour.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Estimate</th>
<th>Std. Error</th>
<th>t value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>0.6939494</td>
<td>0.0011552</td>
<td>600.7176810</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>-0.0452330</td>
<td>0.0018037</td>
<td>-25.0781626</td>
</tr>
<tr>
<td>ln_node_count_in_path</td>
<td>-0.0343892</td>
<td>0.0021250</td>
<td>-16.1830423</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>0.0329773</td>
<td>0.0015086</td>
<td>21.8594912</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>0.0294973</td>
<td>0.0012306</td>
<td>23.9694912</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>0.0286627</td>
<td>0.0010793</td>
<td>26.5563714</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>0.0269224</td>
<td>0.0015702</td>
<td>17.1461181</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>0.0220992</td>
<td>0.0013034</td>
<td>16.9549475</td>
</tr>
<tr>
<td>minimum_capacity</td>
<td>-0.0202257</td>
<td>0.0019893</td>
<td>-10.1673123</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>-0.0189429</td>
<td>0.0008035</td>
<td>-23.5767835</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.0134197</td>
<td>0.0012877</td>
<td>-10.4217248</td>
</tr>
<tr>
<td>weighted_avg_lane_count</td>
<td>0.0116605</td>
<td>0.0016448</td>
<td>7.0893763</td>
</tr>
<tr>
<td>weighted_avg_capacity</td>
<td>0.0110938</td>
<td>0.0021120</td>
<td>5.2526397</td>
</tr>
<tr>
<td>intersection_density</td>
<td>0.0080723</td>
<td>0.0013717</td>
<td>5.8848426</td>
</tr>
<tr>
<td>intersections_per_km</td>
<td>0.0078615</td>
<td>0.0019162</td>
<td>4.1026814</td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td>0.0041117</td>
<td>0.0022037</td>
<td>1.8658449</td>
</tr>
<tr>
<td>right_turns_made_at_intersections</td>
<td>0.0035097</td>
<td>0.0010033</td>
<td>3.4980585</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>0.0031294</td>
<td>0.0015637</td>
<td>2.0012918</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>0.0027850</td>
<td>0.0010648</td>
<td>2.6156380</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>-0.0023816</td>
<td>0.0013415</td>
<td>-1.663892</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.0016848</td>
<td>0.0013246</td>
<td>1.2548853</td>
</tr>
<tr>
<td>ln_speed</td>
<td>-0.0011233</td>
<td>0.0017014</td>
<td>-0.6602217</td>
</tr>
<tr>
<td>ln_RMS_radians_turned</td>
<td>0.0008444</td>
<td>0.0012012</td>
<td>0.7029961</td>
</tr>
<tr>
<td>ln_approx_activity_level_in_radius_along_path</td>
<td>0.0003937</td>
<td>0.0013281</td>
<td>0.2964510</td>
</tr>
</tbody>
</table>

Residual standard error : 0.2645
Adjusted $R^2$: 0.129

All values are expressed as standard deviations of natural logarithm of speed. Variables (except for intercept) are ordered by decreasing coefficient magnitude. t-values greater than 2.6 can be considered significant.

It was found that the model does not improve much beyond 100 trees, and default values were used for other hyper-parameters (no limit on tree depth, number of variables sampled for each tree = n/3, and a minimum of 5 observations on terminal nodes).

The RF approach produces a better fit, with a quasi-$R^2$ value of 0.3705. The variable importance plot shows good agreement with the magnitude of coefficient estimates in the OLS model, at least from the perspective of increase in RSS upon random permutation of variables. There are two surprises, the first is transit system accumulation, which increases node purity (decreasing RSS) but does not significantly perform on the other measure.

The other is that, in the RF model, speed is a very significant variable, compared to the OLS regression result. In the variable importance plot, it ranks in the top 4 for both measures of importance, while, in Table 4.4, it has a small coefficient value and low significance. This indicates that there is a more complex relationship between hourly average speed and its variance that cannot be captured by OLS.
Figure 4.7: Random forest variable importance plot for standard deviation model.

Colour denotes the reduction in regression sum of squares when trees are split on that variable.

4.8.3 Comparison of simulated results versus actual

In order to get a preview of the distribution of speeds that will be produced by the simulation, instantaneous predictions were generated for both $\ln(speed)$ and its standard deviation, for each observation in the data set. For each combination, a simulated value was generated by sampling from a normal distribution with its parameters set to the combination’s values. The resulting values are plotted against the actual values for both OLS and RF models, at two stop-to-stop combinations, in Fig. 4.8 and Fig. 4.9. The two figures show simulated vs actual speeds for two regions, the first being in the central shopping district where speeds are generally slower, and the second for a sparsely-built industrial area, where the buses run faster.

In both figures, it is clear that the RF approach more clearly follows the trend of actual observations, which tend to be pretty noisy. Especially for the case of the industrial area, the RF approach performs significantly better than OLS. In both cases, the simulated values (in blue) produce distributions that match well with observations, as can be seen from the density plots.
Figure 4.8: Comparison of OLS vs RF speed profiles for a stop-to-stop combination in the shopping district.

(actual values in black, predicted in red and simulated values shown as dashed blue lines. Predictions are instantaneous. Simulated values are a single realisation for each observation, sampled from a normal distribution with the predicted value as average, and a predicted standard deviation from the SD model in the right-hand margins.

The RF simulated values tends to produce some outlier values that are much larger than what would be realisable in reality for both cases. In such cases, simulated values will be truncated to a maximum of 90 km h$^{-1}$. 

Figure 4.9: Comparison of OLS vs RF speed profiles for a stop-to-stop combination in a sparsely-built industrial area.

(a) OLS vs actual

(b) RF vs actual

Actual values in black, predicted in red and simulated values shown as dashed blue lines. Predictions are instantaneous. Simulated values are a single realisation for each observation, sampled from a normal distribution with the predicted value as average, and a predicted standard deviation from the SD model.

Fig. 4.10 shows the density of the actual distribution of speeds, versus those from the RF point prediction, as well as the simulated values. Overall, the approach produces a distribution of speeds that matches very well with that observed in reality.
Figure 4.10: Comparison of RF point prediction density vs. that produced by stochastic model and the original source data.

The stochastic values are produced by taking the point prediction of $\ln(speed)$ as mean, and the random forest prediction of standard deviation for the observation, and sampling from a normal distribution with those attributes.

### 4.9 Temporal autocorrelation

As bus bunching is a significant effect in transit networks, the expectation is that bus speeds should be autocorrelated, as the buses are forced to travel at the same speed when they catch up with each other. This autocorrelation should be over and above the agreement one expects between consecutive observations at a given stop-to-stop combination due to the similarly-valued independent variables influencing speed, as these are not expected to change much from one observation to the next. Specifically, therefore, the expectation is that speeds would be positively autocorrelated; slow speed observations would follow other slow observations, and the same for fast speeds.

However, as Fig. 4.8 and Fig. 4.9 have shown, there is no clear positive autocorrelation to be seen in actual speed observations beyond the variations captured by the regression model. Indeed, speeds really appear to be log-normally distributed around this underlying signal.

The actual and simulated speed values were investigated for their autocorrelation characteristics, to see if the spectral characteristics of actual and simulated values are similar. The first step was to remove the effect of any systematic variation. A simple method for achieving this, frequently used in time series analyses, is ‘first differencing’; simply taking the value of each observation and subtracting that of its predecessor. This was done on a stop-to-stop basis, and
is illustrated for a single case in Fig. 4.11. The degree of autocorrelation can then be calculated by comparing the value of each first difference observation with its predecessor, and determining the slope of the simple regression line that characterises this relationship. Fig. 4.12 shows such a comparison, both for actual and simulated first differences. The two plots have a trend line slope of approximately -0.49, which is close to the average value for all stop-to-stop combinations.

A signal can be characterised by looking at the degree of correlation, not only between observations and their predecessors (a lag of one observation), but also by comparing each observation with its predecessor’s predecessor (a lag of 2), and so on. Fig. 4.13 compares, for all stop-to-stop combinations, the distribution of autocorrelation in the actual, predicted and simulated values for a lag of 1 and 2. The distributions of simulated values show good agreement with the actual autocorrelation structure, which is not captured by using the RF prediction on its own.

4.10 Spatial autocorrelation

There are clear spatial patterns in the residual of the OLS regression, as can be seen in the second plot in Fig. 4.14. Red points tend to have red points in their immediate neighbourhood, and likewise for blue points. The points in this plot is the geometric midpoint for each stop-to-stop combination, and each point’s residual value is for a randomly selected hourly average, uniformly sampled between 6am and 11pm.

A standard test for autocorrelation, the Moran’s I test, confirms the presence of positive autocorrelation, with the OLS regression showing a value of 0.07, considering the 10 nearest neighbours of each point. Three techniques to quantify the degree of spatial autocorrelation in the parameter estimates as well as the residual were investigated: Geographically Weighted Regression (GWR), spatial simultaneous autoregressive lag (LAGSAR) and spatial simultaneous autoregressive lag and error (SACSAR). The outcomes from these regressions were compared also against the RF results.

4.10.1 Background

4.10.1.1 Geographically Weighted Regression (GWR)

In GWR, spatial autocorrelation is captured by performing localised regression, thus selecting a point and its neighbours, estimating parameter coefficients, and performing a weighted averaging of coefficient estimates for all neighbourhoods that each point forms a part of. The size of the neighbourhood can be fixed or variable, and is optimised before performing the repeated
Figure 4.11: Temporal autocorrelation for a single stop-to-stop combination.

(a) Recorded speed observations between two stops in an urban area.

(b) Difference between consecutive observations.

By taking the difference between consecutive observations (first differencing), systematic variation is reduced and negative autocorrelation is exposed, as data points alternate above and below the zero line in the second plot. Note that the first plot shows the speed in kilometres per hour, while the second expresses it as number of standard deviations of the natural logarithm for all speeds in the dataset; the unit used in the standardised regressions.

Regression GWR has been implemented in the R package, spgwr (Bivand and Yu 2015).
Instead of trying to capture the autocorrelation in variable coefficients, this class of models argue that there is some degree of autocorrelation in either the error term, or in both the error term and ‘spatial lag’; a correlation in the outcome variable with other outcome values in its neighbourhood. The first case (correlation in error term between neighbours) is referred to here as a [LAGSAR] model, with a single parameter, \( \lambda \), that defines the extent to which neighbouring error terms correspond. For the case where there is also expected to be a spill-over effect in the outcome variable between neighbourhoods, the [SACSAR] model has an additional parameter, \( \rho \), that captures this effect, along with the correlation in residual parameter, \( \lambda \).

Neighbourhoods have to be predefined for this class of models, at least in their implementation in the spdep package in R ([Bivand and Piras, 2015]). A lot of work can go into defining neighbourhoods. The author attempted constructing special network-connected neighbourhoods for this work, but the payoff compared to a simple k-Nearest Neighbour (KNN) approach was negligible. Furthermore, as the purpose of the exercise was to make accurate predictions for the simulation model, and the machine learning methods very early on demonstrated themselves to be far superior to any linear regression approach, further effort in investigating complicated network neighbourhood structures for spatial autocorrelation models was not warranted. Consequently, KNN neighbourhoods were used in all estimations for the results presented in this.

Figure 4.12: Temporal autocorrelation at a stop, actual vs. simulated
Figure 4.13: Comparison of temporal autocorrelation distributions, simulated vs. actual, compiled from observations at all stop-to-stop combinations.

The values for the autocorrelation function were calculated on a stop-to-stop combination basis, for a lag of 1 (so each observation against its preceding observation) and a lag of 2 (each observation against the observation preceding the previous). Nearly all speed observations for all stop-to-stop combinations show negative autocorrelation when compared with the previous value. This appears to be due to the lognormal distribution of speeds, as the simulated speeds ($\mu = \text{RF prediction}, \sigma = \text{RF SD prediction}$) show nearly the same autocorrelation (blue), while autocorrelation is less pronounced if only the RF predicted values are used (red).

Increasing the neighbourhood size increases the time required for estimation, and at some point, will fail to capture any localised effects anymore, causing our estimation to look like the original [OLS] regression result. A reasonable value for the number of neighbours, $k$, was established by performing [SACSR] estimations for a selection of values of $k$, then comparing the [AIC] of these models and selecting the $k$ for which [AIC] was a minimum. This final neighbourhood size of $k = 6$ was then used throughout.
Figure 4.14: Maps of GWR, OLS and RF residuals.

Residual values have been truncated to -1 and 1.
<table>
<thead>
<tr>
<th>Variable</th>
<th>5th Percentile</th>
<th>Median</th>
<th>95th Percentile</th>
<th>OLS estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-0.1309460</td>
<td>-0.0351611</td>
<td>0.0387586</td>
<td>-0.0276203</td>
</tr>
<tr>
<td>intersection_density</td>
<td>-0.1855149</td>
<td>-0.0914125</td>
<td>-0.0455203</td>
<td>-0.1107565</td>
</tr>
<tr>
<td>left_turns_made_at_intersections</td>
<td>-0.1340092</td>
<td>-0.0607305</td>
<td>-0.0033048</td>
<td>-0.0567042</td>
</tr>
<tr>
<td>left_turns_passed_at_intersections</td>
<td>-0.1216626</td>
<td>-0.0771184</td>
<td>-0.0264728</td>
<td>-0.0729348</td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>-0.1219737</td>
<td>-0.0865292</td>
<td>-0.0531857</td>
<td>-0.0879356</td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.0522642</td>
<td>0.0865611</td>
<td>0.1853949</td>
<td>0.0928315</td>
</tr>
<tr>
<td>ln_path_length</td>
<td>0.3441541</td>
<td>0.4066697</td>
<td>0.4471002</td>
<td>0.3937816</td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.1559253</td>
<td>-0.1230659</td>
<td>-0.0797787</td>
<td>-0.1198247</td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate_in_radius_along_path</td>
<td>-0.2945161</td>
<td>-0.2160806</td>
<td>-0.1533300</td>
<td>-0.2081557</td>
</tr>
<tr>
<td>right_turns_made_at_intersections</td>
<td>-0.1063041</td>
<td>-0.0472849</td>
<td>0.0243066</td>
<td>-0.0333408</td>
</tr>
<tr>
<td>right_turns_passed_at_intersections</td>
<td>-0.1369389</td>
<td>-0.0636160</td>
<td>-0.0215069</td>
<td>-0.0674694</td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>-0.0562016</td>
<td>-0.0266513</td>
<td>0.0044600</td>
<td>-0.0328061</td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_path</td>
<td>-0.2497957</td>
<td>-0.2216725</td>
<td>-0.1763259</td>
<td>-0.2168003</td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>-0.1676314</td>
<td>-0.0546001</td>
<td>-0.0068158</td>
<td>-0.0681584</td>
</tr>
<tr>
<td>weighted_avg_capacity</td>
<td>-0.0955286</td>
<td>-0.0477410</td>
<td>-0.0099487</td>
<td>-0.0412958</td>
</tr>
<tr>
<td>weighted_avg_intersection_complexity</td>
<td>-0.1074329</td>
<td>-0.0473778</td>
<td>0.0036883</td>
<td>-0.0540961</td>
</tr>
</tbody>
</table>

Residual sum of squares: 2111.62
Effective degrees of freedom: 3641.317
Approx. residual standard error: $\sqrt{2111.6 \div 3641.3} = 0.7615$
Quasi-global R2: 0.4346

All coefficients express standard deviations change in dependent variable (\(\ln(speed)\)) for a standard deviation change in their own value. Ordinary least squares regression coefficient estimates included for comparison in the final column.
4.10.2 Results

4.10.2.1 GWR: coefficient summary

Of the three methods presented, GWR is perhaps the most interesting, as it tries to capture autocorrelation in terms of the variation in coefficient estimates across the island. This variation is summarised in the coefficient summary in Table 4.5, showing the range of each coefficient and comparing it with the OLS estimate. The neighbourhood size was chosen to be variable and optimised for the purposes of this estimation.

The estimation was performed on hourly averages, selected at each point for a random hour of the day between 6am and 11pm. The reason for not using all hourly averages, is that the R implementation only allows a single observation per coordinate. Arguably, one could perform repeated regressions with different hours of the day for each coordinate, and average the estimated coefficients at each location, to eliminate any possible bias resulting from using only a single random hourly average per coordinate.

The median values do not show significant deviation from the OLS estimates, which is equal to the mean of the coefficient estimates for the selection of hourly averages used in this estimation. For a number of variables, it appears as if their influence tends to be highly localised, as the 95th percentile of these variables show values to be close to zero, and even of opposite sign of the average coefficient estimate. These variables include the number of right turns made at intersections, the squeeze capacity, and the weighted average intersection complexity.

4.10.2.2 GWR: Spatial variation in coefficient estimates

Perhaps more interesting is to view the spatial variation in coefficient estimates. In Fig. 4.15 the colour of the points in the map are blue if the estimate is greater than the average value from OLS and red if less than the average value. The intensity of the colours indicate the degree of deviation from the mean value. There appears to be no clear consistency in the patterns for the different coefficient estimates. Spatial variation therefore appears not to be a consequence of the correlation between dependent variables. The only exception is arguably left_turns_made_at_intersections and transit_system_accumulation; however their patterns are only approximately similar.

For some of the variables, such as intersection_density, left_turns_made_at_intersections, right_turns_made_at_intersections, transit_system_accumulation, and weighted_average_intersection_complexity, a diagonal line running from the south-west to the north-east can be observed that bisects the island, with values above or below the mean coefficient estimate lying on either side. However, the transition doesn’t occur at exactly the same place.
The reasons for these variations are not immediately clear, and perhaps relate to geometric details of the network topology across the island. Street patterns in the southeast quadrant, for instance, tend to appear more regular and grid-like, while the west has more variety in its spatial organisation. Quantifying these patterns is outside the scope of the thesis. However, it appears that there might be value in analysing the network with, e.g. a space syntax approach, or other methods capable of quantifying geometric network qualities, an adding these as dependent variables to the model estimation.

4.10.2.3 LAGSAR and SACSAR

The results from this class of models are not that interesting in and of themselves. Instead, only by comparing the results with GWR and OLS does one get an idea of the relative influence of
For LAGSAR, SACSAR, OLS and GWR coefficient values are shown (Median values for GWR). For RF, variable importance measures are shown. The last two rows of the table compares the sum of squares and Moran's I for the residuals of each model. Moran's I calculated for a KNN network, k=6.

<table>
<thead>
<tr>
<th>Variable</th>
<th>LAGSAR</th>
<th>SACSAR</th>
<th>OLS</th>
<th>GWR Median</th>
<th>Random Forest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>-0.00466</td>
<td>-0.01502</td>
<td>-0.02762</td>
<td>-0.03516</td>
<td></td>
</tr>
<tr>
<td>fraction_of_path_excl_buslane</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>intersection_density</td>
<td>-0.06558</td>
<td>-0.09251</td>
<td>-0.11076</td>
<td>-0.09141</td>
<td></td>
</tr>
<tr>
<td>intersections_per_km</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>left_turns_made_at_intersection...</td>
<td>-0.05507</td>
<td>-0.05909</td>
<td>-0.05670</td>
<td>-0.06073</td>
<td></td>
</tr>
<tr>
<td>left_turns_passed_at_intersection...</td>
<td>-0.06999</td>
<td>-0.07395</td>
<td>-0.07294</td>
<td>-0.07712</td>
<td></td>
</tr>
<tr>
<td>ln_approx_activity_level_in_ratio</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ln_dwell_count_at_stop</td>
<td>-0.07757</td>
<td>-0.08221</td>
<td>-0.08794</td>
<td>-0.08653</td>
<td></td>
</tr>
<tr>
<td>ln_freespeed</td>
<td>0.08447</td>
<td>0.08974</td>
<td>0.09283</td>
<td>0.08656</td>
<td></td>
</tr>
<tr>
<td>ln_node_count_in_path</td>
<td>0.37967</td>
<td>0.39281</td>
<td>0.39378</td>
<td>0.40667</td>
<td></td>
</tr>
<tr>
<td>ln_path_length</td>
<td>0.11520</td>
<td>0.11274</td>
<td>0.11983</td>
<td>0.12307</td>
<td></td>
</tr>
<tr>
<td>ln_path_length_over_euclidean_distance</td>
<td>-0.03225</td>
<td>-0.03415</td>
<td>-0.03281</td>
<td>-0.02665</td>
<td></td>
</tr>
<tr>
<td>ln_RMS_radians_turned</td>
<td>-0.20975</td>
<td>-0.21395</td>
<td>-0.21680</td>
<td>-0.22167</td>
<td></td>
</tr>
<tr>
<td>ln_smartcard_transaction_rate...</td>
<td>-0.04101</td>
<td>-0.04470</td>
<td>-0.04130</td>
<td>-0.04774</td>
<td></td>
</tr>
<tr>
<td>minimum_capacity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>right_turns_made_at_intersection...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>right_turns_passed_at_intersection...</td>
<td>-0.06594</td>
<td>-0.06976</td>
<td>-0.06747</td>
<td>-0.06362</td>
<td></td>
</tr>
<tr>
<td>squeeze_capacity</td>
<td>-0.03225</td>
<td>-0.03415</td>
<td>-0.03281</td>
<td>-0.02665</td>
<td></td>
</tr>
<tr>
<td>traffic_controlled_nodes_in_pay...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transit_system_accumulation</td>
<td>-0.08408</td>
<td>-0.08166</td>
<td>-0.06816</td>
<td>-0.05460</td>
<td></td>
</tr>
<tr>
<td>weighted_avg_capacity</td>
<td>-0.04101</td>
<td>-0.04470</td>
<td>-0.04130</td>
<td>-0.04774</td>
<td></td>
</tr>
<tr>
<td>weighted_avg_intersection_comp...</td>
<td>-0.04805</td>
<td>-0.04614</td>
<td>-0.05410</td>
<td>-0.04738</td>
<td></td>
</tr>
<tr>
<td>weighted_avg_lane_count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \lambda )</td>
<td></td>
<td>0.15364</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \rho )</td>
<td>0.21917</td>
<td>0.10366</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Residual sum of squares:</td>
<td>2185.10000</td>
<td>2186.50000</td>
<td>2251.50000</td>
<td>2111.60000</td>
<td>392.2</td>
</tr>
<tr>
<td>Moran’s I (KNN,k=6):</td>
<td>0.09938</td>
<td>-0.00143</td>
<td>0.09194</td>
<td>0.07866</td>
<td>0.02618</td>
</tr>
</tbody>
</table>
spatial autocorrelation that cannot be captured by local variation in coefficient estimates alone.

For the sake of completeness, in Table 4.6 the results from the two model estimations are compared with not only the OLS and GWR median values, but also with the relative importance of values from the random forest estimation. At the bottom of the table there is a comparison of the residual sum of squares for all of the models. The Moran’s I-value for all of the models’ residuals was also calculated, using the same k-nearest neighbourhood, with $k = 6$.

What is interesting for these two models is that the SACSAR, while having more parameters than LAGSAR, has a marginally worse residual sum of squares value. However, there is almost no spatial autocorrelation detectable in its residual compared to the first. Both of the models performed significantly better than both the OLS and GWR models. The GWR model, while performing the best of the ordinary regression models, in terms of residual sum of squares, still has a significant degree of autocorrelation in its residual. Some simple investigations into running the GWR regression, followed by SACSAR, depletes the degree of autocorrelation in the residual completely while improving the residual sum of squares even further. However this is questionable practice, as it is very hard to separate the influence of the various components in such hierarchical regressions.

Clearly, from a prediction accuracy point of view, the random forest approach outperforms the linear regression approaches by far. Even though the RF regression does not contain any spatial variables, the degree of spatial autocorrelation in the residual is only 1/3 of that of the GWR.

Therefore, one can conclude that, at least to some extent, the random forest approach manages to capture interactions between the dependent and independent variables that would otherwise have been ascribed to autocorrelation in a linear regression approach.

4.11 Conclusion

In its current state, there are a number of shortcomings in the regression modelling that will be left to future work. Firstly, none of the regression models take account of the bus vehicle type, which one expects would dramatically influence how fast it can travel between stops, as different vehicle types have different acceleration profiles. While it is possible to associate a vehicle type with each bus line using the information from the web page stated earlier, a more interesting prospect would be to perform cluster analysis on the smart card data to automate the vehicle classification process.

Furthermore, it is not entirely clear that the regression models would be capable of adequately modelling changes in speed due to increased or decreased service frequencies. The only variable that takes account, to some extent, of service frequencies, is dwell_count_at_stop, which
simply counts the number of dwell operations taking place at the stop, derived from the trajectory reconstruction process. One might argue that, if service frequency is increased or decreased, this variable will have some influence on the predicted bus speed. However, judging from the magnitude of the variable’s coefficient in the OLS regression analyses, that influence might not be very big (assuming that we ignore the demonstrated multicollinearity problems in judging relative variable influence). In the RF analysis, it appears to have a stronger influence, however, the strength of that influence would have to be tested in a sensitivity analysis as it is not immediately apparent from the importance measures plots.

However, it is the author’s expectation, instead, that the influence of service frequency would not be reflected so much in the regression model alone, but that its effects would rather be captured in the simulation, as service frequencies affect the likelihood of bus bunching to occur.

The investigation into GWR shows that there are some systematic variations in the values of coefficients across the island that appear to bear some relation to the spatial organization and geometric qualities of the urban fabric. Supplementing the regression with e.g. a space syntax analysis might remove these variations and produce more accurate predictions.

Finally, from recent detailed analyses, it appears as if the MVO identification and removal pre-processing, preferentially targets observations of buses travelling on highways for removal. This is not surprising, in retrospect, as these buses probably tend to travel much faster than ordinary services. These observations will need to be marked and separately checked for MVOs. All of the regression models would have to be run again, with a dummy variable identifying those stop-to-stop combinations that are on highways, and its interactions with other variables will have to be tested.
Chapter 5

Transit simulation

Contents and figures from this chapter are originally from a book chapter on smart card driven transit planning (Fourie et al., 2016). The core contribution of the author was to modify the mobility simulation link dynamics module, previously created by Sergio Ordóñez, in order for it to take predicted values from the speed average and standard deviation models, for every transit stop-to-stop link, in 15 minute time bins across the whole day. Furthermore, the author ran the simulations presented in this chapter, performed all analyses presented here, and was the lead author in the book chapter publication that followed. Alexander Erath should be credited for the original idea of a data-driven transit simulation system, as well as the application scenario of splitting a long bus line in two.

This chapter covers components discussed in the system design chapter (Chapter 2) in more detail. The focus is specifically on the key elements of the transit simulation not yet covered in preceding chapters, and further focuses on topics of validation of simulation results, as well as the above-mentioned fictitious case study.

5.1 Key elements of the data-driven transit simulation

5.1.1 Demand generation

MATSim is an activity-based simulation framework, and its demand description is a timed sequence of activity locations and connecting trips for each agent in the study area. Generating an agent-based demand description from the smart card data is at first glance a straightforward task; each boarding and alighting location can be used as an activity location in an agent’s activity schedule. However, this would mean that each transfer in a transit trip would be identified as a significant activity, and the demand description would be over-specified by fixing transfer
location.

It is important that realistic transfer locations, and their associated walking and waiting times, rather emerge from the simulation than be specified in the demand description. If the purpose of the simulation is to test e.g. network reconfigurations, forcing the agents to visit their old transfer locations would be unrealistic. To this end one needs to identify the initial boarding and final alighting location of each multistage trip in the smart card data, and use these as approximate activity locations of the agents.

To assign transit users trip start times and identify individual multistage transit trips, the timing of transactions in the dataset is analysed on a per-user basis. First, when a user taps out and taps in again later, a threshold of 25 minutes was used to categorise those transactions as transfers or not transfers. If the time between alighting and boarding is more than 25 minutes, one assumes that the user has left the system, therefore they accumulate newly recorded access waiting time upon re-entry.

The second challenge is to assign trip start times to bus users, as the time that they get on the bus or train is not the time they arrived at the bus stop or train station. The reconstructed bus trajectories were analysed to extract headway times between consecutive services of the specified line. It was assumed that (i) users wait exclusively for services of the line that they boarded in the transaction, ignoring other lines that serve the same stops (ii) they don’t have external information on bus arrivals. This is not always true as users can be waiting for more than one candidate bus service serving their origin and destination stops. Such users would take the first service that arrives, or perhaps a service that allows them to travel seated rather than standing. They also can have more information about reliable bus arrivals from experience, or digital apps which estimate bus arrivals, allowing them to change their bus service of choice on-the-fly, or to minimise their waiting time at the bus stop.

Given these, admittedly flawed but expedient, assumptions, a uniformly distributed user arrival time was assigned to users at the bus stop within the corresponding headway. Thus, a MATSim activity plan for each CEPAS user was generated, and dummy activities were assigned at the initial boarding and final alighting locations of each trip, in order to make the demand well-formed for execution in MATSim.

5.1.2 Link dynamics

One idea that has remained central to the data-driven transit simulation is that the time taken to travel between transit stops be determined by a stochastic process, rather than emerge from the bus’s interactions with other traffic in a detailed network topology.

In a MATSim [QSim](#) a vehicle enters a link and is assigned a minimum travel time equal to the
free speed travel time of the link. When this time has passed, the vehicle is placed in a buffer at the end of the link. The vehicle has to wait in the buffer until all vehicles ahead of it have been cleared to pass into the next link in their routes. When the vehicle reaches the front of the buffer queue, the next link in its route is checked for available capacity, and the vehicle is only allowed to pass into the next link if there is both available capacity and the flow capacity of the current link is not violated. The process repeats for each time step until both these conditions are fulfilled.

Instead of assigning link travel time automatically using the free speed travel time, the MATSim API allows one to intervene and assign one’s own minimum link travel time. Initially, the data-driven transit simulation minimum link travel times were assigned by sampling from a normal distribution with mean and standard deviation estimated from our highly detailed, complete simulation of all Singaporean motorised traffic. Following the development of the speed regression models however, stop-to-stop travel times could now be sampled using a unique distribution for each stop-to-stop link by time of day.

As link speeds between most stops appear to be log-normally distributed, the simulation is loaded with predicted values from the instantaneous and standard deviation models of the natural logarithm of stop-to-stop speed, for each stop-to-stop link in the simplified network, calculated for every 15 minute interval during the course of the simulated day. When a bus enters a stop-to-stop link, the predicted values of average and standard deviation of $\ln(\text{speed})$ is retrieved for the time of entry, and a value is sampled from a normal distribution with these parameters. The final speed value in metres per second is taken to be the exponential function value of the sampled value, and the link length is divided by this speed to determine the minimum link travel time for the bus. When this time has passed, the bus is placed in the buffer at the end of the queue.

Clearly, it is possible for the usual MATSim dynamics of links behaving as first-in-first-out queues to be violated if, e.g. two buses entering at approximately the same time are assigned radically different travel times, allowing the second bus to reach the buffer at the end of the link before the first. However, once they enter the buffer, the usual MATSim dynamics prevail, and either bus is only evaluated once all other buses ahead of it in the buffer queue have been cleared.

Each stop-to-stop link connects to a short link containing the destination stop, where buses can queue up to perform their dwell operations. The bus would only be allowed to leave the stop-to-stop link and enter the bus queue link if there is enough space, in other words if the buses ahead of it have completed their dwell operations. Therefore, this interaction between buses still allows for simulated bus speeds to be markedly different from the values assigned to them by the stochastic process, and the simulation is capable of capturing the effects of bus bunching.
5.1.3 Dwell time model

Sun et al. (2014) analysed the CEPAS transaction data by vehicle and estimated regression models of boarding and alighting times, determined by the bus attributes (double/single-decker, articulated, step entrance/low floor entrance), as well as the occupancy. From these, they devised a combined model that determines the total dwell time, depending on the number of passengers boarding, alighting and on board the bus. Their findings were that boarding and alighting proceed at more-or-less constant rates if the bus occupancy is below a critical point. As illustrated in Fig. 2.3, if the bus has more passengers than the critical occupancy, boarding passengers have to wait for the excess passengers to alight to make boarding possible.

The model is stated as follows:

\[ Ac = \max \left\{ b(n_b - 1) + \max \left\{ n_o - n^*, 0 \right\} \times a, a(n_a - 1) \right\} \]  

(5.1)

where \( Ac \) represents total passenger activity time (the time taken by passengers to complete boarding and alighting), \( n_b, n_a, n_o \) refer to the number of passengers boarding, alighting and on board, while \( n^* \) is the critical occupancy of the bus type. The rest of the variables in the model are coefficients estimated from the smart card data.

The model was implemented in Java using the MATSim API which allows one to explicitly control passenger boarding and alighting time, and keep track of vehicle boardings and alightings through associated simulation events.

5.2 Validation and performance

The modified MATSim simulation model was run for 50 iterations, and various measures of system performance were compared against the original smart-card data and the trajectory reconstruction-related data.

Fig. 5.1 compares the distribution of bus speeds between stops in the simulation against the speeds that were derived from the trajectory reconstruction process. Both the shape of the distributions and absolute numbers correspond very well.

5.2.1 Headways, dwell times and bus bunching

Fig. 5.2 shows the distribution of headways in the simulation versus those derived from the trajectory reconstruction process. In its current state the simulation appears produce too many short headways; this is due to somewhat excessive bus bunching that occurs during the simula-
Figure 5.1: Comparison of bus speed between stops from the simulation against those from reconstructed trajectories.

Figure 5.2: Comparison of bus headway distributions from the simulation against those from reconstructed trajectories.

Fig. 5.3 shows the joint distribution of headway versus number of stops along the route. This figure shows that, both in the reconstructed trajectories and the simulation, headway variability increases with increasing number of stops along the route, however the effect is much more pronounced in the simulation. Specifically, it shows that the simulation produces many headways in the 0-1 minute bin, which indicates bus bunching. This behaviour in the simulation is probably largely due to the fact that the first-in-first out queue dynamics prevent buses of the same service from passing each other in the bus queue links at transit stops. Passing behaviour should be implemented in future work, as buses of the same service can pass each other in reality when the first bus is already engaged in a dwell operation at a stop. However, this in-
tervention would first require a detailed investigation into the prevalence of passing behaviour under various service conditions.

As the trajectory reconstruction process does not extrapolate the trajectories beyond the last recorded transaction for a circuit run, headways for the stops towards the end of a route might be inaccurate, which accounts for the lighter shading of the joint distribution of headway versus stop number in the smart card data. However, the distribution of the headways does appear considerably narrower for the smart card data than what the simulation produces. It is not clear if any bus bunching control measures were in operation for the buses, whether it be centralised control from the operations center, or by intelligent actions of the bus drivers themselves. Such measures would naturally account for the increased reliability of services. However, it would also be worth investigating if allowing buses of the same service to pass each other when one bus is already occupied at a bus stop, serves as a bunching control measure in itself.

In Fig. 5.4, the dwell time of buses in the simulation is compared against those derived from the trajectory reconstruction process. In terms of absolute numbers, nearly a million dwell operations with zero length occur in the simulation; these are cases where no boarding or alighting transactions take place. In the trajectory reconstruction, dwell operations that have been interpolated are assigned a zero dwell time. Dwell operations where only a very small number of transactions are recorded within a time span of less than six seconds, are assigned an arbitrary minimum dwell time of that value, which is responsible for the second spike in dwell times that observable in the histogram. In terms of absolute numbers, the sum of these
trivial cases for the smart-card data corresponds reasonably well with the number of dwell operations in the simulation where no passengers board or alight.

Because the absolute number of dwell operations for the nontrivial cases are different for the simulation and the smart-card data, a fairer comparison might be the density plots in the second part of the figure, which reveals reasonably good correspondence between the simulation and the dwell times from the trajectory reconstruction process.

The trajectory reconstruction process produces 1.58 million dwell operations compared with the 1.7 million dwell operations recorded in the simulation; the difference between these numbers is due to the fact that the trajectory reconstruction process does not extrapolate the trajectories of buses beyond the first and last recorded transactions. So, if the first recorded transaction for a bus occurs at a stop after the first in its route profile, or the last recorded transaction is before the end of the line, then no dwell operations are created for the stops before the first transaction, or after the last transaction.

Fig. 5.4 also shows that, of the 1.7 million dwell operations in the simulation, nearly one million have a zero duration, meaning that no passengers were picked up or dropped off. This means that buses in the simulation only pick up or drop off passengers approximately 40% of the time. Consequently, the simulation also produces more dwell operations of longer duration, as fewer dwell operations have to serve the same number of passengers. This might be a contributing factor to the higher incidence of bus bunching observed in the simulation.

If one assumes that the actual total number of dwell operations also comes to 1.7 million, then the number of cases where buses don’t take on any passengers at stops for that particular day in the actual transport system comes to approximately 580,000, which accounts for approximately
34% of all dwell operations, meaning that buses in reality pick up or drop off passengers 66% of the time, in comparison to the 40% observed in the simulation. This difference might be due to the best response routing in the simulation resulting in increased coordination between agents and buses, with agents selecting services that get them to their destination with less access waiting time on average than the service that they picked in reality. Agents might also not be as averse to crowding as people in reality, causing them to opt for the next empty vehicle less frequently; a hypothesis that will require further investigation into the ridership of vehicles in the simulation versus those in reality.

The space-time diagram shown in Fig. 5.5 compares the trajectory reconstruction results against the simulation for a bus line with 74 stops along its route. While the shapes of the trajectories compare reasonably well, it is apparent that the simulation produces more bus bunching than what this bus line experienced in reality, confirming what was presented in the histogram in Fig. 5.3.

### 5.2.2 Passenger travel time measures

Fig. 5.6 compares the trip travel time from the simulation with that of the smart-card data, where access and egress walking and waiting times have been excluded from the times recorded in the simulation. The histogram therefore compares only the sum of in-vehicle travel times, and transfer walking and waiting times. It appears that the combination of a realistic schedule of departures, with the modified simulation dynamics, ensures that trip travel times compare well with actual values.

Fig. 5.7 similarly shows very good agreement between the bus stage in-vehicle times for the simulated versus actual values, although smart card values appear slightly skewed to longer times. While the simulated speeds are stochastic, in order to display the same range of values as those observed in reality, it is possible that not all dynamic effects have been captured adequately for perfect agreement, or that agents are routed more optimally than passengers are in reality. As it is not known when passengers board or alight from trains, a similar graph for rail modes could not be constructed. However, the good agreement for trip travel time across all modes inspires confidence that the simulation of the rail mode is reasonably accurate, as passengers would have switched away or switched to using the subway during the simulation if this transport mode performed markedly different from reality.

Fig. 5.8 compares the density of transfer times in the simulation against that which was derived from the smart-card data. This plot only displays non-inter-rail transfers for both the simulation and the smart card data, as the actual transfer location in the rail system cannot be directly ascertained from the CEPAS data. The number of non-inter-rail transfers, for the 25% sample in the simulation, comes to 418,230, which inflates to 1,672,920 for a full sample, compared
Figure 5.5: Comparison of space-time graphs of CEPAS versus simulation for a long bus line in Singapore.

with the 1,410,694 derived from the 25 minute transfer time rule in the CEPAS data.

The simulation produces significantly more non-inter-rail transfers, and there are significantly more transfers in the 0-1 and 1-2 minute bins than what is found in reality. This excess of transfers, and very short transfer times, is possibly due to the coordination that occurs between demand and supply as a result of the best response re-routing during the simulation. Once a reliable inference of the number of inter-rail transfers has been performed on the CEPAS data, it would be interesting to see if the phenomenon persists in the rail system as well.
Figure 5.6: Comparison of simulated versus actual trip travel times across all modes of transit (excluding access and egress times).

Figure 5.7: Comparison of simulated versus actual bus stage-in vehicle time.

5.2.3 Computation times of simplified simulation

Using only best response re-routing, the simulation reaches a relaxed state in very few iterations. After only five iterations very little change in the average score of agent plans can be observed with increasing iterations. From repeat experiments it was found that a 25% sample of all agents is required to get realistic results; all counts at a passenger level recorded in the validation section have therefore been scaled up by multiplying them by four.

Experiments were run on a latest generation 24 core Intel Xeon computer, with 64 GB of RAM. The initial routing of all agent plans takes approximately seven minutes, while a single iteration takes approximately four minutes. It is therefore possible to have usable results in under an hour. In the case of a standard MATSim simulation of Singapore (Erath, 2012), where we
simulated both public and private transport, many more iterations are required for the system to reach a relaxed state, and a full simulation takes up to two days to complete. The simplified simulation therefore represents a big step forward in terms of computation time performance of a large-scale multi-agent transport simulation.

5.3 Application

To show the potential of the simplified transit simulation, a fictitious study case was designed. The proposed scenario splits one of the longest bus lines in Singapore, which has more than 90 stops. The line was split according to the method used in [Lee et al. (2012)] in order to minimise the number of transfers resulting from the split; in this case the optimal split point happens to be close to the center of the route. Agents are prepared to re-route their transit routes within the MATSim co-evolutionary algorithm until they reach equilibrium (100 iterations). That means the agents who were taking the long line or any other line in Singapore can decide to take the new split line or switch to another transit line. As in the case of the validation study, a 25% sample of the population was simulated, with vehicle carrying capacities reduced to a quarter of their real-world values. The following section compares the performance of the line split against the baseline case.
5.3.1 Impact on bus bunching

Fig. 5.9 shows the space-time diagram during the morning peak of the bus service before and after the split, with cases of bus-bunching highlighted in red, and line thickness increasing with bus ridership. The plot confirms that incidences of bus bunching is significantly reduced during the morning peak hour, and that headway reliability is improved considerably, especially towards the end of the bus route. Note that departure times were replicated from the start of the service for buses departing on the second part of the line split, which means that these services start with an inherent lack of reliability. Furthermore, even though the number of stops are now only half that of the original in the two resulting routes, the simulation suggests that bus bunching can result relatively early and that 45+ stops might still be too many bus stops for a reliable bus service. However, this result has to be interpreted in the light of the earlier findings that the simulation tends to result in somewhat more bus bunching than what is observed in reality.

5.3.2 Excess waiting times

Excess Waiting Time (EWT) is one of the most common reliability indicators for high frequency transit services (e.g. a service frequency of five or more buses per hour). From the definitions used by the London transport authorities, [EWT] assessment includes calculation of
Figure 5.10: Comparison of the excess waiting time before and after a long bus line has been split into two separate routes

the following two elements, restated here for convenience:

**Average Scheduled Waiting Time (SWT):** the time passengers would wait, on average, if the service ran exactly as schedule, assuming that waiting time is, on average, half of headway time:

\[
SWT = \frac{\sum_{s \in S} H_s^2}{2 \sum_{s \in S} H_s}
\]  

(5.2)

**Average Actual Waiting Time (AWT):** the average time that passengers actually waited:

\[
SWT = \frac{\sum_{s \in S} H_a^2}{2 \sum_{s \in S} H_a}
\]  

(5.3)

Where \( s \) represents each service of a bus line (excluding the first one), \( H_s \) is the scheduled headway of the service and the previous service, and \( H_a \) is the actual headway of the service and the previous service. [EWT] is simply the difference between [AWT] and [SWT] and represents the additional waiting time experienced by passengers.

The formulas have this form because [AWT] and [SWT] are weighted averages of all the service headways of a line, and the weight is the actual headway. So, if the line is designed to have a constant headway, the calculation of [SWT] can be simplified to \( SWT = 0.5H_s \).
Fig. 5.10 compares the calculation of the EWT of the base case against the split line scenario, in one direction of travel. The plot looks very similar in the opposite direction; EWT reverts to zero at the point with the line split, and consequently passengers experience much better reliability towards the end of the route.

5.4 Conclusion and future work

From the section on validation, our results so far appear to agree well for most part with actual observation. Most importantly, the simplified simulation manages to capture dynamic bus bunching effects; in fact, the effect might be slightly exaggerated in the simulation, and the possibility of mitigating this effect through the implementation of passing behaviour in the QSim will be investigated. The simple fictitious case study also illustrates that the simplified simulation can be used to evaluate proposed changes to the transit system.

The reconstruction of train trajectories is a very interesting problem as inter-rail transfers are not explicit in the CEPAS data. Furthermore, transit passengers need to be allocated to buildings that are close to transit stops, in order to better simulate access walking and waiting times.

Our subway stops are also easily accessible in the simulation, and do not take account of the time that it takes for passengers to travel all the way down to station entrances. Consequently, there is a slightly increased preference for the rail modes in the simulation compared to reality.
Chapter 6

Improving simulation performance

6.1 Development context

The contents of this chapter was largely taken from a journal paper written jointly with Kai Nagel and Johannes Illenberger from the Technical University of Berlin (Fourie et al., 2013). First and foremost, therefore, the author acknowledges the contributions of these two colleagues.

The basic functionality of producing an event stream to rapidly evaluate plan performance of car travellers was originally developed by Johannes Illenberger in order to evaluate cooperative location choice in social networks (Illenberger, 2012). He designed the original Pseudo-Simulation (PSim) class to produce events for private vehicle free speed travel time, by reading the relevant values (link length divided by free speed) from the network description. Professor Nagel provided valuable feedback and guidance, and edited many drafts of both the conference and journal versions of the paper.

The author’s key innovation presented in this chapter, is based on the insight that information generated during Queue Simulations (QSim) can be further exploited in the PSim surrogate model. This requires information to be recorded during the QSim which can then be read by the PSim in order to more rapidly produce an event stream that approximately describes the agent’s trajectory if its plan were to be executed in the QSim. It was hypothesised that fewer QSim iterations would be required if more mutation and selection could take place between QSim iterations.

Furthermore, the original PSim, while relying on the MATSim API, was purpose-built for the evaluation of plans generated to simulate agents participating and coordinating travel within social networks. It was the author’s vision that, instead, the PSim should be incorporated into the MATSim framework, as a general modification to speed up the execution of any given MATSim
scenario. For this reason, the PSim module has been compiled into a MATSim contribution and is documented in the MATSim book (Fourie, 2016a).

### 6.1.1 Transit

The results presented in this chapter are mostly from the initial implementation of the PSim into the MATSim framework, from the time of the publication of the aforementioned journal paper. At that stage, the PSim code was capable of mimicking the simulation output of a scenario where only private vehicles are explicitly simulated on the network, and where transit instead is ‘teleported’ from origin to destination. In the meanwhile, the author has adapted the code to also realistically mimic the output produced when simulating transit on the road network.

This capability required new data structures capable of recording the time taken to travel between transit stops, as well as the waiting times recorded at stop locations during a QSim. These data structures had already been created for the best response transit router developed by Ordóñez Medina and Erath (2013). The PSim code, in its current state, can be applied to most MATSim scenarios that simulate both public and private transportation on the road network, while teleporting all other modes of transportation.

Because the smart card driven transit simulation, that forms the topic of this thesis, is still based on the MATSim framework and relies on (albeit modified) QSim of transit on a network, the methodology presented in this chapter can be easily applied in order to reduce the number of expensive QSim required to run the simulation. However, in its current guise, the smart-card driven transit simulation only performs re-routing during the replanning stage of the simulation. It uses the best response routing algorithm developed by Ordóñez, and therefore there is not much to be gained from running a surrogate model in between QSim iterations to accelerate relaxation. Following any given QSim, repeated replanning will always produce the same set of plans, as the routing algorithm is completely deterministic. A stochastic router, in comparison, would gain from the PSim approach, as each iteration in the simplified simulation would produce a different travel time result, and poorly performing routes would be discarded.

However, in future work, agents will be allowed to perform some time allocation improvements, in order to respond more flexibly to proposed interventions in the transit system, such as crowd avoidance. Furthermore, alternative last mile solutions are becoming increasingly accessible to transit users. These include electric scooters, foldable bicycles, and shared bicycles at stations. The data-driven transit simulation system should be modified in order to deal with these more complicated scenarios, which necessarily will dramatically increase the size of the solution space, and require longer simulation times and more iterations to be run for simulations to reach a relaxed state. For such more complicated scenarios, the capabilities presented in this chapter, and Chapter 7, will once again be of great value in reducing simulation times.
6.2 Motivation

As was shown earlier in Fig. 2.2, MATSim iteratively simulates the interaction between supply and demand, by executing the day plans of agents in a mobility simulation. After each simulation, the executed plans are scored, rewarding agents for time spent performing activities, and penalising them for travelling or arriving late at activities.

Following the scoring step, a number of plans are selected for mutation across various choice dimensions, which can include activity timing, mode choice, or the path taken through the transportation system.

Agent plans evolve in response to transport system performance over generations, as poorly performing plans get discarded, while plans with good scores serve as the basis for further mutation and improvement. The overall process is one of co-evolution (Goldberg 1989; Russel and Norvig 2010) as any given agent’s plans are evolving in the presence of all other agent plans doing the same.

Mobility simulations are time-consuming, as the interactions of all agents participating in the transportation network are executed for every second in a 24-hour simulated day. Plan mutators are comparatively fast (if mutation is simple and random), even when mutation occurs across many dimensions.

As the number of choice dimensions in the scenario increases, the number of iterations and thus the number of mobility simulation runs required to explore the solution space increases. Furthermore, the impact of random changes to day plans on the rate of convergence rapidly diminishes with increasing iterations; therefore a lot of time gets spent on costly mobility simulation with diminishing returns in terms of the rate of system evolution.

Fig. 6.1 shows how the scores of executed plans increase with increasing number of mutations. These results are for a simple experiment on the Sioux Falls test scenario, created by Chakirov and Fourie (2014) and includes transit and private car modes. Plans were randomly mutated for their activity timing, route choice and mode choice. With purely random mutation, convergence proceeds very slowly, and even after 300 iterations none of the plans show a positive score. However, it is clear that increasing the number of mutations improves the score of most executed plans.

6.2.1 Harnessing parallelisation

Recently, advances in computational power have come from parallelisation rather than increasing clock speeds. Consequently, studies focusing on improving agent-based transport simu-
Scores on the right are for the same ordering of plans on the left, so any coordinate in either plot refers to the same executed plan.

6.2.2 The expanding scope of MATSim applications

There is a growing need to integrate existing and emerging model capabilities such as within-household interaction and coordination (Bradley and Vovsha 2005; Meister et al. 2005; Miller and Roorda 2003), ride-sharing (Ciari et al. 2012), social network interaction (Carrasco Mon-tagna 2006; Illenberger 2012), complex mode-chaining, dynamic multi-modal pricing (Tirachini and Hensher 2012), transit, secondary activity location selection (Arentze and Timmermans 2007; Horni et al. 2012a), spatially distributed parking capacity (Waraich and Axhausen 2011), and multi-day, need-based activity modelling (Märki et al. 2011); to produce a truly integrated activity-based transport model. As the agent choice dimensions and constraints increase, the model solution space explodes in size and complexity. Consequently, the MATSim
solution process is expected to require a dramatic increase in the number of iterations in order to effectively explore the high-dimensional solution space. The efficiency of the solution process therefore needs improvement, while retaining the flexibility designed into the current framework.

In this chapter, PSim is introduced as a flexible meta-model to the MATSim framework in order to increase the rate of system evolution. Multi-modelling techniques are frequently used in simulation-based optimisation, where a simplified model of the system is estimated based on a sample of simulated observations. The simplified surrogate- or meta-model ideally takes a deterministic form that is computationally cheap to evaluate.\footnote{See Queipo et al. (2005) for a comprehensive review of surrogate-based techniques.}

In our application, the multi-model system periodically replaces the current QSim for a number of iterations with a simplified Pseudo-Simulation (PSim) that runs approximately two orders of magnitude faster. PSim uses information generated in the preceding QSim iteration to produce an estimate of how well an agent day plan might perform, which allows the existing model framework to select and improve plans before executing them in a full QSim.

Consequently, this modified simulation requires fewer QSim iterations, by making better use of the information generated during the full mobility simulation.

### 6.3 Background

#### 6.3.1 Mutation approaches

In Fig. 2.2, the “replan” action represents the mutations producing evolutionary change. Replanning is done through the chaining of modules into strategies. An example strategy might be:

Draw 10% of agents, [randomly select a previously executed plan from memory for each agent and make a copy of it], [adjust the start time and duration for each activity in the plan by a random number of seconds less than half an hour], [find the quickest network route between activities based on travel times from the previous iteration], mark these plans as ready for execution.

For all remaining agents, [select a previously executed plan from memory based on plan score], mark these plans as ready for execution.

In this example, each set of brackets denotes a replanning module. Some modules are merely plan selectors, and do not mutate plans. Other modules can be divided into random-response
and *best-response* mutators. For the strategy set out above, the start time and duration adjustment module is random-response, while the router is a best-response replanning module, using a Dijkstra algorithm to find the lowest cost route through the network at a given time of day.

### 6.3.2 Best-response vs. random-response replanning

Best-response modules, though computationally burdensome, reduce total simulation time by exploiting traffic information from the previous iteration, to produce a near-optimal solution to the mutation they are suppose to effect. In the example above, the Dijkstra router produces a truly optimal shortest path for each set of origin and destination points in the agent’s plan.

In contrast, random-response modules rely on the trial-and-error of the evolutionary algorithm to produce better plans across many iterations, and do not guarantee any improvement in plan fitness.

More complex best-response modules have been developed that explore multiple dimensions of the agent decision space, in order to dramatically reduce the number of iterations until convergence (e.g. [Meister et al., 2006; Horni et al., 2012a; Dubernet and Axhausen, 2012]). In general, these modules apply a meta-heuristic approach to explore the solution space, ideally evaluating solution quality using the same utility-based scoring function selected for the scoring step of the MATSim framework. Such monolithic replanning modules have a number of disadvantages. Firstly, they are purpose-built; if a scenario element is not included in the module, its influence is not considered in the solution. For instance, suppose modx, a time-and-mode optimizing module, consistently finds that the best departure time for an agent is 7 am, by car, just when the congestion pricing starts on the highway connecting that agent to work. If modx does not consider road-pricing in its design, the resulting plan will be sub-optimal, as the router will, say, find a lower-cost but slower route to work for the given departure time. A more favorable possible alternative, e.g. departing earlier to avoid the road pricing, is unlikely to be found, as modx optimizes one sub-problem and the router another.

As the feature set of MATSim grows with time, these modules therefore become obsolete, and require significant re-design to remain relevant. However, due to their design complexity, best-response replanning modules are harder to maintain and integrate with new functionalities than simple random-response modules.

### 6.3.3 Simulation-based optimisation using surrogate models

Multi-modelling approaches have been applied in a number of transport modelling contexts. In MATSim, the monolithic replanning modules argued against earlier represent meta-models
of the simulation; simplified representations that are used to perform fast, cheap evaluation of agent plans.

Meta-modelling is very useful in Intelligent Traffic Systems (ITS) applications, that require fast turnaround times. For instance, Osorio and Bierlaire (2009) combine the output from an AIMSUN dynamic traffic micro-simulator with a surrogate model that analytically captures stationary queue distributions. They use this approach to perform simulation-based optimisation of signalling plans in a congested network (Osorio and Bierlaire, 2008).

Meta-models are also used extensively in simulation model calibration, as model parameters need to be adjusted in order for the simulation to better reproduce reality. In Osorio et al. (2015), the authors embed an analytical problem-specific description of how the calibration parameters are related to the simulation-based objective function within their calibration algorithm.

Surrogate-based methods are also useful in performing sensitivity analyses or simulation-based optimisation without the need to run expensive simulations. Generally, a few experiments are run and simulation performance measures are recorded against the parameter values selected. Then a response surface is fitted that attempts to predict simulation measures (e.g. kilometres travelled, average travel time, mode share) as a function of simulation parameters, e.g. travel costs. For instance, in Chen et al. (2014), a road pricing strategy is optimised using a Kriging meta-model fit against results of a 32 experiment latin hypercube design.

The PSim approach differs from these methods in that the meta-model operates at the level of the individual agent, rather than trying to predict the system output at aggregate level. It also retains the feedback and learning effects of the MATSim framework, that generally do not form part of typical surrogate-based methods.

6.3.4 Feedback and learning

The idea of predicting the outcome of actions through learning and feedback between the mental and physical domains is not new to transport simulation (Arentze and Timmermans, 2001; Rieser et al., 2007). A multi-level feedback loop, using transport system metrics on one level to inform the location decisions of households and firms, and individual learning on the other as agents respond to resulting changes in demand patterns, has also been the subject of recent investigation (Nicolai et al., 2011). Also, UrbanSim (Waddell et al., 2003) can use so-called ‘skims’ which means to use a previous output of the assignment model in order to avoid running it – this implies the assumption that travel speeds in the transport system remain the same over a couple of UrbanSim iterations.
Figure 6.2: Schematic illustration of the modified MATSim simulation process with pseudo-simulation.

6.4 Design

Figure 6.2 illustrates the principle behind the multi-model approach. The system is fed with an initial demand of agent plans, which get executed in QSim. Plans are scored and sent to the replanning modules. An inner loop is then executed for a number of iterations, where new plans are executed in the Pseudo-Simulation (PSim), scored, and sent for replanning. After, say, \( p \) such iterations, plans are selected again for execution in QSim, scored, and the inner loop repeats again for another \( p \) iterations. The outer loop repeats \( q \) times, then terminates with a final QSim and scoring step, leaving a relaxed demand.

6.4.1 MATSim events

In MATSim, QSim writes out time-stamped, atomic units of information called events, which describe what is happening to each agent at all times. Trawling through these events, it is possible to reconstruct every agent’s trajectory through the transportation system, and the time they spent at various activity locations.

Consider, for example, an agent traveling from home to work in a small network. Her event stream might appear as in Fig. 6.3.

The XML code shows the simulation time in seconds for each event. This agent (with ID=1), therefore ends activity “home” at six in the morning, departs by car (vehicle ID=1), then enters and leaves a number of links in the network to arrive at work at 06:20:46. The agent departs from work at the scheduled time of 5pm, as specified in her day activity plan, and continues home. Each link traversed is identified uniquely by a link ID.
**Figure 6.3:** Example MATSim XML event stream for a single agent.

```xml
<event time="21600.0" type="actend" person="1" link="1" actType="home" />
<event time="21600.0" type="departure" person="1" link="1" legMode="car" />
<event time="21609.0" type="wait2link" person="1" link="1" vehicle="1" />
<event time="21610.0" type="left link" person="1" link="1" vehicle="1" />
<event time="21610.0" type="entered link" person="1" link="6" vehicle="1" />
<event time="22057.0" type="left link" person="1" link="6" vehicle="1" />
<event time="22057.0" type="entered link" person="1" link="15" vehicle="1" />
<event time="22487.0" type="left link" person="1" link="15" vehicle="1" />
<event time="22487.0" type="entered link" person="1" link="20" vehicle="1" />
<event time="22846.0" type="actstart" person="1" link="20" legMode="car" />
<event time="22846.0" type="actend" person="1" link="20" actType="work" />
<event time="61200.0" type="actend" person="1" link="20" actType="work" />
<event time="61200.0" type="departure" person="1" link="20" legMode="car" />
<event time="61200.0" type="wait2link" person="1" link="20" vehicle="1" />
```

In QSim, the time taken to traverse a link is generated by the QSim dynamics (see Dobler and Axhausen [2011]), and is therefore a stochastic, emergent property of the simulation. For transit, the time at which an agent boards or alights from a bus or train depends on the trajectory of the vehicle through the network and the congestion and previous dwell operations it had completed.

In order for PSim to appear as a realistic replacement for QSim, it needs to produce an event stream that looks similar to the original, in order for the scoring function to produce a score for each executed plan that would be similar to the score it would receive from being run in the QSim. Events therefore need to be generated in the exact same chronological order, with the same sequence of links being traversed at the correct times. For transit, boarding and alighting needs to take place at appropriate times and at the correct transit stops for the transit line specified agents plan.

### 6.4.2 PSim operation

From the QSim event stream, one can deduce the travel time for each agent on each link during the course of the simulated day. A simple way of doing this is to slice the simulated day up into arbitrary time intervals, say 15 minutes each, calculate the average travel time for each link during every interval, and store these values in a lookup table.

Similarly, for transit one may record the waiting time that agents spend at the bus stop, as well as the time taken to travel between transit stops. Average values for every 15 minute interval during the course of the day can then be calculated. These values can be stored in a lookup table, that records the bus stop identification number and its associated waiting times during the course of the day. For transit travel times, the lookup table would store average travel times...
between stops by time of day, for each combination of origin/destination stop IDs.

Suppose a replanning module now produces a new plan for the agent above, where she leaves home a little later, or takes a different route to work. The PSim module constructs an event stream that represents her expected experience in the transport system, by reading the appropriate times from the lookup table for each link in her route, at each relevant time interval. It passes this event stream to the scoring module, which now produces an expected score for the new plan, and keeps the scored plan in the agent’s memory. After repeating the process a number of times, we reach the agent’s memory limit, and the poorest performing plan is discarded at the end of each iteration.

The agent is now learning not from the full stochastic QSim but a simplified representation of it; consequently PSim is a surrogate model for QSim. After a number of iterations, the agents are passed back to QSim to evaluate actual plan performance and produce an updated lookup table of travel and waiting times, and the process repeats.

No physical interaction occurs between agents in PSim, so it can fully exploit modern multi-core computer architectures, as no synchronisation between threads is required and access to data structures outside a PSim thread is read-only. Load balancing is simple; plans scheduled for execution are simply divided up between threads. Event processing is also completely parallelised, as are re-planning operations.

QSim always requires the full set of agent plans, as travel times emerge from their interaction. As there is no interaction between agents in PSim, it makes sense to only simulate newly generated plans, that do not have a score associated with them yet. This cuts down on the expected computational load even further, as each iteration only generates a small number of new plans, depending on the rate of replanning prescribed by the replanning strategy.

### 6.5 Experimental setup

The multi-model approach was tested for compatibility, computational performance and solution quality by comparing its results for a large-scale simulation scenario against those produced by a baseline simulation run, that uses the default, QSim-only approach. The goal of this setup is to test if performance gains from the multi-model approach have any implication on the solution state compared to the standard approach.
6.5.1 Simulation scenario

The MATSim development scenario of Swiss car traffic crossing or operating within a 30km radius circle around Bellevue, Zurich, as used in the secondary activity location choice study of [Horni et al. (2012b)]. was used in all experiments in the results section of this chapter. The scenario, originally developed by [Balmer et al. (2008)], and updated and further documented in [Balmer et al. (2010); Horni et al. (2011)] is regularly used as a benchmark in MATSim investigations.

The same 10% sample from [Horni et al. (2012b)] study was used in this chapter, as well as the same network representation and facility information. The scenario contains 67,239 agents traveling in a network of 60,518 links, and a total of 1,697,196 activity facilities. An arbitrary morning toll was introduced on all links exceeding a capacity of 4,000 vehicles per hour.

The following re-planning modules were used in equal measure, with the total replanning rate (proportion of agents replanned) varied as part of the experimental setup:

1. activity start time and duration adjustment;
2. re-routing using travel times from the previous iteration;
3. sub-tour mode choice – switches the mode of transport of a randomly selected sub-tour to car/transit given that, for this scenario, all agents have access to cars;
4. secondary activity location choice: shopping and leisure activities are switched to a randomly chosen location from a set of qualifying facilities.

Transit is not explicitly simulated, as this capability would require a full transit schedule of vehicle departure times, and a full set of transit lines and routes. Instead, trips using transit are ‘teleported’ during the simulation from origin to destination with a travel time that is twice that of the free speed shortest path through the network ([Rieser et al. 2009]).

6.6 Results

6.6.1 Characterizing solution state

MATSim employs stochasticity at various points in a simulation run, such as agent selection for different modes of replanning, plan selection for execution, and transition rules at intersections during a QSim. In order to make runs repeatable, a seed number is set for the Java random number generator at the beginning of a simulation run.
In the experiments presented here, the same random seed was used for all simulation runs, except a baseline QSim-only run. Then, when comparing the solutions of two QSim-runs with the same parameters except random seed, one has an indication of the minimum deviation one may expect between any two runs of the same scenario.

The baseline against which simulation runs were compared was selected as the simulation state obtained by running the scenario for 101 iterations with QSim only, at an overall replanning rate of 30% per iteration, with a maximum agent memory of 5 plans per agent.

Five measures were used to characterize solution state for comparison against the baseline:

6.6.1.1 Average executed QSim score

The 101st iteration score of 175.4 for the baseline run is taken as a reference value. For all other runs, the first QSim iteration where the score was greater or equal to this value was selected and the rest of the measures were calculated.

6.6.1.2 Departure profile RMSD

Agent departures are compared at 5 minute intervals for the simulated day. The root mean square deviation (RMSD) from the baseline departures is taken as an indication of how similar a simulation state is to the baseline in terms of activity timing.

6.6.1.3 Mode share

Car mode share (number of car trips / total number of trips) is compared for the large-scale scenario, as mode choice is one of the dimensions included in the replanning strategy.

6.6.1.4 Daily link volume RMSD

The daily volume of car traffic traversing every link in the network is compared against the volumes produced by the baseline run. The root mean square deviation (RMSD) from the baseline link volumes is used as an indication of how similar a simulation state is to the baseline in terms of car traffic volumes.
6.6.1.5 Agent total travel time difference

The event stream is processed to compare the total travel time experienced by each agent in comparison with those produced by the baseline run. The difference is compared for each agent between the two runs, and count the percentage of agents that experienced a difference below five minutes and one minute, respectively.

The reference value for each measure refers to the value produced by the reference case; i.e. the QSim-only run where only the random number seed differs from the baseline setup.

6.6.2 Varying QSim:PSim ratio

When keeping the replanning rate constant, it was found that increasing the number of PSim iterations between QSim iterations increases the rate of convergence, as can be seen from Fig. 6.4. This figure compares the utility vs. number of QSim iterations for two QSim:PSim ratios (red) against the reference case (black).

In general, for a given intermediate utility score, the number of QSim iterations required to achieve that score is approximately inversely proportional to the total number of iterations executed during the simulation, e.g. QSim + PSim iterations.

6.6.3 Performance test

Fig. 6.5 compares the influence of QSim:PSim ratio, number of computational cores and replanning rate on simulation (wall clock) time. Here it is clear that the multi-model strategy is only effective as the number of cores committed to the simulation is increased.

Fig. 6.6 shows the wall clock time it takes, with different set-ups, to reach a certain level of convergence, as described earlier. One notices that the computing (= wall clock) time for replanning scales inversely linear in the number of cores. That is, with an ever growing number of cores, that number will shrink more and more. This is due to the computational (and conceptual) decoupling of the replanning: every agent replans for herself. Second, one notices that replacing most of the regular QSim runs with PSim runs, as discussed in this chapter, results in significantly reduced QSim contributions to the overall wall clock time, even if one counts in the additional time for the PSim and the additional overhead. At this point, it was possible to reduce the computing time by more than a factor of two, when comparing the 16 core results from the default approach to the fastest version of using the 16 core machine with the multi-model approach.
Figure 6.4: Average executed score versus QSim iterations comparing two ratios of QSim:PSim to a QSim-only run.

Both multi-model runs have a replanning rate of 0.3.

An interesting result here is that lowering the replanning rate, while increasing the number of PSim iterations in the inner loop gives the best overall performance, with its most significant component being time spent on overhead operations. The reasons for this improved performance in comparison to the other 16 core multi-model run will be explored in the discussion section to follow.

6.6.4 Solution state

6.6.4.1 Departure profile RMSD

Departure profile RMSD, mode share and daily link volume RMSD for both modes of operation are compared against the reference run in Fig. [6.7]. Note from the shape of the RMSD plots that the system has not reached a stable state at the reference score iteration, therefore the system departs from this state in further iterations. This is due to the slow rate of evolution of the random-response replanning modules, and the large number of dimensions being explored in the model. The slope of the RMSD curves only drop off at much higher iterations, especially for departure profile RMSD.
Figure 6.5: Score evolution vs time for large-scale scenario, comparing the influence of QSim:PSim ratio, number of computational cores and replanning rate.

Both the standard QSim-only model and the multi-model approach reach their minimum RMSD value at the iteration where their score equals the reference score of 175.4. However the multi-model approach differs from the baseline by a larger margin than the QSim-only reference run at 101 iterations.

6.6.4.2 Mode share

The multi-model approach produces markedly different car mode shares when compared to the reference run (Fig. 6.7b). The swing towards public transport is much larger for the multi-model runs than for the reference run. The routing and travel time of transit is independent of network conditions for our simulations, as transit was not explicitly simulated in order to save simulation time. The meta-model gives many more agents the chance to consider that during the initial iterations, with lots of car congestion, public transit is an attractive alternative. An agent’s optimal departure time with public transit is, however, different from the same agent’s optimal departure time with car.
This swing to transit can be minimised by lowering the overall replanning rate, as well as the relative proportion of plans passed to the sub-tour mode-choice module. A run where this strategy was employed is indicated by the red line in Figure 5(b). For this run, the QSim:PSim ratio was set to 1:24, and the replanning rate at 0.1. The proportion of plans sent for sub-tour mode-choice mutation was set to half that of other replanning modules.

6.6.4.3 Daily link volume RMSD

The daily link volume RMSD does not show a minimum at the reference score iteration for any of the runs, and takes longer to reach a minimum. Even though the minimum value is approximately twice that of the reference case, it is still relatively small in absolute value.
Figure 6.7: Departure profile RMSD, car mode share comparison and daily link volume RMSD.

The plots compare the baseline case against the reference QSim case, and two multi-model runs with varying replanning rate and QSim:PSim ratio. Colored dots indicate the iteration where each run achieved the reference score of 175.4.

6.6.4.4 Agent total travel time difference

Table 6.1 compares the agent total travel time difference for the three runs at the reference score iteration, along with the other measures of solution state discussed above. \( RMSD_{dep} \) denotes departure profile RMSD; \( RMSD_{link} \) is the daily link volume RMSD; \( \Delta_{traveltime} \leq 5\text{min.} \) and \( \Delta_{traveltime} \leq 1\text{min.} \) denote the percentage of agents with a total travel time difference (from the baseline) less than 5 minutes and 1 minute, respectively; \( share_{car} \) denotes car mode share.
Table 6.1: Summary of solution state measures, compared against the baseline case.

<table>
<thead>
<tr>
<th>Run descr.</th>
<th>QSim iter.</th>
<th>RMSD$_{dep}$</th>
<th>RMSD$_{link}$</th>
<th>$\Delta_{traveltime} \leq 5\text{min.} , (%)$</th>
<th>$\Delta_{traveltime} \leq 1\text{min.} , (%)$</th>
<th>share$_{car}$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>101</td>
<td>32.56</td>
<td>5.67</td>
<td>77.1</td>
<td>66.4</td>
<td>80.7</td>
</tr>
<tr>
<td>0.3Q:P=1:9</td>
<td>20</td>
<td>50.85</td>
<td>24.53</td>
<td>74.6</td>
<td>65.0</td>
<td>76.2</td>
</tr>
<tr>
<td>0.1Q:P=1:24</td>
<td>13</td>
<td>56.42</td>
<td>30.00</td>
<td>76.1</td>
<td>66.7</td>
<td>80.0</td>
</tr>
</tbody>
</table>

Each measure is taken at the point where the average executed score is equal to that of the baseline QSim-only case, at iteration 101.

It was found that the magnitudes for $\Delta_{traveltime}$ between the three cases to be comparable; at least 74% of agents have a total travel time that lies within 5 minutes of that experienced in the baseline run.

### 6.7 Conclusion

The multi-model approach was designed to be consistent with the pre-existing simulation logic of MATSim, and it appears to produce comparable results. In all cases, using the multi-model approach reduces the number of time-consuming QSim iterations required to achieve a given average plan score.

#### 6.7.1 Performance

The multi-model approach scales well with an increasing number of cores. The experiments presented in this chapter revealed that the interplay of replanning rate and number of PSim iterations in the inner loop have an important influence on convergence rate. Having a relatively low replanning rate with a higher number of PSim iterations in the inner loop produces the target score in less QSim iterations and less wallclock time.

At first glance, this is a surprising result, because the expected number of plans generated from one QSim iteration to the next is comparable for the two 16-core multi-model runs in Fig. 6.6. The first run has a replanning rate of 0.3 and QSim:PSim ratio of 1:9. Consequently, in 1+9 iterations, the expected number of new plans produced per agent comes to 3, with a standard deviation of 1.44. In comparison, the second run has a replanning rate of 0.1 and QSim:PSim ratio of 1:24, so in 1+24 iterations, it produces only 2.5 new plans per agent on average, with a standard deviation of 1.5.

The reason for the quicker convergence is probably due to the larger number of combinations
of replanning modules that can act on any given plan in successive inner loop iterations for the second case. Even if any given combination has only a small chance of occurring; if it is favorable, it will be retained.

The expected value calculation also shows why the total replanning time of the second run is significantly less than the first: In total, it produces 16.7% less plans per outer loop cycle. It suffers, however, from an increased overhead due to a larger total number of iterations.

### 6.7.2 Solution state

Even though the different measures of solution state depart from those produced by the reference QSim-only run, the departure is not that great for the two measures critical to transport system performance, namely link volume and experienced travel time. The difference in mode share is a cause for concern however. A strategy to minimize the overshoot effect, is to lower the replanning rate and relative contribution of subtour mode choice to the replanning strategy. However, further investigation is warranted, in a comparative study with full transit simulation instead of the teleportation strategy used in this chapter.

This study also shows that it is important to consider the relative contribution of each replanning model to the simulation state, because utility on its own is not a complete indication of what is happening in the simulation.

### 6.7.3 Application to data-driven simulation

The multi-model approach should prove useful in reducing simulation times for most applications of MATSim. Its simple design should make it easy to maintain as MATSim functionality is extended. In this chapter, it has been shown to work well with an extensive list of existing MATSim capabilities. It remains, however, for the PSim approach to be tested with the data-driven transit simulation. As was argued at the beginning of the chapter, this extension will become essential as the scope of applications of the data-driven approach is increased to allow for more flexible replanning than only best-response re-routing.
Chapter 7

A simple framework for distributed simulation

When a MATSim scenario is relatively small, it is possible to run simulations in an appreciable time even on a laptop computer. As scenario complexity and size increases, the time required for replanning can easily outstrip that of running Queue Simulations (QSim), even when using meta-modelling techniques such as Pseudo-Simulation (PSim). Up until now, such scenarios have been run on Symmetric Multiprocessor Systems (SMPs), with a large number of computational cores, and hundreds of gigabytes of Random Access Memory (RAM).

The cost of ownership of such systems is restrictive, but the barrier to entry has been lowered by global access to cloud-based computing, like Amazon Elastic Compute Cloud, and Google Compute Engine. These services allow the user to rent access to fleets of high-powered computers at a nominal rate. However, the biggest and best virtual machines on offer from these services, while being large SMPs in their own right, still only have only a limited number of cores and RAM. Therefore, when simulations might require more computational capacity than what is available on any given SMP, there is a clear need for the computational load to be distributed across multiple machines.

Further motivation for running simulations in a distributed way, comes from the promise of possibly extracting deeper insights into the statistical properties of phenomena that a given simulation scenario might produce. The traffic conditions observed in any given MATSim simulation is only a single realisation of a stochastic system. It is quite possible that exactly the same simulation that shows free flow conditions in one realisation might produce gridlock in another. Our view of traffic simulation therefore needs to become more similar to that of weather simulations, where repeated simulations produced distributions of weather conditions, and our predictions therefore become probabilistic. Such “ensemble” simulations will also require computation power far in excess of what a single SMP can offer.
In this chapter, a concept design for a distributed simulation system is presented, based on the MATSim framework. This framework need not only apply to the smart card driven transit simulation that forms the subject of this thesis; instead it is a general framework that can be applied to most MATSim scenarios. Initially, only a master-slave configuration is proposed, but it should be a straightforward task to extend the approach to accommodate ensemble simulations.

The distributed simulation approach adds a number of parameters to the already large collection that forms the input for a MATSim run. Deciding on the correct values of these parameters has a strong influence on the rate at which the simulation will reach a relaxed state. However, with such a large number of variables, and long simulation times, one cannot proceed in a trial and error fashion in deciding on this parameter set. Instead, a simple Kriging model is constructed from a set of experiments designed using the Latin hypercube sampling methodology. Initial results show this approach to be promising, as a general technique for performing simulation-based optimisation in MATSim.

### 7.1 Motivation and background

As was argued in the previous chapter, the tight coupling of agents in agent-based transport simulation makes it a hard problem for decomposition and distribution across multiple nodes. The existing literature on parallel agent-based transport simulations (e.g. [Cetin et al., 2003; Charypar et al., 2007; Nagel and Rickert, 2001]) tend to focus on distributing the network loading step, rather than the distributed storing and mutation of agent plans as the system presented in this chapter attempts to achieve. However, these are important practical concerns for running large-scale simulations, as this section will motivate.

#### 7.1.1 Transit simulation sample size

The usual way to improve simulation times in MATSim, is to only simulate a sample of the full population. In the case of scenarios that only simulate private vehicle interactions on the network, the sample size can be as small as 1% and still produce realistic travel times, if network link capacities are appropriately scaled.

When transit is explicitly simulated on the network in MATSim, vehicles are dispatched for every departure in the transit schedule. There is no way past this requirement, as dispatching fewer vehicles would mean that the service operates at a lower frequency, and therefore the transit service performance will not be realistic. Simulating transit therefore has the minimum performance impact of simulating the movements of all transit vehicles.
One way of improving the computational requirements of simulation, is to also simulate only a sample of the full population, and scale the capacities of the transit vehicles in the simulation accordingly. If a vehicle approaching a transit stop is relatively full, then a single agent, now representing multiple agents due to reduced sample size, can take up all of its remaining capacity when boarding. The rest of the agents waiting at the stop will have to wait to board the next vehicle, and incur the penalty of waiting time. If this doesn’t happen very often, then the simulation will still produce realistic transit performance results, however one expects there to be a lower limit beyond which the dynamics of ridership become unrealistic. Furthermore, as the sample size is decreased while the number of transit departures remain the same, the number of transit vehicles on the network can adversely affect the simulation dynamics of other modes, as the number of transit vehicles start to rival their own, and take up too much capacity in the capacity-scaled network.

From many trials, it appears that, for most scenarios, a sample size of 25% presents a lower limit before capacity effects start to dramatically affect simulation results. This minimum sample size limitation consequently has implications, not only on the time taken for running the QSim, but also the time required for replanning, as well as memory requirements.

7.1.2 Agent memory and diversification

If we have to simulate large sample sizes, and scenarios become more complex, then it is important that every agent has the capacity to hold enough options “in mind” for the simulation to reach a state where valid conclusions can be drawn from results. The capacity for agents to store more than one day plan in memory, and then select from and mutate these plans, is central to the MATSim solution logic. If an agent doesn’t have the storage capacity, then plans that might have produced useful mutations later in the simulation will simply be discarded.

From the previous argument, if we need to simulate large sample sizes, as well as store many plans per agent, then the requirements of the simulation will ultimately exceed the available memory capacity offered by any given SMP. Consequently, the analyst will have to compromise on the agent memory capacity, or sample size, or both, and will never know what the implications of the compromise was on the quality of the results.

7.2 Design

It should be stated again that the aim is not to distribute the QSim across multiple nodes. Instead, the structure of the MATSim framework is exploited. As argued previously, the QSim is difficult to adapt in order to exploit multiple nodes, because of the high degree of synchronisa-
tion required between nodes for each simulation step. Replanning operations, on the other hand, can occur asynchronously, as each agent’s plans can be manipulated without coordinating with other agents. In this initial design, it is therefore only replanning operations that are distributed across multiple computational nodes. The quality of newly produced plans are evaluated using [PSim] which has been shown previously to fully exploit multicore architectures.

7.2.1 Master/slave configuration

The master/slave configuration is illustrated in Fig. 7.1. The tasks taken on by the master and slave nodes is illustrated in Fig. 7.2. The master node runs [QSim], capturing the interaction of all agents as in a normal MATSim run. It has several data structures that record transport system performance measures during the course of the simulation. These could include travel times on all links, and/or the time taken by transit vehicles to travel between bus stops.

The exact selection of performance measures that need to be compiled, depend on what information is required by [PSim] in order to produce an event stream that closely mimics that produced by the [QSim] on the master. Also, some replanning modules on slave nodes require information feedback from the [QSim] on the master in order to perform best response replanning. For instance, best response re-routing requires updated link travel time information in order for agents to avoid congestion.

In return, slave nodes run pseudo-simulations and perform replanning on the share of agents that have been allocated to them, based on the information they receive from the master. After a specified number of iterations, plans are selected according to a specified strategy and transmitted to the master.

Slaves can be configured independently from the master, in terms of maximum number of plans that they can hold per agent, and one is not precluded from allocating different replanning strategies to different slave nodes. The simplest is to keep the configuration on all slaves the same, and this approach has been followed in this chapter.

7.2.2 Serialisation

Initially it was considered to have the master transmit all scenario information to the slaves. However, as MATSim data structures are explicitly prohibited from being serialisable, it means that both the master and the slave nodes required procedures to copy scenario data into a serialisable format, transmit it to the receiver, then deserialise and repackage in the default MATSim format. It was therefore decided only to perform these actions for the most essential elements in order to get the distributed simulation to work, and minimise the number of sites for potential
Figure 7.1: Master/slave configuration for distributed simulation.

In this configuration, the master exclusively runs QSim, while the slaves exclusively run PSim. The master provides slaves with updated travel times, waiting times and other pertinent simulation performance parameters. The slaves perform replanning and evaluate plan performance with PSim, then select plans and transmit them to the master for execution in QSim.

Generally speaking, the XML file containing agent plans is the largest data structure that needs to be loaded in any given MATSim scenario. This is especially true if there is more than one plan per agent. In most scenarios, transport supply information remains constant throughout the MATSim simulation run; there is arguably no need to transmit supply data structures across the network between nodes (except for supply performance measures, that is). If simulation data is read from a shared location, then the performance impact of all slaves reading the transport supply information is a relatively small. However, for all slaves to read the entire population file containing all agents and their plans, in order for them to only store a small share of the information, is wasteful. Therefore, it was decided to not only create a serialisable plan object, along with all its component classes, but also to make agents serialisable.

**Initialisation** Having serialisable agents means that the population file is read only once by the master node. The master then transmits a share of agents and their plans to each one of the slaves. Slave nodes deserialise these objects, and copy them to default data structures. Slaves can then perform the initial routing of plans if plans have not been routed. Initial routing is usually a costly initialisation step that can take an inordinately long time to perform on a single computer for a large network, especially when routing transit. In the case of the Singapore scenario, the initial routing time was reduced from two hours down to five minutes when the task was distributed across ten 24 core nodes on the ETH Zurich Euler cluster computer.
Load balancing Another important implication of being able to serialise not only plans, but also agents, means that it is possible for the master node to perform load balancing. Generally speaking, this is not necessary in a homogeneous network, such as a cluster computer or a cloud-based computation facility. The capacity of all slaves is equal, and therefore they would require the same amount of time, more or less, to perform pseudo-simulation and replanning. However, if the network consists of heterogeneous nodes, with different amounts of memory and levels of performance, then one cannot assign the same number of plans to all nodes. Instead, the master has to monitor their performance and redistribute agents away from nodes that perform poorly, or that are about to run out of memory.

A simple load balancing algorithm was developed that prioritises memory over performance, as running out of memory on any given slave node would ruin the entire simulation. However, if memory is not a limiting factor, then the load balancing algorithm will determine the time taken to perform operations on each agent, by each of the slave nodes. The master will then reallocate agents across slave nodes in such a way that the projected total processing time will be equal, and no single slave node will hold up the simulation process.
7.2.3 Serial vs. parallel operation

The master node can be configured to have slave nodes perform replanning and pseudo-simulation while it is executing the \textit{QSim} (parallel mode), or to wait for the latest transport system performance measures, and only then run a specified number of iterations (serial mode).

In parallel mode, slave nodes will pass the master a set of plans that was created and executed in \textit{PSim} against travel time information from two \textit{QSim} iterations earlier. For serial mode, slave nodes are fed information from the previous \textit{QSim} iteration on the master. In serial mode, the distributed simulation operates analogously to the alternating \textit{QSim}/\textit{PSim} simulation that was the subject of Chapter 6.

Generally, traffic conditions do not change that much from one iteration to the next, and the benefits of having updated travel time information fed to the slave nodes is outweighed by the increased time required for the master to wait until all replanning has completed on slave nodes. However, the serial mode functionality was included, as some capacity constrained scenarios appear to require travel time information to be more up-to-date.

If the system operates in parallel mode, then generally, slave operations can complete before the master has finished \textit{QSim} and possible replanning operations of its own. The master accepts incoming plans from each slave node on a separate thread, and therefore the \textit{QSim} is not affected if the master has more cores available than the number of threads allocated to the \textit{QSim}.

The system has been designed to accept additional slave node connections even while the simulation is running. When additional nodes are registered with the master, it will redistribute the load across all slave nodes, and the total time taken to complete slave operations will decrease. In this way, especially in parallel mode, the impact of replanning and \textit{PSim} on the slave nodes can be minimised to the point where the master does not need to wait for slave operations at all. One can simply keep adding slave nodes to the simulation until the time required for operations on slaves is less than the time required for \textit{QSim} and replanning on the master.

7.2.4 Distributed simulation as a replanning strategy

From the outset, the distributed simulation was implemented as a replanning strategy on the master node. This meant that a lot of details about how slave nodes operate could be abstracted away, and the master node could largely operate similar to a normal MATSim run. In fact, for both the master and slave Java classes, the standard MATSim controller was assigned as a delegate, and the required communications are invoked using event handlers that are registered with the delegate simulation controllers.
In initial experiments, all plans were stored on slave nodes. The master only held a single plan for each agent, as each agent’s selected plan was replaced by one supplied by a slave node. The number of plans on the master node was limited to one per agent.

However, it was found, for some scenarios, that having only a single plan on the master produced erratic results. For instance, the MATSim scenario developed for Singapore used a navigation network that was transformed into a MATSim network description, with capacity information derived from road classifications. This network initially had a lot of capacity bottlenecks that would even cause a normal MATSim run to produce erratic jumps in the executed score, as gridlock would sporadically arise from one iteration to the next.

In a default MATSim run, alternative plans were available for each agent, so the simulation would recover well from erratic gridlock in subsequent iterations. It is reasonable to assume that poorly performing plans that caused gridlock in the first place, would be discarded from agent memory. However, if the master only has a single plan per agent, replaced every iteration by another plan from the slave, this information is lost immediately.

Consequently, initial tests on the Singapore network produce results that were even worse than a standard MATSim run. The decision to implement the distributed simulation as a replanning strategy proved fortuitous for network capacity-constrained cases such as this. By increasing the number of plans stored on the master to a reasonable number for each agent, and performing some replanning operations on the master itself, the problem was eliminated and the distributed simulation performed better than the default MATSim configuration. However, for most cases, if demand and supply are well matched, the simulation will perform well even if there is very little or no replanning on the master.

7.3 Method

The perspective taken in this set of experiments, was to maximise performance, in the sense of minimising the number of iterations. As was already argued, the time required for operations on slave nodes can be reduced to a negligible amount, by simply adding more nodes. This is feasible in the age of cloud computing, as extra capacity can be rented at a nominal rate at short notice. The determinants of total simulation time is therefore the time taken to run the QSim on the master as well as any optional replanning that may take place on the master, and the number of iterations to reach a target score.
7.4 Scenario

The system is designed for general, large-scale MATSim scenarios, with the possibility to extend in future to the data-driven transit simulation. For the performance tests, a large-scale, general MATSim scenario of Singapore was used that has both private and public transport simulated on the network, while car passengers and taxi users are ‘teleported’.

The Singapore MATSim scenario contains the following data structures:

**Network.** 20,345 nodes, 40,115 links, of which 1,381 are exclusive to bus and 250 to rail.

**Facilities.** 110,677 activity facilities.

**Transit.** 4,995 transit stops, served by 337 lines, resolving to 786 route profiles, and a total of 52,401 services run over the course of the simulated day.

**Demand.** A 25% sample of the travelling population, represented by 978,628 agents.

7.4.1 Testing overhead impact of increasing number of nodes

The assumption that increasing the number of slave nodes does not have an impact on the simulation time, needed to be tested first, as in general for distributed applications, an increase in the number of computational nodes increases the overhead of communication required between nodes. Generally one expects performance to increase until a point is reached where the overhead produces diminishing returns or even performs more poorly with increasing nodes.

7.4.1.1 Euler computer cluster, ETH Zurich

Running the distributed simulation on the cluster computer at ETH Zurich means that slave nodes are relatively “fat”; each node is a high-end SMP in itself with 24 cores and between 64 and 256 GB of RAM. Individual nodes are also connected via a very fast network physical layer that runs at speeds comparable to those of the integrated circuitry of the computers that it is composed of. The scheduling system of the cluster computer generally does not allow one to assign more than 20 nodes to any given job, and no experimental configuration could be devised where the time required for communications had any significant impact on performance.
7.4.1.2 Amazon Elastic Compute Cloud and MIT StarCluster

The distributed simulation framework was also tested on the Amazon Elastic Compute Cloud (EC2) using MIT’s StarCluster software. StarCluster is a set of Python modules that automatically configures nodes in the Amazon cloud in such a way that they all reside within the same physical network, have access to a single shared network volume, and password-less secure shell access between all nodes.

The system also allows one to secure capacity using Amazon spot instances; this is excess computational power that Amazon auctions off at highly reduced prices. One need only specify a maximum bid per spot instance, and StarCluster will secure and configure the requisite number of virtual machine instances and maintain the cluster until the specified spot price is exceeded by another bidder. At the time of writing, it was possible to secure a spot instance 40 core, 256 GB RAM node at less than US$.35 per hour while a 4 GB, two core node cost less than two US cents per hour.

A distributed simulation was set up with a master running on a 40 core node, and all slaves running on 4 GB, two core nodes. The maximum allocation of 50 spot instances allowed by Amazon was utilised. Once again, no appreciable influence on simulation time due to the overhead of communicating with 49 slave nodes was detected when running the distributed simulation in parallel mode.

This result can possibly be explained, not only by the fact that both the Euler cluster computer and Amazon’s own resources run on very fast physical network layers, but also by the fact that relatively little communication is required between master and slave nodes. Communications need only happen once per iteration, and the time taken for communication, on a separate thread for each slave, is far less than that required by the QSim for a reasonably sized network.

Of course, for both experiments, the master was running on a node with excess capacity, which meant that the threads that manage tasks on slave nodes, and receive and transmit to the slaves, would never interfere with the QSim on the master. It is possible that less positive results might be achieved in a heterogeneous network running on a slower physical network layer, and if the master has less excess capacity. However, with cloud computing becoming cheaper by the day, and spot instances allowing one unfettered access to one’s own dedicated supercomputer at under US$10 an hour, the argument hardly seems worth discussion.

7.4.2 Experimental design: finding optimal parameters

As the number of slave nodes was not found to be a limiting factor in performance, the only remaining question was to come up with an optimal set of simulation parameters that would
reduce the number of \texttt{QSim}s required in a simulation run to a minimum. This means running a large number of time-consuming simulations, as there is no analytical way of establishing the correct parameter set.

The cost of performing this parameter search is justifiable. In most contexts, a scenario will be run repeatedly in different experiments, so the overhead of running these parameter search experiments will pay off in the total time saved when running hundreds of simulations. Furthermore, as was argued previously, the cost of securing the computational resources in order to run multiple simulation experiments in parallel is very low.

A popular way of performing this kind of simulation-based optimisation, is to fit a Kriging model to a set of simulated observations (e.g. Chen et al., 2014). Instead of providing a formal treatment, the approach will be explained qualitatively, and the interested reader is referred to e.g. Kleijnen (2009) for a review of the technique.

The basic principle behind Kriging is illustrated in Fig. 7.3. Here, the simulator is some black-box function producing the dashed grey line if it were to be run for all values of the dependent variable. However, as it is an expensive simulation, we only have a few observations from a (poor) experimental design. The simulation was run for a random selection of dependent variable values, producing the observations in red.

In its simplest form, a Kriging model is a weighted averaging of observations, with the weights dropping off with increasing distance (the blue line in Fig. 7.3). The rate at which the influence of observations decays is determined by the covariance function specified for the model. In such an ordinary Kriging model, values tend toward the average of all observations, but are drawn to pass through the observations themselves. The solutions produced by ordinary Kriging are sufficient for many applications. However, if there is a known trend in the response variable as a function of the input variables, then a universal Kriging model will tend towards the OLS regression line when far from observations, as is illustrated by the line drawn in teal.

7.4.2.1 **Objective: highest score in fewest QSim iterations**

To perform experiments in a practical time frame, each simulation was not run until a specified score was reached, as was the case in Chapter 6. The expectation was that such an approach could take an inordinately long time if the choice of parameters were poor. Instead, the assumption driving the experimental design is that the best performing set of parameters will also produce the highest executed score in a pre-specified number of \texttt{QSim} iterations.

It was therefore decided to only run 20 iterations on the master for each experiment with innovation turned on, after which four more iterations were run with innovation turned off and no new plans being received from slave nodes. This means that the simulation then only runs
Figure 7.3: Example comparing two Kriging models estimated from sparse observations, against the true response of an unknown function with linear trend.

Including the trend in the Kriging estimation improves the fit, especially for predicting outside the range of observations.

with whatever plans are in the master node’s memory. The number of plans agent on the master was limited to 3, while the number of plans the agent on each slave node becomes one of the parameters of the experiment.

7.4.2.2 Parameters and replanning strategies

Six parameters were identified to be potentially relevant in maximising the average executed plan score in a limited number of iterations:

- **master_replan_rate** The total mutation rate on the master (excluding the replanning strategy for receiving plans form slaves).
- **injection_rate** the proportion of plans received before every QSim from slave nodes.
- **slave_plan_memory_cap** The total number of plans per agent kept in memory on slave nodes.
- **serial_operation_mode** This is a dummy variable that has a value of one if slaves operate in serial mode, and zero if the slaves are allowed to perform their operations in parallel to the QSim on the master.
- **psim_iters_per_qsim_iter** As in the previous chapter, the number of PSim iterations that are allowed to run on slave nodes for every QSim iteration on the master is expected to be a
The same set of replanning strategies were employed on all slaves, as well as the master node, with the exception of the strategy on the master that accepts plans from slaves. The strategies include the routing of plans based on the latest travel time information, the mutation of the start time and duration of plans, and switching of trip transportation mode between private car, transit, car passenger, and taxi mode. All of the replanning strategies have the same weight assigned to them, making it equally likely for an agent to be subjected to any of the three strategies if they are selected for mutation. For agents that are not selected for mutation, the best performing plan in their memory is selected for execution.

7.4.2.3 Latin Hypercube Sampling

A suitable combination of all the parameter values needed to be tested, selected in such a way that a robust Kriging model can be estimated from as few simulation runs as possible. A good experimental design for a Kriging response surface generally requires that each observation has its own distinct set of parameter values selected in such a way that the parameter space is adequately explored across all dimensions, and any trend effects can be picked up by OLS regression.

An experiment was designed based on the Latin hypercube methodology to test combinations of the six parameters in 30 individual simulation runs. A Latin hypercube design is a space-filling design, well-suited to expensive experiments. Parameter values are chosen in such a way that they appear to be uniformly distributed across all dimensions.

The default methodology was modified so that the expected number of plans produced per agent on slave nodes is limited to a maximum of five. This number is the product of the `psim_iters_per_qsim_iter` and `slave_replan_rate`. Furthermore, the total replanning rate on the master cannot exceed a value of 100%, therefore the variables `master_replan_rate` and `injection_rate` need to add up to one.

The distribution of simulation parameters, as well as their associated average executed score after 20 QSim iterations with innovation on and a further 4 iterations with innovation off, appears in Fig. 7.4.
Figure 7.4: Experimental design for deriving Kriging model of distributed simulation parameters

Scores of executed plans in each simulation indicated by colours ranging from blue to red; refer to the left-most column of plots to relate colours to score values.

7.5 Results

A number of variations of Kriging model parameters were tested. These included estimations of the trend with first order interactions between variables. However, while using first-order interactions might improve a regression of average executed score versus the parameters, it constitutes over-fitting for such a small number of experiments. Estimating a Kriging model with first-order interactions produces predicted values that are unrealistic when compared with
Table 7.1: Kriging model estimation results for predicting average executed score after 20 mutating and 4 non-mutating QSim iterations.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Trend coef.</th>
<th>Covar. coef.</th>
<th>Quasi-optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>76.579</td>
<td></td>
<td></td>
</tr>
<tr>
<td>master_replan_rate</td>
<td>12.201</td>
<td>0.195</td>
<td>0.246</td>
</tr>
<tr>
<td>injection_rate</td>
<td>10.932</td>
<td>0.259</td>
<td>0.334</td>
</tr>
<tr>
<td>slave_plan_memory_cap</td>
<td>-0.055</td>
<td>190.000</td>
<td>3</td>
</tr>
<tr>
<td>serial_operation_mode</td>
<td>-1.149</td>
<td>2.000</td>
<td>0</td>
</tr>
<tr>
<td>psim_iters_per_qsim_iter</td>
<td>0.171</td>
<td>73.019</td>
<td>29</td>
</tr>
<tr>
<td>slave_replan_rate</td>
<td>20.604</td>
<td>0.206</td>
<td>0.172</td>
</tr>
</tbody>
</table>

Variance estimate: 63.55148

The quasi optimum refers to the set of parameter values produced by differential evolutionary optimisation with an objective function that enforces the constraints described in the text.

Generally, finding the correct set of parameters for a Kriging model is an extensive investigation in its own right. For this exploratory application, a number of different Kriging models were estimated using different covariance functions and trend model formulations. The suitability of the models were ranked based on the subjective plausibility of predictions for input values that lie close to the observations. Another seat-of-the-pants validation method that was used is the so-called leave-one-out cross validation. Each observation is left out in turn, and its value predicted by a Kriging model with the same parameters passing through through all the other points. Competing models can thus be ranked for their predictive accuracy, by minimising the difference between the leave-one-out predictions and their actual values. From this process, a model was selected with a trend component that is a simple linear model of all variables, with no interaction effects, and the default Matérn (5/2) covariance function for the weights of observations.

The results of the Kriging model estimation can be seen in the first two columns of Table 7.1. From this estimation can be seen that the variables that are influential in the trend components of the estimation, tend to be less influential in the covariance component, and vice-versa. The strongest determinants of trend appear to be the three rate parameters, with the replanning rate on the slave nodes having a coefficient nearly twice that of the master.

A Kriging model with a trend component can be used, within bounds, to optimise the parameter of interest. The DiceKriging package (Roustant et al., 2012) in R ships with a number of optimisation routines for this exact purpose. However, as our solution space is constrained as described before, the default optimisation routine for finding the set of parameters that would maximise average executed plan score would constantly stray into the infeasible region. In-
Figure 7.5: Parameters from from DEO solution (red dot) and surrounding solution space in Kriging model for a selection of variables.

Each plot is a plane through the solution. The dotted red line in bottom-righthand plot shows the constraint in the experimental design, to the left of which all combinations of `psim_iter_per_qsim_iter` and `slave_replan_rate` will produce a maximum of 5 plans per agent on slave nodes per QSim iteration.

Instead, an objective function was formulated with the Kriging model solution as its output if variables were in the feasible domain, and a large penalty if infeasible. The objective function was optimised using DEO as explained before.

The best objective function value was found to be 110.54 utils, for a set of parameters as can be found in the last column of Table 7.1. When a simulation was run with these values, it produced a value of 111.89 utils, which is sufficiently close to the predicted value from the Kriging model. An OLS model with no interactions predicted only a value of 98.63 utils, which proves that the response surface is sufficiently altered by neighbouring observations to
produce meaningful predictions.

A set of plots of the neighbourhood of the DEO solution, for combinations of the most interesting variables (those that show most variation), is shown in Fig. 7.5. It appears that one might have even done better than what the optimisation procedure found, and that DEO might not be the best optimisation procedure for the task. This plot also shows that the parameters master_replan_rate, injection_rate and slave_replan_rate can be traded off against each other over a relative large range for only a small effect on objective function value. Furthermore, from the contour plots containing the variable psim_iters_per_qsim_iter, it appears that its covariance coefficient has a strong effect on the behaviour of the model, even though it has a relatively small trend coefficient.

7.6 Conclusion and outlook

This chapter has shown that the distributed approach brings more flexibility to the mixed simulation described in the previous chapter, while overcoming the limitations of memory and processing power of a single server, at least for replanning operations.

The experiments to determine distributed simulation parameters suggest that Kriging could be a viable way for performing simulation-based optimisation in MATSim. While the experiment in this case was designed to optimise the distributed simulation parameters for producing the largest average executed score in the specified number of QSim iterations, it can be similarly used to find a set of cost parameters that would produce a target mode share distribution, or vehicle count on selected links, etc.

A possible critique of the experimental approach is that the number of iterations is not used as a predictor variable, and there is no variation in the number of iterations across the different simulation runs. One is therefore reluctant to use the model to predict extrapolated values for the average executed plan score at higher iteration values. An experiment design using the Latin hypercube sampling methodology, with the number of iterations as the seventh variable, could be used to estimate a new Kriging model of average executed plan score. The model can then be used to draw an efficient frontier for all the other variables where the average executed plan score is above a certain critical value.

Another Kriging model can be estimated to predict the total simulation time given the combinations of variables, assuming that all simulations were run on the same server fleet. The efficient frontier can then be complemented with an expected total simulation time and the experimenter would then be free to select a combination of variables expected to minimise simulation time given the computational resources available. Unfortunately, the experiments in this chapter were not always run on the same hardware, and neither were there any guarantees about the
allocation of resources to the experiments when they were run, therefore no firm conclusions on total simulation time could be drawn.

7.6.1 Next step: ensemble simulations

Because ‘master’ and ‘slave’ are really only arbitrary labels for the architecture proposed in this chapter, one can do away with these concepts and think of general information exchange between MATSim simulations. In this case, simulations would become peer-to-peer, and the same set of communications that were used in this chapter would be used to transmit plans between peers, with each peer replanning only a portion of the agent population. In this way, one can run arbitrarily modified MATSim simulations, while exploiting the capabilities of distributed replanning, without having to know anything about how the MATSim simulation works. If each simulation had its own unique random seed to drive stochastic operations, then the result would be an ensemble simulation, and one would be able to analyse the stability of traffic patterns across multiple simulation runs.
Chapter 8

Surrogate data synthesis

The current version of the data driven transit simulation relies completely on the actual transit smart card records. In presentations of the method to interested authorities, privacy concerns are frequently raised. This chapter presents a first attempt to synthesise surrogate data from aggregate sources, with the intent that the surrogate data can be used as an agent-based demand. The goal is for the surrogate to have the same statistical properties as the original data and match the spatial and temporal levels of detail required for agent-based simulation. What remains to be proven is that a simulation with the surrogate demand will produce results that are indistinguishable from those produced by the original data.

This chapter was submitted in very similar form for presentation at the 2016 Autonomous Agents and Multiagent Systems International Conference [Fourie 2016b].

8.1 Motivation

Data sources that provide space-time stamps of individuals at appreciable resolutions are heavily guarded, for obvious reasons, and rarely make their way into the broader research community because of the associated privacy concerns. When they do, their use is usually subject to very strict data privacy controls, governed by non-disclosure agreements that consume considerable time and effort to negotiate.

The more people that have access to a data source, the more reluctant the authority should become to giving others access, as their exposure increases with every new user. Identifying a leak becomes more difficult and policing the data requires more resources. This makes research very difficult to replicate and validate by researchers that do not enjoy access. It also allows a preferred group of researchers to set the agenda when new data becomes available.
8.1.1 Aggregates

Authorities and data stewards are generally amenable to release aggregated totals of sensitive information; for instance, most census bureaus will release marginal totals and joint distributions of population attributes at aggregate spatial resolutions. The resolution of these aggregate distributions is generally such that no individual will be exposed, and in many cases authorities further protect individuals’ data by randomising the value of aggregates that are below a certain minimum threshold.

Such aggregates may be released one-time only, or the data steward might provide a service that allows interactive queries. With the case of interactive queries, repeated queries might make it possible to determine individual information. To combat this, so-called differential privacy algorithms add noise to aggregates, limit the number of queries a user can make and ensures that every request has a different layer of noise applied to it.

In the field of transportation planning, aggregate travel information might be released in the form of origin-destination (OD) matrices by mode of transport and time of day (usually morning and evening peak, and off-peak periods). While, in general, flow data is considered to be sensitive, such aggregate flows are more likely to be released and form the basis of static transport assignment models.

Typically, when disaggregate information is lacking, an agent-based transport demand can be synthesised from such an OD matrix (e.g. Fourie 2010). However, because of the coarseness of especially the temporal resolution, an agent-based demand that is synthesised from OD matrices, will not nearly reflect the richness of the original disaggregate data source that was used to generate it. Furthermore, current approaches to representing transport demand in an aggregate form can not characterise the entire day tour of the individual; this information is lost in the process of aggregation.

How can the information that an authority releases at aggregate level from disaggregate data sources be maximised to the extent that one (a) can construct realistic day tours from the aggregate data that reflect the same statistical properties as the original disaggregate data sources; (b) while at the same time guaranteeing the privacy of individuals to the extent that the reconstructed day tours do not correspond to the trajectory of any given individual in the original dataset?

8.2 Overview

This chapter is an exploration of an approach that allows one to synthesise a high dimensional data set from two-dimensional aggregate distributions, of arbitrary resolution, derived from the
Figure 8.1: Pairwise histograms of smart card data and its principal components.

Two-dimensional histograms of all pairwise joint distributions of public transport trips, in principal component and original variable space, that form the primary input for high-dimensional data synthesis. Blue values represent relatively low counts, going towards reddish colours for high counts. Relative scales for plotted values are left out intentionally; this figure is only serves as an illustration of the input data required for an Iterative Histogram Matching (IHM) synthesis, and to illustrate how principal component plots display more variation and joint structure than the original variable space plots.

original big data source. This high-dimensional dataset can be a reconstruction of the joint distribution of individual trips or entire day tours of individuals within the study area. In fact, the approach shows potential to be near-limitless with regard to the set of attributes whose joint
distribution one attempts to reproduce.

The method relies on iterative histogram matching of rotations of the supplied two-dimensional aggregate distributions, in principal component and original variable space. The method makes it possible to record an entire high-dimensional dataset as a series of images, such as is shown in Fig. 8.1 and then reconstruct a very close approximation of the original dataset from the images.

The bottom left-hand quadrant of Fig. 8.1 shows two-dimensional histograms of all pairwise combinations of trip start/end time and origin/destination longitude/latitude coordinates derived from transit smart data collected during a typical weekday in Singapore. The temporal resolution is five minutes, while spatial data has been aggregated into bins of 500 metres along both the X and Y axes. For the purpose of this chapter, transit smart card transaction X-Y coordinates have been randomised to be normally distributed from their actual discrete locations at bus stops and train stations, with a standard deviation of 100 metres along both axes. The idea is that, in this way, one can represent more general data sets where movement does not only occur between a limited number of points, but across continuous space.

The top right quadrant of Fig. 8.1 shows the two-dimensional histograms of all pairwise combinations of principal components of the smart card data. In all the plots, the blue end of the spectrum denotes low counts for each two-dimensional bin, while the red areas denote high count values.

In order to construct a synthetic transport demand which would very closely match the actual observed trip making recorded in the original smart card data set, the method presented here would require

- the series of images as an input;
- along with a mapping of the colour values to count values;
- the extent of each of the recorded variables (minimum and maximum values);
- the vectors for scaling and centring the data;
- the rotation matrix that produces the principal component projection.

An important first insight into the method is that it would also make it possible to construct agent-based transport demand scenarios, by simply altering any of the images that served to describe the joint distribution, using a simple digital painting program. This is possible because, in contrast to other fitting approaches such as iterative proportional fitting and generalised raking, the method of iterative histogram matching is insensitive both to minor inconsistencies between the various two-way projections that constrain the resultant synthesised dataset; as
well as the so-called zero-cell problem. It therefore presents a simple and intuitive approach to construct a high dimensional agent-based demand without the need for complex activity based demand generation.

Most importantly, this approach makes it possible to make an entire data set available to other researchers in safe and anonymous form by providing such an image series and supporting information inside a digital document, or simply synthesising a surrogate data set for release.

The following section provides some background on high-dimensional data synthesis approaches in the literature, as well as the methods of principal component analysis and iterative histogram matching that were used to derive the approach presented in this chapter. This is followed by a description of the method and the input data that was used, as well as a section discussing the quality of results achieved. The chapter ends with a discussion and an agenda for future work.

8.3 Context

The purpose of this section is to broadly frame the data synthesis technique presented in this chapter in the context of current data anonymisation techniques.

8.3.1 Quantifying privacy

The advent of large-scale, affordable computation, storage and the world wide web, created increased awareness about mining data for model development, and how making data available to third parties might affect individuals who have provided their information. Until 2000, several authors had identified various methods to anonymise data; however, the challenge was still to satisfactorily quantify and guarantee the degree of anonymisation.

Sweeney (2002) provided a set of procedures in order for a data release to comply with one of the first generally acceptable measures of data anonymity in the modern age: $k$-anonymity. The author defines $k$-anonymity at the level of data releases: A release provides $k$-anonymity protection if the information for each individual contained in the release cannot be distinguished from at least $k - 1$ individuals also appearing in the release.

$k$-anonymity appears to have garnered interest across various domains as a standard of data anonymisation, later to be supplemented by various other methods named for their parameters; e.g. ‘$l$-diversity: Privacy beyond $k$-anonymity’ (Machanavajjhala et al. 2007), and ‘$t$-closeness: Privacy beyond $k$-anonymity and $l$-diversity’ (Li et al. 2007).

\footnote{\textsuperscript{1}See Müller and Axhausen (2012) for a review of fitting algorithms and their limitations.}
8.3.2 Privacy protection specialization

The field of data anonymisation is active and evolving, and it is easy to imagine why. For the literature of protection methods, there is a shadow literature for attackers, attempting to exploit weaknesses in data protection techniques, and evolving their own techniques in an ever-spiralling arms race.

For a field that had to continuously adapt in response to the evolution of Information and Communication Technology (ICT), emergent applications, and concomitant attacks, the need for domain-specific specialisation became very clear. Spatial data privacy protection had been recognised early on as a special case. Once spatial data monitoring had become more pervasive and real-time, e.g. two-way GPS and cellular phones, data stewards became interested in monetising these sources while maintaining user privacy. One of the earliest treatments of anonymisation of real-time spatial data is by Beresford and Stajano (2003), introducing the concept of spatial 'mix zones'; where individuals’ identifying pseudonyms are periodically replaced, making it very hard to identify who entered and left a particular area.

8.3.3 The special case of trajectories

Persistent data monitoring highlighted a further concern in spatial data privacy: it is not enough to anonymise locations such that at least \( k - 1 \) or more individuals might be occupying an individual’s location at any given time. Studies such as De Montjoye et al. (2013) show that sequential spatial observations, e.g. trajectories, can be reconstructed and compromise individual privacy, even if times and locations are known only approximately. Therefore, trajectory privacy preservation became a specialisation in its own right.

Chow and Mokbel (2011) provide a taxonomy of trajectory privacy-preservation techniques in their review of the state of the art in 2011. The quality of privacy protection is determined, for most cases, by the \( k \)-anonymity measure; i.e. a particular user’s trajectory cannot be distinguished from \( k - 1 \) other users. They classify privacy-preserving techniques according to the method and application. Techniques are classified into three categories:

**False locations.** The user sends a false location, or actual location along with several fake locations.

**Space transformation.** Location information is transformed into another space where spatial relationships are encoded.

**Spatial cloaking.** User locations are blurred until \( k \)-anonymity is satisfied.
8.3.4 Continuous trajectory privacy protection

Chow and Mokbel (2011) distinguish between protecting privacy in continuous Location-Based Services (LBS), where the data provider needs to protect privacy in near-real time, versus protecting privacy in trajectory publication, where data is mined and analysed post-hoc. For the first case, they recognise two sub-categories: where a consistent user identity is required (usually a pseudonym or dynamically generated identity), and a second category where it is not. They review seven different techniques, all of which qualify for the second category, while only two techniques are appropriate if consistent user identities are required.

The two universal techniques are spatial cloaking and dummy trajectories. With spatial cloaking, the blurring of users’ locations is generally achieved by grouping user stay locations together into bounded areas until $k$-anonymity is satisfied, and the trajectory is passed to the query as the set of bounding boxes and time intervals of the stay locations. Dummy trajectories is a technique where a set of fake trajectories are selected and passed to the query along with the user’s actual trajectory, with certain measures guaranteeing the privacy and quality of the fake trajectories.

Other techniques rely on perturbing the user identity and, as such, cannot provide persistent user identities. These techniques include:

- **Mix-zones** where the user enters an area where their identifier is swapped out for a new, unused pseudonym. $k$-anonymity is achieved if the mix zone contains at least $k$ users who are all together in the zone together for a specified period of time, and each spends a random duration inside it. They identify the special case of vehicular mix zones; where an adversary could exploit turn restrictions, traffic light timings and other attributes of transportation networks to link pseudonyms together in order to expose the trajectories of individual users if special provisions are not made.

- **Path confusion.** Multiple observations of vehicle location samples with anonymised pseudonyms can be linked together based on spatio-temporal correlation. Path confusion prevents such target-tracking algorithms from linking together consecutive observations with a high degree of confidence.

- **Euler Histogram-based on Short IDs.** This is a special case for answering aggregate queries in traffic monitoring and is arguably the most complex technique listed in their review. The reader is referred to Xie et al. (2010) for more details.

8.3.5 Published trajectory privacy protection

Chow and Mokbel (2011) go on to identify four anonymisation techniques for releasing trajec-
tory data for spatiotemporal range queries and data mining.

**Clustering-based anonymization.** Trajectories that are within a pre-specified radius from each other are grouped together into an aggregate trajectory, which is the arithmetic mean of the individual trajectories.

**Generalisation-based anonymisation.** For algorithms that cannot run on aggregate trajectories, this approach allows for individual trajectories to be synthesised by sampling from likely locations in sequence.

**Suppression-based anonymisation.** For the special case where locations are not continuous but discrete, and adversaries can have access to subsets of a person’s trajectory, conditions exist where they might be able to link their sub-trajectories to a larger, anonymised trajectory set and identify individuals in the larger set. This method uses an algorithm to iteratively suppress locations until the probability of inference is sufficiently low.

**Grid-based anonymisation** subdivides space in to a rectangular grid, with grid size selected to satisfy specified privacy constraints. Neighbouring grid cells are grouped into partitions, and trajectories crossing from one partition to another are split into two sub-trajectories. Movement within a partition is masked; only the time spent is reported. This approach is therefore more useful for answering spatial density queries.

### 8.3.6 Generalisation-based anonymisation

From the review, the approach presented in this chapter is probably closest to generalisation-based anonymisation [Nergiz et al., 2008], as it is also a synthesis-based method. Essentially, it is a method for obtaining $k$-anonymity for published trajectory databases. It is divided up into an anonymisation (i.e. encoding) and reconstruction step, similar to the the encoding and decoding steps of the method presented in this chapter.

**Anonymisation** is summarised in Fig. 8.2 for attaining $k = 3$ anonymity. The iterative process starts by selecting a single trajectory and its closest neighbour in space and time. Additional trajectories closest to the bounded regions are found and added until the group contains $k$ trajectories. In each addition, as for trajectory $T_3$ in the figure, the bounding boxes in space and time are expanded, and additional points outside a certain threshold are discarded, as with point $t_7$ in the trajectory.

**Reconstruction** is illustrated in Fig. 8.3. $k$ locations are uniformly sampled from each of the bounding boxes, and linked together to reconstruct $k$ trajectories. The authors came up with a fast nearest-neighbour algorithm and an objective cost function to optimise the grouping of trajectories in the anonymisation phase. They test the quality of the reconstructed trajectories
The generalisation-based approach trades off spatiotemporal accuracy for trajectories that are
Figure 8.3: Generalisation-based anonymisation: reconstruction step.

(a) Location samples selection

(b) Trajectory reconstruction

Source: Chow and Mokbel (2011), adapted from Nergiz et al. (2008).

less likely with increasingly stringent minimum privacy requirements, as the area from which positions are sampled need to be drawn large enough to satisfy $k$-anonymity requirements. As the histogram matching approach described in this chapter instead tries to record the joint distribution of stay locations, it can describe unlikely trajectories probabilistically at arbitrary resolution.

8.3.7 Surrogate data synthesis in agent-based transport demand modelling

Data synthesis from aggregate sources is not new in the field of agent-based transportation modelling. Especially in the field of population synthesis, a lot of work has been focused on deriving the full joint distribution of population attributes from census control totals and, usually, a micro-sample of households records that have been sanitised of any identifying information. Recently developed procedures have also looked at synthesising the actual reference sample through machine learning techniques and simulation-based approaches (e.g. Farooq et al. 2013, Sun and Erath 2015).

Unfortunately, these techniques cannot be applied in the context of reconstructing transport demand at the level of detail that we are interested in. Firstly, a reference sample of individuals
Value $x_{src}$ is mapped to a corresponding value on the target distribution, $x_{target}$, having the same empirical cumulative probability.

cannot be provided, as it would potentially compromise their privacy. In order to apply the population synthesis approaches therefore, one would have to synthesise a micro-sample, using some generative algorithm that would reproduce the joint distribution. Multiple histogram matching was identified from the image and signal processing literature (e.g. [Borgnat et al., 2012; Shapira et al., 2013]) as a potential candidate for synthesis, due to its capability of reshaping a ‘blob’ of data points in high-dimensional space to fit any joint distribution.

The method presented in this chapter relies on two techniques: histogram matching and Principal Component Analysis (PCA). A short overview of both techniques is presented here.

### 8.3.8 Histogram matching

Histogram matching is a process whereby values from a source empirical cumulative distribution is transformed to those of a target distribution, by mapping values with the same cumulative probability to each other, as is illustrated in Fig. 8.4.

Histogram matching has been successfully employed in a wide range of applications, but especially the fields of image processing and machine vision; for instance, to adjust the dynamic range of an image to that of a target or reference image in order for image recognition to be performed under varying lighting conditions. A Google Scholar search for the term “histogram matching” produces, at the time of writing, more than 350,000 results.
Applications where histogram matching is used in surrogate data synthesis are harder to find. The technique is used for texture synthesis, which is to create variations of a photographic texture based on the properties of a sample (e.g. Chen and Wang 2009). These can be useful in computer generated imagery applications. A single article outside of the texture synthesis domain was found where histogram matching is explicitly mentioned as a step in surrogate data synthesis, albeit in the generation of time series data with a relatively simple joint distribution in comparison with transportation data (Borgnat et al. 2012).

8.3.9 Principal Component Analysis (PCA)

PCA is a very popular method of reducing the dimensionality of high dimensional data sets, and is extensively documented in the literature (e.g. Jolliffe 2002). PCA can be summarised as rotating data in high dimensional space in such a way that the marginal distribution along each of the orthogonal axes of the transformed dataset has maximum variation compared to any other high dimensional rotation.

Informally, the process of deriving the principal components can be summarised as follows: the first principal component is a vector in high dimensional space along which variance is a maximum. The next principal component lies in the plane orthogonal to its predecessor, and is once again chosen to lie in the direction of maximum variance. By repeating the process in the plane orthogonal to each principal component vector, we reach a maximum number of orthogonal vectors equal to the number of dimensions of the original dataset. As each vector attempts to maximise variance, higher numbered principal components necessarily display less variance than their predecessors. The normalised vectors of principal components can be compiled into a rotation matrix; performing a matrix multiplication of a surrogate dataset of the same dimensions with this rotation matrix would orient the data in the same way as the principal components of the original dataset.

8.4 Design idea: iterative multiple histogram matching

Instead of matching only in one or two dimensions, the idea for this work is to use the technique to shape high-dimensional surrogate data until it looks like real transport data. The data is high-dimensional because, if one were to record information only for a single trip, six dimensions are required; X-Y coordinates of origin and destination, trip start time and duration. To record two-trip day tours, 12 dimensions are required (assuming there aren’t any guarantees that people always depart from the destination of their last trip). In the transit smart card data, it was found that more than 95% of card users had up to four trips in their day tour. Representing a four trip
Figure 8.5: Illustration of the iterative multiple histogram matching process against a two-dimensional target histogram.

Points in blue are to be transformed until their distribution is similar to those in orange. Density plots in the margins show the distribution of points for projections of both data sets. Blue points are initially randomly scattered in space (1); histogram matching along both axes produces image (2). Both data sets are rotated through an arbitrary angle by matrix multiplication (3), and the process is repeated to produce (4).

day tour as a single data point requires 24 dimensions.

In initial experiments to test the idea of using histogram matching for high-dimensional surrogate data synthesis, marginal histograms of a large number of rotations of the target dataset were constructed, and the rotation matrices along with associated marginal distributions were recorded. During reconstruction, an arbitrary dispersion of surrogate data points in high dimensional space was put through the same rotations as was recorded for the original dataset.
After every rotation, the marginal histograms of the surrogate data was matched to that of the original dataset. After many rotations and repeated adjustments, the surrogate dataset would be sufficiently similar to the original dataset that no point in the surrogate would move by an appreciable amount if subjected to any further adjustments.

However, this approach required an excess of information to be stored, as properties of the joint distribution would emerge only very gradually after a very large number of rotations. Especially the spatial arrangement of data points is very important for transport planning purposes; to reconstruct the detailed spatial patterns of data from distributions along arbitrary vectors in even only six-dimensional space would require thousands of distributions to be recorded.

### 8.4.1 Two-dimensional histogram matching

However, if, instead of marginal distributions, two-dimensional joint distributions could be used, then much less information needs to be stored. By choosing the appropriate bin sizes along each dimension in a two-dimensional histogram, one can guarantee that aggregation will not expose any single point in the original data. It is this realisation that has steered the work documented in this chapter towards using two-dimensional histograms as an input, making it possible to perform rotations of the histograms instead. Then one would simply match marginal distributions of the surrogate when rotated through the same angle as the 2D histogram, for each pairwise combination of variables.

This process is illustrated for the joint distribution of origin X and Y coordinates of transit smart card transactions in Fig. 8.5. In each of the steps illustrated, the target 2D histogram is displayed in orange, and the surrogate dataset is shown in blue.

The process starts with an arbitrary rotation of the 2D histogram and surrogate dataset. Marginal distributions are shown above and to the right in corresponding colours. Step (2) shows the result of histogram matching of both marginal distributions. The dataset and 2D histograms are then rotated again through an arbitrary angle, as shown in step(3). Histogram matching along both axes then produces step (4). If the process is repeated many more times, the surrogate data starts looking a lot like the original data set.

So, for all pairwise combinations of variables, an appropriate bin size is chosen along each dimension, and the number of observations in the original dataset falling into each combination of bins is recorded. In this way, detailed spatial patterns can be reproduced in the surrogate dataset without having to record a large number of rotations and their associated marginal histograms during encoding of the original.
8.4.2 Using PCA to improve restoration of joint distribution

Unfortunately, reproducing the pairwise joint distributions of all constituent variables does not adequately capture their joint distribution in higher dimensional space, in a similar way that performing histogram matching along the margins from a single angle for the two-dimensional distributions shown in Fig. 8.5 does not adequately reflect the true joint structure of the orange target data set, and produces only a very naïve reconstruction. We therefore require additional information on the joint distribution of all the variables in order to produce realistic origin destination pairs, with realistic trip start and end times. To this end, PCA was selected as a means of recording the maximum amount of information on the joint distribution in the most parsimonious way.

Recording the two-dimensional histograms of all combinations of principal components as in Fig. 8.1 carries a lot of information about the covariance of the original data. Two-dimensional histograms also make it possible to perform iterative histogram matching along all possible two-dimensional rotations of the principal components, further obviating the need to record many more rotations of the dataset along with its associated marginal distributions, as was explained before.

8.5 Method

The method consists of an encoding and synthesis step. Encoding is straightforward, a series of two-dimensional histograms are constructed for each pair of variables in the target dataset, given specifications of maximum resolution along each dimension. The process is repeated for its principal components.

Synthesis is more time consuming. Firstly, in order to speed up convergence, a surrogate data set is constructed in principal component space, through repeated conditional sampling. Then, iterative histogram matching against multiple random rotations in both principal component and original variable space is performed, using 2D-histograms such as those shown in Fig. 8.1 as target distributions.

The following subsections briefly describe the encoding and synthesis processes.

8.5.1 Encoding

1. Scale and centre the data along each dimension such that each variable has unit variance and zero mean. Record the scale and centre vectors for reconstruction.
2. Discretize each dimension to a suitable resolution, recording the range and resolution of each variable.

3. Record for each combination of now discretely-valued variables, its two-dimensional joint distribution as a 2D-histogram, as shown in the lower-left quadrant of Fig. 8.1.

4. Perform a PCA on the original data, recording the rotation matrix for the synthesis step.

5. Discretize each principal component to a suitable resolution.

6. Record for each combination of now discretely-valued principal components, its two-dimensional joint distribution as a 2D-histogram, as in the upper-right quadrant of Fig. 8.1.

8.5.2 Synthesis

Synthesis consists of two steps: producing a synthetic sample in joint distribution space, followed by iterative histogram matching to the previously recorded two-way joint distributions.

8.5.2.1 Producing a sample of synthetic data points in joint distribution space

The method of iterative histogram matching ultimately produces the same result even if we only start out with a sample of points that were uniformly sampled from the n-dimensional space bounded by each variable’s range. However, the method converges much faster if we start with a relatively plausible set of points in the high dimensional space.

Sampling from the principal components and transforming back into variable space is arguably a reasonable way to produce such a plausible set of points. However, as we do not have the full joint distribution of principal components, but only the two-way joint distributions, the best way to proceed is to perform a stepwise construction. The method can be summarised as follows:

1. Start by sampling from the first two principal components to produce a set of points at least equal in number to the number of trips or tours that we intend to produce (Note that the method therefore allows us to oversample trips or tours).

2. For each point produced in step 1, and all remaining \( n \) principal components, successively sample from each combination of two-way joint distributions conditional upon the last principal component value assigned to the data point. So, conditional upon its value for principal component \( x \in [1..n] \), or \( PC_x \), find the weights corresponding to that value.
for $PC_x$ in the plot of $PC_x$ vs $PC_{(x+1)}$, and perform weighted sampling for a value of $PC_{(x+1)}$.

3. In this way we produce a set of points that is contained within and bounded by the joint distribution of the original dataset.

4. The process ends by transforming the data points back into the original variable space, by performing a matrix multiplication with the transpose of the principal component rotation matrix.

### 8.5.2.2 Synthesis through iterative histogram matching

Matching two-dimensional histograms in principal component space ensures that all the important characteristics of the joint distribution of variables is captured. However, these data points will not have accurate pairwise distributions in the original variable space, and will appear somewhat blurred as can be seen in Fig. 8.6(a). This figure shows data points that were created in principal component space and fit against the two-dimensional histograms of all the pairwise combinations of principal components, and then transformed back into the original variable space. These points now need to be fit against two-dimensional histograms of all pairwise combinations of variables in the original variable space, in order to more finely match the spatial patterns in the observed data.

Once the process has been completed in original variable space, some of the points might have been nudged away significantly from the positions assigned to them during the fitting stage in principal component space. Therefore, the process of fitting in principal component space needs to be repeated in order to match all the pairwise distributions recorded. This will nudge points away again from the positions assigned in original variable space; however, when the data is transformed back into that space, the amount of blurring will be significantly less. By repeating this process of fitting, transforming, and fitting again, points ultimately reach a position in the joint distribution space where they manage to match all the pairwise joint distributions recorded for the original dataset, in both principal component and original variable space.

The process can be summarised as follows:

1. For a given pairwise combination of variables, whether in original variable or principal component space, perform a matrix multiplication that will rotate both the synthetic data and the 2D histogram through the same random angle.

2. Perform histogram matching along both axes of the rotated datasets, as illustrated in Fig. 8.5.
3. Transform the surrogate data set back to its original orientation by multiplying with the transpose of the rotation matrix from step 1.

4. Repeat steps 1 to 3 for an arbitrary number of iterations or until sufficient agreement with the 2D target histogram is achieved, for instance performing a similar two-dimensional aggregation of the synthetic data and comparing cell totals with the target 2D-histogram.

5. Repeat steps 1 to 4 for each available pairwise combination of variables.

6. Transform the synthetic dataset back into principal component space (or original variable space) by performing a matrix multiplication with the PCA rotation matrix (or its transpose).

7. Repeat steps 1 to 6 until sufficient agreement between all synthetic and target 2D-histograms is reached.

In the case of the smart card data, the data does not visibly change anymore after 50 iterations of steps 1 to 6 in the fitting process. It remains to find some universal measure of convergence for the process that could apply across different datasets of varying dimension; one idea is to follow the rate of change of the test statistic of a two-sample Kolmogorov-Smirnoff test of the distribution of the norm of all synthesised data points against that of the original data.

Sometimes it can happen that some points in the synthetic dataset remain at positions outside the two-way joint distribution for many iterations. In a post-processing step, these points may simply be removed these points from the synthesis if one is oversampling, or force them to the nearest point in the 2D histogram that has a non-zero value, using a nearest neighbour search. By repeating this post-processing several times during the fitting process the number of ‘stragglers’ will decrease with increasing number of iterations.

The procedures outlined here have been implemented in R [R Core Team, 2016], with the following libraries used for ease of implementation and speed: magrittr [Bache and Wickham, 2014], dplyr [Wickham and Francois, 2016], tidyr [Wickham, 2016] and data.table [Dowle et al., 2015].

8.6 Results

In this initial exploration, only trip-wise reconstruction was considered, in the interests of time, as higher dimensional data sets of entire tours still takes an inordinately long time to reconstruct with the prototype code. Therefore validation is done at the level of single trips only.
Figure 8.6: Comparison of spatial distribution of trip origins in surrogate data and smart card records.

Values are aggregated into counts within 500x500m cells, with the red end of the colour spectrum denoting relatively high counts, and blue denoting relatively low ones.

In the procedure set out in this chapter, the primary source of validation is always at hand; namely the two-way joint distributions of variables, both in principle component and original variable space. At the very least, therefore, once the process is complete, the surrogate dataset should produce exactly the same set of pairwise joint distributions that was used to match against during synthesis. Fig. 8.6 compare 2D histograms of trip origin coordinates for the surrogate against the original smart card data. Fig. 8.6(a) shows the surrogate before any histogram matching has been done, i.e. it is the result of the conditional sampling process in principal component space, transformed back to the original data orientation. While it is clear that surrogate data is concentrated approximately in the right region, it would take 50 iterations of IHM to produce a reasonable match with the distribution of the original data, at this resolution (500mx500m cells). Each iteration involved running matching operations for 8 random rotations of every pairwise combination of principal components and original variables, amounting to 50 x (15x8x2 + 15x8x2) = 24,000 histogram matching operations.
However, having secondary sources of validation helps measure the extent to which the true joint distribution of the target data source has been captured. A comparison of the distribution of speeds (based on Euclidean distance between trip origin and destination coordinates) of the synthesised dataset with that of the original smart card data, is a strong indicator of how well the joint distribution of the target data set has been reproduced, as its calculation involves interaction of all six of the variables recorded (distance between origin and destination coordinates, divided by the difference between end time and start time). This comparison is shown in Fig. 8.7, both before histogram matching was performed (i.e. the speed distribution produced by successive principal component sampling, shown in pink) and after 50 iterations (red line). It appears that most of the trips produced in the surrogate dataset are slightly faster than what would be expected for their corresponding percentile in the smart card data.

Another way of quantifying how well the joint distribution is reproduced, is to compare how trips are distributed to destinations in time and space, from a given set of origin coordinates. Fig. 8.8 compares trip destinations aggregated to 500m x 500m cells, departing from a selection of busy coordinates, for (a) the original smart card data and (b) the 50th iteration of the synthesised data. Colour denotes travel time, circle size denotes number of trips.

The surrogate manages to capture the overall spatial distribution of trips and their associated travel times reasonably well. Most of the bigger dots denoting popular destinations for this set
Figure 8.8: Comparison of patterns of trip destination counts and travel time in smart card data and 50th iteration of synthesis.

(a) Smart card data

(b) Synthesised data

Dots are spaced 500m apart in both dimensions. Trips depart from the cells inside the red rectangles. Colour denotes travel time (in seconds), circle size denotes number of trips.
of origins are reproduced, however on the west side of the island the match is not as good as for the north-south corridor. Furthermore, surrogate destinations are reported to be reachable in a shorter time than what the original data reported. This finding is consistent with the speed comparison plot in Fig. 8.7.

8.7 Conclusion and Outlook

The approach set out in the preceding sections has been shown to produce a good approximation of the original joint distribution of transit trips in time and space. The method affords data stewards control over the resolution at which they provide data to the public, guaranteeing anonymity. The approach also affords a very concise way of summarising high dimensional datasets, with the possibility of encoding the entire dataset into an image depicting its component variable pairwise joint distributions. Such an image can easily be transformed into the set of two-dimensional histograms that form the basis of the approach, if the range of each variable is known, along with the principal component rotation matrix, and the scale and centre vectors that were used to transform each variable in such a way that it has zero mean and unit variance. The initial experiment had the issue that trip speeds tend to be too fast in the surrogate when compared with the original data. This issue will have to be addressed in future work. A possible solution might be to have trip duration as an alternative to trip end time, as trip duration is currently derived by subtracting trip end time from trip start time.

8.7.1 ‘Painting’ scenarios

It is important to note that the approach will still produce a result even when the individual pairwise combinations are not consistent; therefore it becomes possible to ‘paint’ an agent-based demand using a graphical editing tool. An existing demand might therefore be modified by painting in new origin or destination locations. Therefore this approach could be a very accessible way of drawing up scenarios for agent-based transport simulations, or for any other modelling application relying on high dimensional disaggregate data.

8.7.2 Higher-dimensional reconstruction, performance

Initial tests with entire tours composed of up to 4 trips per person have shown promisingly good agreement with observation after a limited number of iterations. However, such higher dimensional datasets require far more time to achieve convergence along all dimensions, as the number of pairwise combinations increases dramatically with the number of variables. If
8.7.3 Alternative encoding method: grand tour sinograms

Instead of encoding the data in the form of two-dimensional histograms, an alternative approach would be to smoothly rotate the dataset through interesting projections, and continuously record the distribution of the data along the primary axis of the projection. The output of this process would be similar to a sinogram in computer aided tomography, but instead of recording the intensity of the scanning beam as it is rotated around a subject, we record the percentiles of the rotated data along its new primary axis.
The principle of this idea is illustrated in Fig. 8.9 for a two-dimensional dataset recording trip origins in the smart-card data. The dataset is rotated through 360° in 1° increments, and for each rotation the distribution along the new X axis is recorded. Each line in the figure represents a percentile for each of the 360 rotations.

For high dimensional data sets, one would construct a series of orthogonal rotation matrices, that will rotate the dataset smoothly through its principal components and in the planes of original variable combinations (such as 2D coordinate planes). One may also look for interesting projections along the way, for instance projections where the data is highly separated into clusters, or where it is very narrowly distributed. Such projections can carry a lot of information about the joint distribution.

This encoding has the further advantage of not requiring a discretisation of either the original or surrogate distributions. For any given rotation in the sinogram, one would simply

- multiply the surrogate with the same rotation matrix, which could be an interpolation of two adjoining matrices that serve as neighbouring points on the x-axis of the sinogram;
- use an interpolation technique (e.g. b-spline or Kriging) to draw a line that passes through the points where corresponding percentiles in the original distribution and the surrogate distributions occur for the primary axis of the rotated data;
- use this line as a function to re-map values along the primary axis of the rotation;
- repeat until points don’t change much anymore with further operations.

The capability to smoothly rotate through various interesting projections and interactively visualise 1D and 2D projections of those rotations, is already implemented in the R package, tourr (Wickham et al., 2011). The authors of the package developed this capability as an interesting way to explore high dimensional data sets. However, the proposition of continuously recording the projection distributions in a sinogram appears to be a novel idea, and will be developed in future work.

Finally, of course, once the method has been refined to the extent that the joint distribution is matched accurately enough that speed distributions etc. are near-indistinguishable from those in the original data, the surrogate dataset should be fed into the data-driven transit simulation, to see if comparable results are produced.
Chapter 9

Discussion and outlook

As suggested in the section on design science research methodology in Chapter 1, at some point the design research process has to be evaluated and the outcomes determined to be ‘good enough’; firm conclusions are written up and areas of future research are highlighted. In this chapter, conclusions from the previous chapters are summarised, and avenues for future research or improvement are highlighted.

Trajectory reconstruction  Chapter 3 covered the process whereby transit smart card transaction records were transformed into bus vehicle trajectories for the entire day. This process was originally conceived only to be able to visualise and analyse the performance of the actual transit system. It was only later on that we realised what the true value of this transformed dataset would be, when it was found that the performance of the transit simulation system was not adequate given the assumption of using the same normal distribution for all stop-to-stop speeds. The unprecedented predictive accuracy of the simulation system relies completely on the trajectory reconstruction process being a reasonable representation of how buses travelled during the course of the day. Therefore, any further advances that could be made should improve accuracy even further and obviate the need for multivariate outlier detection and other data cleaning processes further down the line.

Firstly, the cause for late tap-ins needs to be investigated, possibly resulting in a model being developed to predict the cause for a late tap-in observed in the transit smart card data. In this way the trajectory reconstruction would be able to better predict the actual departure time of the bus at this stop. Another worthwhile improvement would be to automatically predict the bus type from the smart card data using cluster analysis and machine learning. This would form a valuable input parameter to the speed regression models.
Speed regression modelling  A major conclusion from the speed regression modelling chapter was that some of the data points that had been identified as multivariate outliers might in fact be valid clusters of observations resulting from similar localised conditions. These clusters had not been observed before only the very late stages of the analysis process, mainly due to the extremely large number of observations making the dataset relatively unwieldy to view in its entirety using scatterplot matrices, parallel coordinate plots, principal component analyses, etc. Eyeballing only a sample of the data does not necessarily allow one to observe these clusters as there might simply be too few observations for it to register.

In future, therefore, it would make sense to first do some automatic cluster analysis at various thresholds. The task is then to verify each cluster on its own to see if the observations in the cluster are truly erroneous, arising from mistakes in the trajectory reconstruction process; or that the observations are in fact structurally distinct from the rest of the dataset; e.g. buses travelling on highways having substantially different dynamics from those operating within neighbourhoods.

This conclusion highlights an interesting new challenge arising from the sheer volume of data available to the transport analyst. As datasets become increasingly unwieldy and impossible to visualise in its disaggregate form at full scale, the analyst will have to rely on specialised data exploration techniques to gain a better understanding of the structural qualities of the data.

Furthermore, it would be interesting to see whether the inclusion of space syntax attributes improves the predictive accuracy of the speed regression models.

Transit simulation  The simplified transit simulation features several sites for potential improvement.

The simulation produces considerably fewer dwell operations where passengers are picked up compared to what is observed in the smart card data. From some basic calculations it appears that buses in reality pick up or drop off passengers 66% of the time, in comparison to the 40% observed in the simulation. This difference might be due to the best response routing in the simulation resulting in increased coordination between agents and buses, with agents selecting services that get them to their destination with less access waiting time on average than the service that they picked in reality. Agents might also not be as averse to crowding as people in reality, causing them to opt for the next empty vehicle less frequently; a hypothesis that will require further investigation into the ridership of vehicles in the simulation versus those in reality. For a start, it would be worthwhile to investigate if introducing a congestion term to the transit routing cost parameters significantly changes the number of dwell operations where passengers are taken on in the simulation; if the number changes as a result it is fair to assume that the current shortfall in the simulation is due to coordination between agents and particular
services that minimise their waiting time.

Furthermore, the simulation in its current state does not allow for any change in the arrival time of passengers at the stop. However, the expectation is that in reality, passengers would at least have some leeway to more or less adapt their arrival time in response to service frequency. Also, if one assumes that passengers can check the arrival time of buses using some smart phone application, it is essential that the simulation be able to capture the timing coordination that would take place. It would therefore be worthwhile to implement a limited activity departure time mutation strategy whereby passengers are allowed to change their departure time within a limited window from their originally assigned time.

Another shortcoming in the current implementation is that passengers are assigned a single route at the start of the mobility simulation. However, in reality, passengers might decide to take a bus operating on an overlapping route between origin and destination if that bus happens to arrive before their originally intended service. This is a form of dynamic replanning that might have a significant impact on simulation times as all passengers waiting at the stop would need to be evaluated whenever a bus arrives at that stop in order to see if the bus can drop them off at or near their intended destination.

**Peer-to-peer ensemble simulation** The distributed simulation framework is currently a master/slave architecture, where the master node passes travel time information to slaves and the slaves supply the master with mutated plans that have been run in a the QSim meta-model. The framework can easily be extended to have slave nodes execute queue simulations themselves; in which case the notion of master and slave is no longer useful. Instead, it becomes a peer-to-peer simulation, with plans constantly being exchanged between nodes, and multiple realisations of the simulation taking place. This configuration should arguably produce results similar to the ensemble simulations used in weather prediction, allowing one to quantify the stability of traffic patterns and travel times produced across simulations.

**Integration and automation** The preceding three chapters of this thesis stand somewhat separate from the rest of the text. What is still lacking is a scenario of the data-driven transport demand model that tests the suitability of the performance improvements presented in Chapter 6 and Chapter 7 with an agent-based demand derived through the data synthesis process described in Chapter 8. All indications are, however, that the methods presented in those chapters should be applicable to the data-driven simulation, but of course that needs to be extensively tested. However, as such integration exercises generally take substantially more time than usually expected, it had to be left for future work.

A bigger challenge would be to have most of the processes to prepare and run the transit simu-
lation automated in order to dramatically reduce the turnaround time for producing simulation scenarios. The trajectory reconstruction process and speech regression modelling is already relatively automatic; however, any transit simulation in MATSim still requires a substantial amount of manual work to associate bus route profiles with the road network. There are a semi-automatic map matching techniques such as the one by Ordoñez Medina and Erath (2011) that facilitates the task some extent.

What was perhaps not highlighted earlier in the thesis is that the various routing cost parameters used in the simulation came from the stated preference survey conducted by LTA 2008. These routing parameters appear to work quite well although the cost of accessing the MRT is still lacking. The automatic calibration of such costs would be useful. Having simulations run in parallel to explore the solution space, perhaps using a meta model such as the one presented in Chapter 7 could be very useful for scenarios where these cost parameters are completely unknown.

**Surrogate data synthesis**  Currently the iterative histogram matching technique only works with data that is continuous in all dimensions. It remains to be seen whether categorical variables can be realistically associated with synthetic data points using some machine learning classification models. Besides testing the synthetic data in the transit simulation it would also be interesting to see if the data can be used to estimate models to predict e.g. the frequency of trip-making given regional accessibility. Such a model would therefore provide an indication of how well the synthetic data reproduces the relationships that the real data has with exogenous variables of interest.
Bibliography


Horni, A., K. Nagel and K. W. Axhausen (2012b) High-Resolution Destination Choice in Agent-Based Demand Models.


Pieter Jacobus Fourie

Personal Information

Address: 1 CREATE Way
06-01 CREATE Tower
138602 Singapore
Phone: +65 9116 8440
E-mail: fourie@ivt.baug.ethz.ch
Nationality: South African
Date of birth: 11th of September 1974
Place of birth: Middelburg, South Africa

Education

Mar 11 – present
Doctoral studies, ETH Zurich, Switzerland
Dissertation title: Data-driven transit simulation
Advisor: Prof. Dr. Kay W. Axhausen

Jul 07 – Dec 09
University of Pretoria, South Africa
Dissertation title: An initial implementation of a multi-agent transport simulator for South Africa
Degree: MEng Industrial Engineering with distinction

Jul 05 – Dec 06
University of Pretoria, South Africa
Degree: BEng Hons (Industrial Engineering)

Jan 93 – Dec 98
University of Pretoria, South Africa
Degree: BEng (Metallurgical Engineering)

Working Experience

Mar 11 – present
Future Cities Laboratory, Singapore-ETH Centre for Global Environmental Sustainability (SEC), Singapore
PhD Researcher: Mobility and Transportation Planning

Jun 09 – Feb 11
Planning Support Systems (now Spatial Planning and Systems), CSIR Built Environment, Pretoria, South Africa
Researcher: integrated transport and land-use planning modelling.

Feb 07 – May 09
Dept. Industrial & Systems Engineering, University of Pretoria, South Africa
Lecturer, graduate researcher
Curriculum Vitae Pieter J. Fourie

1998 – 2009 Various locations in South Africa and Tanzania
Freelance broadcast video & multimedia producer

TEACHING EXPERIENCE

Jan 07 - Jun 08 Lecturer, 2nd year course “Industrial Analysis”, Dept. of Industrial and Systems Engineering, University of Pretoria.
Jan 07 - Dec 08 Supervision of final year projects, Dept. of Industrial and Systems Engineering, University of Pretoria.
Jan 99 - Dec 01 Lecturer, computer-generated animation, Boston Media House, Sandton, South Africa.

EXTRACURRICULAR ACTIVITIES

Mar 11 – present Contributions to the MATSim open-source project, see http://www.matsim.org.
Specific contributions include mobility simulation meta-modeling (org.matsim.contrib.pseudosimulation) and generating travel diaries from MATSim output (org.matsim.contrib.analysis.travelsummary).
MATSim project committee member.
Mar 11 – present Produced promotional/educational videos for projects at Future Cities Laboratory.

SKILLS

Modeling and simulation Agent-based simulation of co-evolutionary systems. Discrete choice modeling and simulation.
Programming Java, R, SQL, Python
Miscellany \LaTeX\ typesetting, Tableau business analytics
IT Linux system administration, version control in git and subversion, distributed computing on ETH Zurich Euler super-computer with LSF and OpenMPI, distributed computing with MIT StarCluster on amazon EC2, test-driven development
Video production Autodesk 3ds Max, Adobe After Effects, Adobe Premiere
Languages Afrikaans native
English native
German basic
BOOK CHAPTERS


ARTICLES


REFEREED CONFERENCE PAPERS


**Conference and Working Papers**


MASTER’S THESIS


PRESENTATIONS AND PUBLIC LECTURES


VIDEO PRODUCTIONS


SUPERVISED STUDENT THESES