Algorithmic and Implementation Aspects of On-line Calibration of Dynamic Traffic Assignment

by

Enyang Huang

B.E. Software Engineering
School of Computer Science and Engineering
The University of New South Wales - Sydney - Australia (2007)

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author .....
Department of Civil and Environmental Engineering
March 31, 2010

Certified by .....
Moshe E. Ben-Akiva
Edmund K. Turner Professor of Civil and Environmental Engineering
Thesis Supervisor

Accepted by .....
Daniele Veneziano
Chairman, Departmental Committee for Graduate Students

Algorithmic and Implementation Aspects of On-line Calibration of Dynamic Traffic Assignment

by

Enyang Huang

Submitted to the Department of Civil and Environmental Engineering
on March 31, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Civil and Environmental Engineering

Abstract

This thesis compares alternative and proposes new candidate algorithms for the on-line calibration of Dynamic Traffic Assignment (DTA).

The thesis presents two formulations to on-line calibration: 1) The classical state-space formulation and 2) The direct optimization formulation. Extended Kalman Filter (EKF) is presented and validated under the state-space formulation. Pattern Search (PS), Conjugate Gradient Method (CG) and Gradient Descent (GD) are presented and validated under the direct optimization formulation. The feasibility of the approach is demonstrated by showing superior accuracy performance over alternative DTA model with limited calibration capabilities.

Although numerically promising, the computational complexity of these base-line algorithms remain high and their application to large networks is still questionable. To address the issue of scalability, this thesis proposes novel extensions of the aforementioned GD and EKF algorithms. On the side of algorithmic advancement, the Partitioned Simultaneous Perturbation (PSP) method is proposed to overcome the computational burden associated with the Jacobian approximation within GD and EKF algorithms. PSP-GD and PSP-EKF prove to be capable of producing prediction results that are comparable to that of the GD and EKF, despite achieving speed performance that are orders of magnitude faster. On the side of algorithmic implementation, the computational burden of EKF and GD are distributed onto multiple processors. The feasibility and effectiveness of the Para-GD and Para-EKF algorithms are demonstrated and it is concluded that that distributed computing significantly increases the overall calibration speed.

Thesis Supervisor: Moshe E. Ben-Akiva
Title: Edmund K. Turner Professor of Civil and Environmental Engineering

Acknowledgments

Thank you to Professor Moshe Ben-Akiva. It has been a great privilege to be able to work with you at MIT. Your knowledge and insights on behavior-oriented traffic simulation have constantly inspired me in my research.

Thank you to Professor Constantinos Antoniou for your encouragement and patience. This thesis is as much a product of your guidance and advice as it is my effort.

Thank you to Dr. Ramachandran Balakrishna and Dr. Yang Wen for laying the foundation for DTA model calibration. Your ideas and work in this area have greatly inspired me. Thanks for your guidance.

Thank you to the MIT Schoettler Fellowship and the MIT-Portugal program for providing financial support for my graduate studies at MIT.

To my academic advisor, Professor Joseph Sussman, and to my program chair Professor Nigel Wilson, thank you for guiding my academic progress at MIT. To Dr. George Kocur, thanks for your advice and mentoring. To Kris Kipp and Jeanette Marchocki, thank you for your help and advice. To my friends and colleagues in Portugal, Jorge Lopes, Carlos Azevedo, Professor Francisco Camara Pereira and many others, it has been a great pleasure working with you. Our collaboration has achieved great things.

To my lab mates at the Intelligent Transportation Systems laboratory at MIT, Dr. Charisma Choudhury, Dr. Maya Abou-Zeid, Cristian Angelo Guevara, Lang Yang, Ying Zhu, Samiul Hasan, Zheng Wei, Sujith Rapolu, Anwar Ghauche, Hongliang Ma, Li Qu, Nale Zhao, Swapnil Shankar Rajiwade, Martin Milkovits as well as my friends from other laboratories, thank you for accompanying me in this journey. May our friendships last forever.

I would also like to mention a few people who have made my MIT experience possible. Dr. Bernhard Hengst, Dr. William Uther and Dr. Toby Walsh, thank you.

Finally, unbounded thanks to Tina Xue from Harvard University for perfecting this thesis. Any remaining errors are my own.

Contents

1 Introduction **17**

1.1 Thesis Motivation and Problem Statement 18

1.2 Sensory Technology State-Of-The-Art 20

1.3 Dynamic Traffic Assignment Framework 23

1.4 Literature Review 27

1.4.1 Calibration for off-line applications 27

1.4.2 Calibration for on-line applications 29

1.5 Thesis Contributions 31

1.6 Thesis Outline 32

2 Online Calibration Framework **33**

2.1 Overview of Framework 34

2.1.1 In-direct Measurement 34

2.1.2 Direct Measurement 36

2.1.3 System Inputs and Outputs 36

2.2 Problem Formulation 38

2.2.1 Notation 38

2.2.2 Measurement Equations 38

2.2.3 State-Space Formulation 39

2.2.4 Direct Optimization Formulation 41

2.3 Characteristics 42

2.3.1 Sensor Type 42

2.3.2 Highly Non-linear 43

2.3.3 Stochasticity 43
2.3.4 High Cost Function Evaluation 44
2.4 Summary 44

3 Solution Algorithms **47**
3.1 Introduction 48
3.2 Algorithm Selection Criteria 48
3.3 State-Space Formulation 50
3.3.1 Extended Kalman Filter Algorithm 50
3.4 Direct Optimization Formulation 52
3.4.1 Categorization on Minimization Algorithm 52
3.4.2 Hooke-Jeeves Pattern Search Algorithm 55
3.4.3 Conjugate Gradient Algorithm 57
3.4.4 Gradient Descent Algorithm 59
3.5 Summary 63

4 Scalable Framework Design **65**
4.1 Overview 66
4.2 Scalable Algorithm on Single Processor 66
4.2.1 The Idea of Simultaneous Perturbation 67
4.2.2 The PSP-EKF Algorithm 68
4.2.3 The PSP-GD Algorithm 70
4.3 Scalable Algorithm on Multiple Processors 72
4.3.1 Background 72
4.3.2 Architecture 74
4.3.3 Definition 75
4.3.4 The Para-EKF Algorithm 76
4.3.5 The Para-GD Algorithm 77
4.4 Summary 78

5 Case Study **79**

5.1 Objectives 80

5.2 Experiment Setup 80

5.2.1 Network and Sensors 80

5.2.2 DynaMIT-R 82

5.2.3 Parameters and variables 82

5.2.4 Experiment Design 83

5.3 Results 84

5.3.1 Offline Calibration 84

5.3.2 Base-line Algorithms Online Validation 86

5.3.3 Scalable Design Validation: PSP-GD and PSP-EKF 92

5.3.4 Scalable Design Validation: Para-GD and Para-EKF 97

5.4 Summary 101

6 Conclusion **103**

6.1 Summary and Findings 104

6.2 Thesis Contribution 105

6.3 Future Research 105

List of Figures

1-1 The overall structure of the real-time Dynamic Traffic Assignment system. 23

1-2 A detailed demand and supply interaction and information dissemination schema in DynaMIT - a state-of-the-art DTA System 26

2-1 The view of the real-time calibration framework from the input/output perspective. 37

4-1 The generic server-client implementation architecture of the Para-EKF and Para-GD algorithms. 74

5-1 The study network - Brisa A5. 81

5-2 The sensor deployments on the Brisa A5 study network 82

5-3 A 45 degree plot of estimated sensor counts against observed sensor counts during off-line calibration 85

5-4 A 45 degree plot of estimated sensor speeds against observed sensor speeds during off-line calibration 85

5-5 Comparison of traffic count observations between Dec-10-2009 and Jan-11-2010 87

5-6 Comparison of traffic speed observations between Dec-10-2009 and Jan-11-2010 88

5-7 Comparison of traffic count RMSN among base-line algorithms 90

5-8 Comparison of segment speed RMSN among base-line algorithms . . 90

5-9 Comparison of traffic count RMSN among GD, PSP-GD, EKF and PSP-EKF algorithms 94
5-10 Comparison of segment speed RMSN among GD, PSP-GD, EKF and PSP-EKF algorithms 94
5-11 Comparison of algorithms' scalability using the average number of function evaluations per state variable per estimation interval 95
5-12 Comparison of algorithms' trade off of scalability against accuracy . . 96
5-13 The speed performance comparison between EKF and GD in their distributed implementations 100

List of Tables

1.1 Classification of indicative traffic data collection technologies by scope. 21

1.2 Main types of data collected using different sensor type. 22

3.1 Classification of the four base-line candidate algorithms (EKF, GD, CG and PS) from framework design, use of derivative and iterative/direct 54

4.1 Distributed Extended Kalman Filter Algorithm - Para-EKF 76

4.2 Distributed Gradient Descent Algorithm - Para-GD 77

5.1 RMSN for the online validation of counts for the four candidate algorithms as well as improvement over the base algorithm 89

5.2 RMSN for the online validation for speeds for the four candidate algorithms as well as their improvment over the base algorithm 90

5.3 Algorithm running speed comparison for CG, PS, EKF and GD. . . . 91

5.4 Count RMSN Performance of PSP-GD and PSP-EKF algorithm with extended state unknowns 93

5.5 Speed RMSN Performance of PSP-GD and PSP-EKF algorithm with extended state unknowns 93

5.6 Computational statistics for distributed implementations of EKF and GD algorithms. Processor configurations of 2, 3 and 5 are compared. Each is repeated for 5 random seeds 99

List of Algorithms

1 The Extended Kalman Filter Algorithm 51
2 The Exploration Move Algorithm 56
3 The Hooke-Jeeves Pattern Search Algorithm 57
4 Non-Linear Conjugate Gradient Algorithm with Polak-Ribiere 58
5 The Gradient Descent Algorithm 60
6 The Bracketing Search Algorithm 61
7 The Brent's Parabolic Interpolation Algorithm 62

Chapter 1

Introduction

Contents

1.1 Thesis Motivation and Problem Statement 18

1.2 Sensory Technology State-Of-The-Art 20

1.3 Dynamic Traffic Assignment Framework 23

1.4 Literature Review 27

1.4.1 Calibration for off-line applications 27

1.4.2 Calibration for on-line applications 29

1.5 Thesis Contributions 31

1.6 Thesis Outline 32

1.1 Thesis Motivation and Problem Statement

Traffic congestion is an important topic in today's society. Not only does it impact the environment, road safety and urban development, it significantly affects the economy. [FHWA, 2001] reported that approximately 30% of daily trips in major US cities occurred within a congested environment in 1997. The economic impact totals $72 billion, which is approximately a 300% increase over 1982. In 2005, travel on U.S. highways amounted to nearly three trillion miles, which is 27.4 billion miles more than in 2004 and almost 25% more than in 1995 [FHWA, 2005].

While travel demand expands steadily, road network capacity has remained largely unchanged. Between 1980 and 1999, the total length of highways was only increased by 1.5 percent. The total vehicle-miles travel during the same period increased by 76 percent [FHWA, 2007]. Although the building of new road infrastructure and service networks proves to be an effective way of mitigating congestion, the cost is too high. To that end, much of the present research in transportation science has focused on the better management of existing road capacities.

Intelligent Transportation Systems (ITS) through surveillance systems, communications, and computing technologies, have the potential of facilitating effective management of transportation systems. Centralized among the many research directions under ITS is the development of high-fidelity Dynamic Traffic Assignment (DTA) models [Ben-Akiva et al., 1991][Ben-Akiva. et al., 2002]. These simulation-based DTA models are capable of replicating traffic network conditions and providing accurate forecasts as well as travel guidance. They benefit travelers through the avoidance of traffic congestion and the better planning of travel routes and departure time.

One of the key enabling factors for the deployment of such DTA system is the availability of an array of sensors that provides timely, accurate and reliable traffic information. Recent advancements of sensor technologies and their applications in surveillance systems have not only pioneered new methods of collecting and communicating network information, but also revolutionized the way traffic information is

utilized within modern DTA systems for enhanced accuracy and effectiveness. With advancements in technologies, information collected from network sensors can now be gathered in a timely manner for the calibration of DTA systems at their operational time, resulting in significantly improved modeling accuracy and reliability. This process of using available sensory information to adjust DTA models in real-time is known as DTA model *on-line calibration.*

The topic of DTA on-line calibration using real-time surveillance information has been widely discussed in literature. While accuracy and robustness are important aspects of DTA on-line calibration, the issue of scalability has received limited attention. For very large networks, the running time of these algorithms is a key determining factor for the successful deployment of these systems. This is because the on-line calibration of DTA systems has much more stringent operational constraints - the calibration procedure has to be completed within limited time intervals. As such, solution approaches that are not scalable for large networks tend to carry little value. While searching for fast candidate algorithms that are applicable for large networks is a top priority, accuracy and robustness cannot be neglected. However, in most cases, as speed improves, accuracy and performance deteriorate. The key is to strike a balance amongst the different objectives, developing *accurate, robust and efficient* candidate algorithms for the on-line calibration of DTA models.

This thesis explores algorithmic and implementation aspects of the established DTA on-line calibration framework from [Antoniou, 2004]. The algorithmic aspect of the on-line calibration entails the exploration of advancements in candidate algorithms - all else being equal, how one algorithm outperforms another in terms of accuracy and speed. The answer is not always clear-cut if only one or two methods are addressed. Often an examination of a spectrum of candidates is needed. Apart from algorithmic advancement, this thesis will also address efficient algorithm implementations. Good algorithmic implementations are just as important as the necessity for algorithmic accuracy and effectiveness. For example, a sequential algorithm may have its *parallel equivalent* implemented, should hardware resources permit. Although identical algorithms are operated, the parallel version of the algorithm may poten-

tially provide faster solutions. The two aspects are essentially complementary with each other, and when combined, have the potential to improve the performance of the existing DTA on-line calibration framework.

Before presenting the formulation, two important functional blocks within the DTA on-line calibration framework are first introduced. They are 1) Sensory Data and 2) DTA System.

1.2 Sensory Technology State-Of-The-Art

The important role of sensor technologies in on-line DTA calibration cannot be overstated. Given the objective of developing fast and effective approaches of on-line calibration algorithms, an abstraction of traffic sensor network that is based on their spatial characteristics and the attributes of the traffic data collected is necessary. In this section, a comprehensive overview of sensor technologies for traffic engineering is provided.

There are currently several technologies for traffic data collection. Each of these technologies has different technical characteristics and principles of operation, including types of data collected, accuracy of measurements, levels of maturity, feasibility and cost, and network coverage. [Antoniou et al., 2008] categorized sensor types based on functionality as point, point-to-point, and area wide:

Point sensors: This is the most basic type and the most widely used type of sensor used in traffic engineering. Examples of point sensors are inductive loop detectors and loop detector arrays [Oh et al., 2002]. With recent sensor technology advancements, new point sensors such as radar [Nooralahiyan et al., 1998], infrared and point video sensors [Pack et al., 2003] are being developed and deployed.

Point-to-point sensors: Point-to-point sensors can recognize vehicles at multiple points in a network. Some point-to-point sensors provide complete traversal of the travel path while others recognize vehicles at lesser locations. This type of sensor usually provides point-to-point travel time, count, aiding in route choice analysis and OD estimations. Typical sensors in this category are: Automated

Vehicle Identification (AVI) systems [Antoniou et al., 2004], Global Positioning System (GPS) [Quiroga et al., 2002], Cell-Phone tracking [Laborczi et al., 2002], License Plate recognition using Dedicated Short Range Communication (DSRC).

Area-wide technologies: These advanced technologies are still under heavy research. One example of area-wide traffic sensor is the unmanned helicopter. This helicopter is autonomous and is able to fly from its base to the investigated area. The helicopter can transmit back traffic information in the format of photogrammetric, video, sound recording, hazard detection etc. to the Traffic Management Center (TMC) [Srinivasan et al., 2004].

Table 1.1 classifies the various traffic data collection technologies by their spatial and vehicle coverage. Based on their coverage of the network, data collection systems can be organized by whether they can track vehicles throughout the entire network (e.g. GPS systems or cell phone tracking systems) or whether they are limited in identifying vehicles in particular locations of the network (e.g. tag identification sensors, or license plate recognition systems).

		Spatial coverage	
		Area-wide	Short-Range
Coverage	All vehicles	Airborne sensors	Loop detectors Radar/infrared/acoustic sensors CCTV (incl. license plate recognition)
	Equipped vehicles	GPS-based Cell phone tracking	Transponder detection

Table 1.1: Classification of indicative traffic data collection technologies by scope. Source: [Antoniou et al., 2008]

Table 1.2 summarizes the data collection capabilities of each sensor technology. A crucial conclusion, based on these two tables, is that no data collection technology is clearly superior, instead, their functionalities, in terms of data collected are complementary.

	Data collection technologies						
	Loop detectors	Radar/infrared/ acoustic sensors	CCTV cameras	License plate recognition	GPS/cell-phone tracking	AVI (1)	Airborne sensors
(Point) flows	X	X	X				
(Point) speeds	X		X				
Occupancies	X		X				
Subpath flows				X*	X	X*	X
Route choice fractions				X*	X	X*	X
OD flows				X*	X	X*	X
Travel times				X*	X	X*	X
Vehicle classification	X	X	X			X	X
Paths				(2)	X	(2)	X

Table 1.2: Main types of data collected by each sensor type. * Data limited by network design. (1) This technology could be used to collect practically any type of information collected by the vehicle (including speed profiles, origin, destination, and path). In this table, only the information that can be collected by "dumb" transponders, that simply report a unique vehicle signature, is reported. (2) Possible if a dense network of detectors was available. Source: [Antoniou et al., 2008]

1.3 Dynamic Traffic Assignment Framework

Dynamic Traffic Assignment (DTA) [Ben-Akiva. et al., 2002] systems combine sophisticated driver behavior modeling and traffic network simulation into a unified model. The system comes in two versions: 1) an off-line version of the system is used to evaluate various traffic management strategies such as lane management, incident management, capacity management as well as evacuation and rescue plans; 2) an on-line version of the system is usually used as a short-horizon prediction tool. The applications of the on-line DTA system include but are not limited to: real-time incident management, traffic control, emergency vehicle routing and traffic congestion forecasts etc.

For the purposes of this thesis, the real-time DTA system is of particular interest. The general structure of the model is given in figure 1-1.

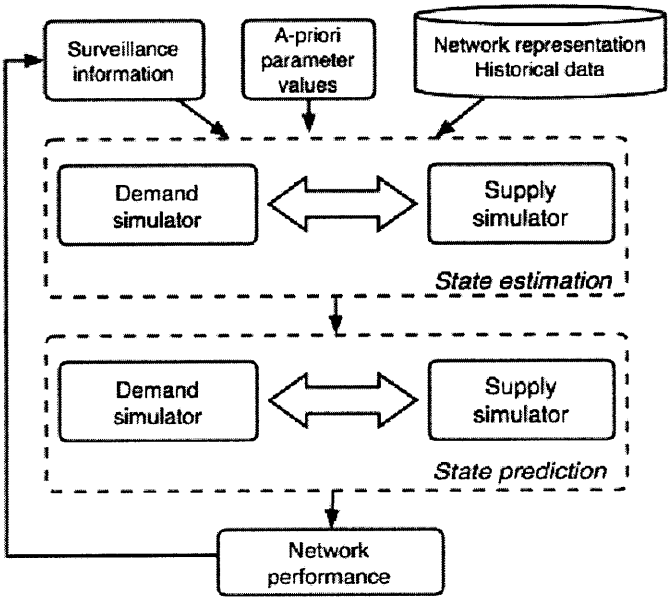

Figure 1-1: The overall structure of the real-time Dynamic Traffic Assignment system. It performs state estimation and state prediction. This involves a number of interactions of demand and supply simulation. Source [Antoniou, 2004]

The inputs of the system are sensory data from surveillance systems, a priori val-

ues of the unknown parameters in the DTA model, as well as historical data, such as time dependent OD flow matrices and the network representation. At the core of the system is an iterative process of state estimation and prediction. The purpose of state estimation is to ensure that the internal model's state is consistent with reality. Once state estimation is complete, state prediction is performed. Within state prediction, future network conditions are anticipated through Monte-Carlo simulations of drivers' behavior and traffic network. The outputs of the overall system are consistent forecasts of network conditions, including link density, flow, speed, as well as travelers' characteristics including their travel time, route choice and departure time. The anticipated information is used to generate guidance and will be incorporated into the next round of calculations [Bottom, 2000].

As can be seen, the most important component of the DTA system is the state estimation and prediction module. This module has two sub-components: 1) Demand simulation and 2) Supply simulation. A detailed description of the interactions of the demand and supply simulator can be found in [Ben-Akiva. et al., 2002].

The traditional approach to demand simulation is to perform real-time Origin-Destination demand estimation and prediction [Ashok and Ben-Akiva, 1993]. OD estimation combines historical and real-time information to obtain the best time-dependent OD matrices. OD prediction uses the current estimation of demand patterns and calculates short-term evolution of future demands [Antoniou, 1997].

The supply model is usually a meso-scopic traffic simulator. The choice of meso-scopic simulator is based on the trade-off between simulation detail and model running speed. Meso-scopic simulators allow relatively high resolution of network traffic modeling at low cost of computational burden. They consist of the following six major components. 1) Network representation; which represents both static (nodes, links etc.) and dynamic (free flow speed, max and min densities etc.) entities on the network. 2) Link output and acceptance capacity model; which simulates movement of vehicles on the boundary between two connecting links/roads. 3) Spill-back model; which models vehicle dynamics during congestion as well as the build up of queues. 4) Speed/density model; which models aggregated vehicle forward speed using road

segment's density information 5) Vehicle movement model; which considers the forward movement of vehicles on a link as well as their lateral selections of lanes. 6) Deterministic Queuing model; which models builds up and dissemination of queues, as well as vehicles' delay at intersections.

Figure 1.3 on the next page is a detailed schema of the demand and supply interaction and information dissemination for DynaMIT, a state-of-the-art DTA system.

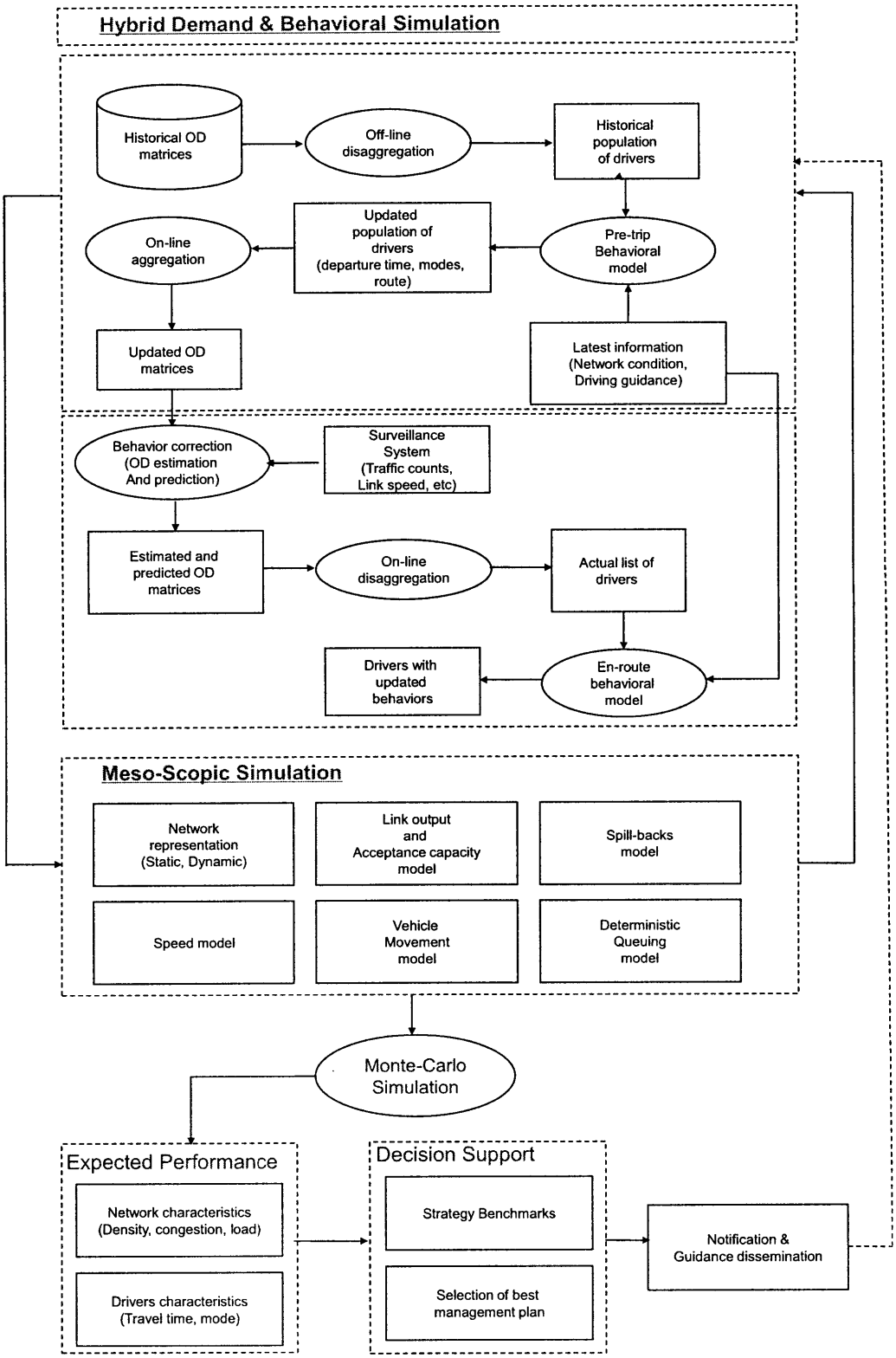

Figure 1-2: A detailed demand and supply interaction and information dissemination schema in DynaMIT - a state-of-the-art DTA System

1.4 Literature Review

The calibration problem in the domain of DTAs has been widely discussed in the literature of computational transportation science. From the operational point of view, existing literatures on DTA calibration can be grouped into two categories: off-line operation and on-line operation.

This section presents an overview of the calibration methodologies applied to the DTA framework. The first section presents the literature related to DTA off-line calibration with sensory information. The second section focuses on the methodologies for the on-line calibration approaches.

1.4.1 Calibration for off-line applications

[Balakrishna, 2002] studies the problem of off-line calibration using DTA models. In the study, multiple types of sensory data are used to calibrate the DTA model to produce accurate OD estimation and prediction. The objective is to minimize the simulated and observed quantities from the sensor data:

$$min_{\beta,\gamma,\chi_p}||M^{sim} - M^{obs}|| \tag{1.1}$$

where β represents the behavior choice parameters, γ represents the supply simulation parameters and χ_p are the OD flows departing from time interval p. M^{obs} are the sensor observations and M^{sim} are their corresponding simulated counter-parts. The objective function minimizes the discrepancy between simulated and observed sensory information from various types of sensors. These sensor types include: traffic flows, link speed and densities.

$$\chi_h = argmin\left\| N_1\left(\chi_h, \chi_h^a\right) + N_2\left(y_h^{sim}, y_h\right)\right\| \tag{1.2}$$

The OD estimation is conducted using equation 1.2. Function N_1 measures the discrepancy, in Euclidean distance, between the estimated χ_h and their a priori values χ_h^a. N_2 measures the discrepancy between the observed vehicle count from loop detec-

tor sensors y_h and their simulated counterpart y_h^{sim}. The calculation of this quantity is:

$$y_h^{sim} = \sum_{p=h-p'}^{h} a_h^p \chi_p \tag{1.3}$$

where p' is the degree of autoregressive process and a_h^p is an assignment matrix that maps OD flows at time interval h to links on the physical network. A rigorous treatment of assignment matrix can be found in [Ashok and Ben-Akiva, 1993].

The approach is later extended to calibrate both the OD flows as well as travelers' driving behavior parameters [Balakrishna, 2002]. In the study, sensor data are fused within a DTA model to calibrate time dependent OD flow matrices, the variance-covariance error matrix for measurement error, autoregressive parameters as well as driving behavior parameters. An iterative approach that jointly calibrates these parameters is proposed.

[Toledo et al., 2003] conducted off-line calibrations of OD flows and travel behavior parameters using microscopic simulations. In the study, aggregated sensor sources are used as inputs. GLS-based OD estimation is conducted to produce explicit constraints, and bi-level heuristic solution algorithms are used to conduct the overall estimation of OD and behavior parameters.

[van der Zijpp, 1987] combined volume counts with trajectory information obtained from automated license-plate surveys for the estimation of OD flows. A measurement equation for the trajectory counts is specified and split probabilities are estimated from combined link volume counts and trajectory counts.

[Balakrishna, 2006] proposed an innovative framework that jointly calibrates both demand simulation (OD flows, behavior parameters) and supply simulation (speed-density relationship, link capacities etc.) DynaMIT, a state-of-the-art DTA system is used to jointly calibrate these parameters in a case study. The study shows promising results in the estimating of unknown parameters from both demand and supply simulation components.

In contrast to equation 1.2 , the formulation jointly minimizes, with different

weights, three components:

$$argmin_{\chi,\beta}\left\| N_1(M,\mathcal{M}) + N_2(\chi,\chi^a) + N_3(\beta,\beta^a) \right\| \tag{1.4}$$

Where M is the observed sensor observation vector, $\mathcal{M}$ is their simulated counterpart from DynaMIT with $\mathcal{M} = f(\chi,\beta)$. χ are the estimated OD flows and χ^a is their a priori values. β is the vector of DTA model parameters including behaviors and supply parameters and β^a is the corresponding vector of their a priori values. N_1, N_2, N_3 are distance functions with adjustable weights.

The Simultaneous Perturbation and Stochastic Approximation (SPSA) [Spall, 1992] [Spall, 1994b] [Spall, 1994a] [Spall, 1998b] [Spall, 1998a] [Spall, 1999] method is found to be the most promising algorithm.

[Vaze et al., 2009] applied the SPSA method to address the off-line DTA calibration problem. In their study, sensor data involving vehicle count and Advanced Vehicle Identification (AVI) information are fused within the DTA framework. The formulation is similar to that of [Balakrishna, 2006]. DynaMIT is used as their candidate simulation software. The study found that calibration using AVI significantly increased the accuracy of the network traffic prediction.

1.4.2 Calibration for on-line applications

[Ashok and Ben-Akiva, 1993] [Ashok and Ben-Akiva, 2000] formulated the real-time OD estimation and prediction problem as a state-space model and solved it using a Kalman Filtering algorithm. The authors' use of deviations of OD flows from their historical values provides an elegant framework for incorporating structural OD information (generated during off-line calibration) into the on-line process [Antoniou, 1997]. [Ben-Akiva. et al., 2002] implemented this approach in the DynaMIT DTA system.

[Bierlaire and Crittin, 2004] outlined an efficient solution algorithm for the OD estimation problem. The method used is a generalization of the secant method, which uses several past iterations to produce linear approximation of the original non-linear form. The method is very efficient in that it is matrix free and derivative free.

[Antoniou et al., 2004] presented a methodology for the incorporation of AVI information into the OD estimation and prediction framework, which was later extended by [Antoniou et al., 2006] to allow for the consideration of any types of available surveillance data.

[Zhou and Mahamassani, 2006] developed a non-linear ordinary least-squares calibration model to combine and fuse AVI counts, link counts and historical demand information and solved this as an optimization problem.

[Antoniou, 2004] developed an approach that formulates the on-line calibration problem as a state-space model, comprised of transition and measurement equations. A priori values provide direct measurements of the unknown parameters (such as origin-destination flows, segment capacities and traffic dynamics model parameters), while surveillance information (for example, link counts, speeds and densities) is incorporated through indirect measurement equations. The state vector is defined in terms of deviations from the calibration parameters and inputs from available estimates.

[Antoniou et al., 2005] formulated the problem of on-line calibration of the speed-density relationship as a flexible state-space model and presented applicable solution approaches. Three of the solution approaches [Extended Kalman Filter (EKF), Iterated EKF, and Unscented Kalman Filter (UKF)] are implemented and applications of the methodology, using freeway sensor data from two networks in Europe and the U.S., are presented. The EKF provides the most straightforward solution to this problem, and achieves considerable improvements in estimation and prediction accuracy. The benefits obtained from the more computationally expensive Iterated EKF algorithm were shown. Applicable solution algorithms are presented and compared in [Antoniou et al., 2007a].

The most computational burdensome part of the EKF method is the calculation of Jacobian Matrix at each simulation interval. [Antoniou, 2004] further tested the Limiting Kalman Filtering algorithm in the context of on-line model calibration and sensor data fusion. The algorithm is able to achieve comparable performance as that of the EKF during relatively stable network conditions. The idea is to compute an

array of Kalman Gain matrix off-line, and select the most appropriate one. This approach saves a significant amount of computational time.

[Wang and Papageorgiou, 2005] presented a general approach to the real-time estimation of the complete traffic condition in freeway stretches. They used a stochastic macroscopic traffic flow model together a state-space formulation, which they solved using an Extended Kalman Filter. The formulation allows for the dynamic tracking of time-varying model parameters by including them as state variables to be estimated. Random walk is used as transition equations for the model parameters. A detailed case study of this methodology is presented in [Wang et al., 2007]. The approach is further validated in [Wang et al., 2008], where the joint estimation of supply parameters and traffic flow variables is shown to lead to significant advantages: 1) Avoidance of prior model calibration, 2) adaptation of traffic conditions and 3) enabling of incident alarms.

1.5 Thesis Contributions

This thesis makes several concrete contributions to the state-of-the-art, specifically,

1. The feasibility and effectiveness of the on-line calibration framework in a real-world application using both state-space and direct optimization formulations are demonstrated.

    - The framework is tested and successfully verified on a medium network with state variables exceeding 600. Previous on-line calibration studies used state size around 100.
    - Multiple sensory technologies are used in the tests, including Via-Verde and automatic Toll Gate Counter technologies. This demonstrates the flexibility of the framework that it does not impose any constraints on the types of sensors that it can handle.

2. A scalable framework design with a single processor: the PSP-EKF and PSP-GD algorithms are presented. The algorithms achieve high degree of accuracy

while maintaining low computational complexity.

3. A scalable framework design with multiple processors: Complete C++ implementation of the distributed version of the EKF (Para-EKF) and GD (Para-GD) algorithms within DynaMIT-R. A detailed computational performance analysis is presented along with the Brisa case study.

1.6 Thesis Outline

In the next chapter, we outline the formulations of the on-line calibration problem. The third chapter presents the algorithm selection criteria and illustrates four candidate algorithms - PS, CG, EKF and GD. Their characteristics and suitability for real-time DTA calibration are also discussed. Chapter four presents practical considerations aimed at framework scalability. Efficient variants of the EKF and GD algorithms are presented. In addition, distributed computing techniques are used in implementing the Para-EKF and Para-GD algorithms. Chapter five is a real-world case study that demonstrates the feasibility of the framework. The base-line algorithms and their scalable extensions are compared and analyzed for accuracy, robustness as well as efficiency. Chapter six concludes this thesis and presents brief future perspectives.

Chapter 2

Online Calibration Framework

Contents

2.1 Overview of Framework 34

2.1.1 In-direct Measurement 34

2.1.2 Direct Measurement 36

2.1.3 System Inputs and Outputs 36

2.2 Problem Formulation 38

2.2.1 Notation 38

2.2.2 Measurement Equations 38

2.2.3 State-Space Formulation 39

2.2.4 Direct Optimization Formulation 41

2.3 Characteristics 42

2.3.1 Sensor Type 42

2.3.2 Highly Non-linear 43

2.3.3 Stochasticity 43

2.3.4 High Cost Function Evaluation 44

2.4 Summary 44

2.1 Overview of Framework

We start by presenting an overview of the general concepts of the on-line calibration framework. The goal is to gather some insights on the intuitions of the methodologies behind the framework. The concept of direct and in-direct measurement is introduced and the inputs and outputs of the on-line calibration framework are summarized.

2.1.1 In-direct Measurement

Sensory information is an important ingredient of the on-line calibration framework - they inform the model about the *observed* network conditions. The information provided could include link speed, link density, OD travel time etc. Without loss of generality, the sensory data could come from off-line sources as well as on-line sources. In addition, they need not be from the same types of sensors.

On the other hand, simulation-based DTA models are designed to vividly reproduce network traffic conditions with high-fidelity. These DTA models are capable of producing their *simulated sensory value*, such as simulated loop detector counts, simulated link travel time, simulated segment speed, etc. Let M_h^{obs} be the sensory observation and M_h^{sim} be the simulated sensory value, the in-direct measurement equation is:

$$M_h^{obs} = M_h^{sim} + \epsilon_h^{obs} \tag{2.1}$$

The DTA models often comprise of a large number of parameters and inputs for adjustment. Depending on the values of these inputs and parameters, they produce predicted network conditions that vary in accuracy. *Calibration* is used to describe the process of adjusting the model's parameters so that it best reflects reality. In other words, the sensor data described above are used to *correct* simulated sensory values computed by the DTA simulators. The corrections take place when there are inconsistencies between the *observed sensor values* and their *simulated counter-parts.* Examples of such observed inconsistencies are:

- observed sensor link speed and simulated link speed
- observed sensor point to point travel time and simulated point-to-point travel time
- observed sensor vehicle counts and simulated vehicle counts at specific locations in a network

Such inconsistencies are resulted from a number of errors in the model. Two of the most common types of errors are: 1) Errors between the true model parameters and the existing model parameters, such as true link free flow speeds and the model's existing link free flow speeds. 2) Errors between true model input value and the actual value used, such as the true OD demand level and the actual demand level applied to the model.

In minimizing these errors, we are effectively adjusting the model parameters and OD demand levels so that the simulated sensory information of DTA models can be more consistent with observed real sensory data. Specifically, for each time interval h at operational time, we have:

$$min_{\pi_h} ||M_h^{sim} - M_h^{obs}|| \tag{2.2}$$

where π represents model parameters and the OD flows being calibrated. M_h^{obs} are the sensor observations and M_h^{sim} are their corresponding simulated counter-part. The objective function minimizes the discrepancy between simulated and observed sensory information from various types of sensors. These sensor types include: traffic flows, link speed and densities.

To construct the correction procedure, DTA models first need to be modified such that each sensor source has its simulated counter-part. This is done by first examining the sensors that are already deployed within a network, and then adding implementations of *virtual sensors* that are functionally identical to the deployed sensors. For example, if there is a sensor on a network that reports point vehicle counts, a *virtual sensor* at the corresponding location in the DTA model that reports

point vehicle counts, should be implemented.

2.1.2 Direct Measurement

In addition to in-direct measurement equations that bridge model parameters with observed sensory values, one can easily construct the so-called *direct measurement* equations, which describe references of the model parameters themselves. For example, let π_h be the model parameters and OD flows at time interval h, all the following may serve as its direct measurement, which we denote as π_h^a [Ashok, 1996],

$$\pi_h^a = \pi_h^H \tag{2.3}$$

$$= \hat{\pi}_{h-1} \tag{2.4}$$

$$= \sum_{p=h-q}^{h-1} F_h^p \hat{\pi}_p \tag{2.5}$$

$$= \pi_h^H + \sum_{p=h-q}^{h-1} F_h^p (\hat{\pi}_p - \pi_p^H) \tag{2.6}$$

The first and second forms of the a priori are the simplest. They represent the historical and the estimated values from the last time interval, respectively. The third form implies that the model parameters follow an autoregressive process of order q. With M model parameters, F_h^p is an M by M matrix of effects of π_p on π_h. The fourth form models the temporal relationship among the deviations in the model parameters by an autoregressive process of order q. The matrix F_h^p is an M by M matrix of effects of $\pi_p - \pi_p^H$ on $\pi_h - \pi_h^H$. It captures correlation over time among deviations which arise from unobserved factors that are correlated over time. Examples of these factors may include weather conditions, special events, construction or incidents.

2.1.3 System Inputs and Outputs

To summarize, the DTA online calibration framework requires several input components. First it needs sensory data. This includes data from different sensors that are located at different network locations. Second, the framework needs a high-fidelity

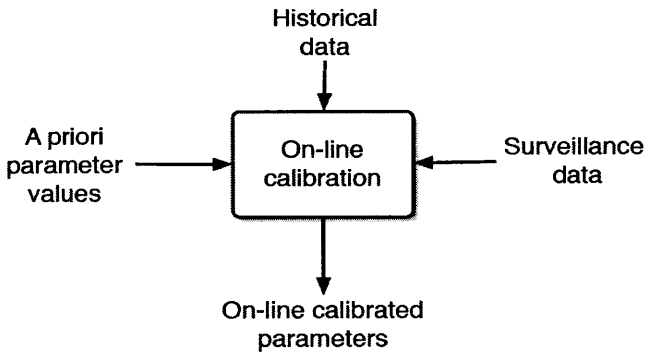

Figure 2-1: The view of the real-time calibration framework from the input/output perspective. Source: [Antoniou, 2004]

.

DTA model. The model needs to have the corresponding physical sensors implemented and be able to generate *simulated sensor data.* Third, the framework requires a set of a priori parameter values and a priori OD-flow matrices. Finally, the framework requires all other necessary information that is needed to operate the model, such as network representations.

The direct outputs of the system are the optimized model parameters (speed-density relationship, route choice, behavior choice etc.) and model inputs (OD demand matrices) [1]. These parameters and inputs are highly consistent with sensory observations and thus are used to generate anticipated network forecasts through using the model. In this way, the framework combines low-level sensor data and fuse them into a pool of consistent information for the network. The schema of the on-line calibration framework from the perspective of inputs and outputs is shown at figure 2-1. The framework takes a priori parameter values, historical data on OD flows, and real-time surveillance information and mingles them together using DTA to produce a set of consistent, realistic model parameters. The updated model parameters are then used to generate complete network traffic conditions through simulation.

[1]Existing on-line calibration studies usually include a subset of the model parameters and input variables.

Next, we will specify the core of on-line calibration - how it transforms the inputs and produces outputs. This may be accomplished via a number of approaches. For the purposes of this thesis, two different formulation strategies are presented. We present the classic *state-space* formulation as well as a *direct optimization* formulation.

2.2 Problem Formulation

In this section, the on-line calibration problem is formulated mathematically. The state-space formulation is identical to that of [Antoniou, 2004].

2.2.1 Notation

By dividing time period T into time interval $1, 2, 3, ..., N$ of size t, the network is represented as a directed graph with a set of links and nodes. Each link is composed of a set of segments. The network contains n_L links, n_G segments and n_{OD} OD pairs. Furthermore, each segment can be equipped multiple types of sensors. For example, n_g^{Count} segments are equipped with count sensors and n_g^{Speed} segments are equipped with speed sensors.

Let π_h denote the vector of OD flows and model parameters subject to on-line calibration at time interval h. ($\pi_h = \{\chi_h, \gamma_h\}$, χ_h is the OD flows at time h and χ_h is the model parameters at time h). Let S denote a DTA simulator and M_h^{obs} denote a vector of observed traffic conditions for time interval h (For example, travel time, segment flow counts). Let M_h^{sim} denote a vector of corresponding simulated traffic conditions from S.

2.2.2 Measurement Equations

Two types of information is used: 1) A priori values of the unknown parameter vector are used for the **direct measurement** and 2) Sensory observations of the network conditions are used for the **indirect measurement**.

Direct Measurement

The a priori values provide direct measurements of the unknown parameters. The off-line estimates π_h^a are commonly used as the a priori estimates.

$$\pi_h^a = \pi_h + \epsilon_h^a \tag{2.7}$$

Indirect Measurement

The indirect measurement is established through the measurement equation:

$$M_h^{sim} = S(\pi_h, \pi_{h-1}, ..., \pi_{h-p}) = S(\Pi_h) \tag{2.8}$$

Note that this formulation is flexible and poses no constraint on the types of sensory measurements it can handle. The formulation is general and it assumes that modeled trips can last for longer than one time interval and up to a maximum of p time interval length (Since model parameters for all p intervals may affect the simulated traffic condition). Π_h denotes the augmented parameter vector such that $\Pi_h = \pi_h, \pi_{h-1}, ..., \pi_{h-p}$. The relationship between the observed network condition and their simulated counter-part is:

$$M_h^{obs} = M_h^{sim} + \epsilon_h^{obs} \tag{2.9}$$

where ϵ_h^{obs} can be decomposed into three components: 1) ϵ_h^f 2) ϵ_h^s 3) ϵ_h^m and $\epsilon_h^{obs} = \epsilon_h^f + \epsilon_h^s + \epsilon_h^m$. ϵ_h^f captures the structural errors resulting from the inexactness of the simulation model. ϵ_h^s captures the numerical simulation errors, and ϵ_h^m captures the measurement errors. Since it is usually impossible to separate the three components, they are treated together.

2.2.3 State-Space Formulation

The classical approach to model dynamic systems is to use the state-space formulation. The state-space model consists of a transition equation and measurement

equations. The first step in defining the state-space formulation is the definition of *state*. For the on-line calibration of DTA systems, a state consists of model parameters and OD flows. Explicitly, let π_h denote the state for time interval h, so that:

$$\pi_h = [\chi_h, \gamma_h]^T \tag{2.10}$$

The transition equation captures the evolution of the state vector over time. The general formulation is that: $\pi_{h+1} = \tau(\pi_h, \pi_{h-1}, ..., \pi_{h-p}) + e_h^{auto}$, where τ is a function that describes the dependence of π_{h+1} in its previous p states. e_h^{auto} is a vector of random errors. In this context, an autoregressive function for τ is used. The transition equation is:

$$\pi_{h+1} = \sum_{q=h-p}^{h} F_q^{h+1} \pi_q + e_h^{auto} \tag{2.11}$$

The measurement equations are in two parts. The direct measurement equations capture the error between the state vector and its a priori values. That is:

$$\pi_h^a = \pi_h + e_h^a \tag{2.12}$$

The indirect measurement equation links the state vector with the sensory observations, explicitly, so that:

$$M_h^{obs} = S(\pi_h) + e_h^{obs} \tag{2.13}$$

The state-space model is now complete. [Ashok and Ben-Akiva, 1993] proposed to write the state-space model in its deviation form from its historical. The recommendation stems from two main reasons. Firstly, the deviation form implicitly incorporates the wealth of information contained from the offline calibrated parameters and OD flows. Secondly, the deviation form allows the normality assumption to hold for the error terms in the model. Without using the deviation form the state variables, such as the OD flows, will have a skewed distribution. Normality assumption is useful in the application of Kalman Filtering techniques. For the reasons,

[Ashok and Ben-Akiva, 1993] and [Antoniou, 2004] write: $\Delta\pi_h = \pi_h - \pi_h^H$, and our final state-space model in the deviation form is:

$$\Delta\pi_h^a = \Delta\pi_h + e_h^a \tag{2.14}$$

$$\Delta M_h = S(\pi_h^H + \Delta\pi_h) - M_h^H + e_h^{obs} \tag{2.15}$$

$$\Delta\pi_{h+1} = \sum_{q=h-p}^{h} F_q^{h+1}\Delta\pi_q + e_h^{auto} \tag{2.16}$$

2.2.4 Direct Optimization Formulation

A state-space equivalent formulation is the so-called direct optimization formulation. [Ashok, 1996] discussed the connection between the state-space and the Generalized Least Square (GLS) direct minimization formulations in general, and, Kalman Filter and least square estimation as their respective solutions in specific. It was argued that the application of the results obtained by the classical GLS to discrete stochastic linear processes leads to precisely the Kalman Filter. Therefore an alternative way of formulating the on-line calibration framework is to adopt the GLS direct optimization. In the direct optimization formulation, we jointly minimize the three errors from the three state-space equations: e_h^a, e_h^{obs} and e_h^{auto}. Because the historicals cancel out when we subtract each equations for the error terms, the direct optimization can be formulated as:

$$\pi_h^* = argmin[N_1(e_h^a) + N_2(e_h^{obs}) + N_3(e_h^{auto})] \tag{2.17}$$

$$= argmin[N_1(\pi_h - \pi_h^a) + N_2(M_h^{sim} - M_h^{obs}) + N_3(\pi_h - \sum_{q=h-p-1}^{h-1} F_q^{h+1}\pi_q)] \tag{2.18}$$

If we assume that e_h^a, e_h^{obs} and e_h^{auto} are normally distributed and uncorrelated, and, N_i signifies *squared distance*, the above can be translated into the following

generalized least square (GLS) formulation:

$$\begin{aligned}\pi_h^* = argmin[&(\pi_h - \pi_h^a)'V^{-1}(\pi_h - \pi_h^a)+ \\ &(M_h^{sim} - M_h^{obs})'W^{-1}(M_h^{sim} - M_h^{obs})+ \\ &(\pi_h - \sum_{q=h-p-1}^{h-1} F_q^{h+1}\pi_q)'Z^{-1}(\pi_h - \sum_{q=h-p-1}^{h-1} F_q^{h+1}\pi_q)]\end{aligned} \tag{2.19}$$

Where, V, W and Z are the variance-covariance matrix of e_h^a, e_h^{obs} and e_h^{auto} respectively.

2.3 Characteristics

Before presenting the choice of solution algorithms, this section provides a glance of the key characteristics of the on-line calibration approach 1) The sensory types that it handles, 2) The speed complexity of single evaluations of the loss function, 3) Stochasticity, 4) High non-linearity within the DTA model.

2.3.1 Sensor Type

The formulation of on-line calibration in the state estimation is general and flexible. It does not impose any restrictions on the types of sensors that it can handle. This is one of the primary advantages of simulation-based DTA systems. Equation 2.8 calculates the simulated counter-parts of the corresponding sensor values in the simulated network. The formulation has no requirements on the specifications of the surveillance systems. To construct the correction procedure, DTA models first need to be modified such that each sensor source has its simulated counter-part. This is done by first examining the sensors that are already deployed within a network, and then adding implementations of virtual sensors that are functionally identical to these deployed sensors within the DTA model. These virtual sensors are capable of reporting identical quantities in the simulated network. For example, if there is a sensor on a network that logs point vehicle counts, one shall add an implementation

of a virtual sensor at the corresponding location in the DTA model that reports point vehicle counts.

2.3.2 Highly Non-linear

The second characteristic of the approach is the highly non-linear relationship between the state variables and surveillance information. This can be demonstrated through the speed-density relationship model. The model calculates link speed (A common form of surveillance data) from simulated link density. The following equation is used:

$$v = v_{max}\left(1 - \left(\frac{k - k_{min}}{k_{jam}}\right)^{\beta}\right)^{\alpha} \tag{2.20}$$

Where the parameters subject to real-time calibration are 1) v_{max}: the maximal link speed, 2) k_{min}: the minimal link density, 3) k_{jam}: the maximal link density at jam condition, 4) α and β: the segment-specific coefficients.

The inherent non-linearity within the DTA model that the loss function contains complicates the solving of the problem. [Balakrishna, 2006] summarized the complications: 1) The loss function might have several local minimal rather than a unique minima, and searching for the global minima is often very difficult 2) Studying the true shape of the loss function becomes impractical, because of the relationships between observations and the large number of model parameters. 3) The interactions of various models further complicate the feasibility of applications of analytical representations.

2.3.3 Stochasticity

The DTA system is also stochastic. The system simulates a large set of vehicles and drivers' behaviors. Thus a certain level of stochasticity is necessary to reflect real world decision making. For example, the pre-trip behavior model applies probabilistic modeling of driver's choice of departure time and routes. The supply simulator processes vehicles in the network in a certain randomized order at each simulation interval. This helps to replicate real-world network dynamics.

The stochasticity in the underlying DTA model produces noise in the simulated traffic conditions. This might impact the quality of the function and derivative evaluations. Empirical analysis has shown small degrees of its affect on simulation performance. The two most successful calibration algorithms in literature, SPSA for off-line calibration and EKF for on-line calibration, both calculate numerical derivatives in the stochastic setting [Balakrishna, 2006][Antoniou, 2004].

2.3.4 High Cost Function Evaluation

Finally, the computational complexity of the DTA simulation model is high - a single evaluation of the loss function entails exactly one running cycle of the DTA simulator for the designated time interval. The running time of the DTA simulation model mainly depends on the driver populations in the model and the network complexity. [Wen, 2009] reported that the overall complexity of the DynaMIT supply simulator for a time interval is $O(Q \times t)$, where Q is the average number of vehicles in the network and t is the length of the time interval. As the cost of single function evaluation is high, algorithms that perform less evaluations of the loss function are preferred.

2.4 Summary

In this section, the DTA model on-line calibration framework is introduced and discussed in detail. The general approach is laid. That is, with the use of a simulation-based DTA system, the framework seeks to minimize 1) The difference between the model parameters and their a priori values and 2) The difference between observed sensory data vector and its simulated counter-part. The inputs of the framework are various surveillance data, a priori time-dependent OD demand as well as a priori model parameters. The outputs of the framework are consistent demand and supply parameter estimates, which the DTA model translates to high-level knowledge of network conditions and traveler benchmarks. The a priori of the OD and model parameters are used to provide direct measurements of the estimates, while the sensor data received in real-time provides in-direct measurements through the DTA model.

The transition equation on the variable state is specified using an autoregressive process.

The overall approach is mathematically formulated. The state-space formulation models the problem as a dynamic system that evolves over time. It consists of a transition equation, a direct measurement equation and an in-direct measurement equation. Another approach is to formulate the problem as a direct optimization problem. The loss function is composed of the sum of three components corresponding to the three equations in the state-space model: 1) discrepancy between observed and simulated sensor values, 2) discrepancy between model inputs and parameters and their a prior values, 3) discrepancy between model inputs and parameters and their projected estimates from the autoregrssive process. The variables in the scope are time dependent OD levels and appropriate set of model parameters. All elements are constrained within their feasible regions.

With high-fidelity DTA models, the framework is flexible enough to handle various types of surveillance systems. However, complex DTA models are inherently non-linear, making it difficult to come up with a precise image of the loss function. In addition, the solution to the minimization is not unique, since there might be many local minimal. Finally, the level of simulation details dictates longer model running time, making evaluations of the loss function costly.

Chapter 3

Solution Algorithms

Contents

3.1 Introduction **48**

3.2 Algorithm Selection Criteria **48**

3.3 State-Space Formulation **50**

3.3.1 Extended Kalman Filter Algorithm 50

3.4 Direct Optimization Formulation **52**

3.4.1 Categorization on Minimization Algorithm 52

3.4.2 Hooke-Jeeves Pattern Search Algorithm 55

3.4.3 Conjugate Gradient Algorithm 57

3.4.4 Gradient Descent Algorithm 59

3.5 Summary **63**

3.1 Introduction

In the previous section, the DTA on-line calibration problem is formulated. A loss function for minimization is defined. The elements subject to adjustment are set of time dependent OD demand matrices and DTA model parameters. The characteristics associated with the minimization problem are also discussed - non-linearity, stochasticity and costly function evaluations. With the discussion of calibration formulations, their characteristics as well as the DTA models, we are able to introduce the base-line solution algorithms that one can apply to the on-line DTA model calibration problems.

First the selection criteria for candidate algorithms are presented. Next we present a number of solution algorithms that correspond to the two formulations from the previous chapter. With the state-space formulation, we present the classical approach, Extended Kalman Filter. Since there lacks literature on the direct optimization formulation, we present three direct optimization algorithms with each representing a distinct categorization in function minimization. The direct minimization algorithms we present are: Pattern Search algorithm, Conjugate Gradient algorithm and Gradient Descent algorithm. Their suitabilities for on-line calibration are discussed, especially from the speed complexity perspective.

3.2 Algorithm Selection Criteria

DTA on-line calibration dictates **accurate, robust and efficient** candidate algorithms. Accuracy is the most important factor. Accuracy entails the capability of taking an array of sensory data to generate highly consistent predictions of future network condition and traveler statistics. With the mathematical formulation, this translates to achieving loss function with values that are as low as possible. However, low value of the loss function value for a particular time interval is not sufficient to justify the algorithm's accuracy. The set of optimized OD level and model parameters need to be verified through their effectiveness on prediction. Hence the requirement

of high degree accuracy must not only be satisfied at *state estimation* but also in *state predictions.*

A number of performance indicators are being developed. [Toledo et al., 2003] uses the Normalized Root Mean Square Error (RMSN).

$$RMSN = \frac{\sqrt{N \sum_{n=1}^{N} \left(Y_n^{sim} - Y_n^{obs} \right)}}{\sum_{n=1}^{N} Y_n^{obs}} \tag{3.1}$$

Where N is the number of observations during an experiment. Y_n^{obs} is the nth observation value and Y_n^{sim} is the nth simulated value.

[Pindyck and Rubinfeld, 1997] uses a number of other performance measures - RMSPE, RMSE and MPE.

$$RMSPE = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left[\frac{Y_n^{obs} - Y_n^{sim}}{Y_n^{obs}} \right]^2} \tag{3.2}$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{n=1}^{N} \left[Y_n^{obs} - Y_n^{sim} \right]^2} \tag{3.3}$$

$$MPE = \frac{1}{N} \sum_{n=1}^{N} \left[\frac{Y_n^{obs} - Y_n^{sim}}{Y_n^{obs}} \right] \tag{3.4}$$

Besides accuracy, the solution algorithms need also be robust. Robustness refers to the ability of maintaining high degree of accuracy over a range of different network conditions (quickly changing set of sensory data). Robustness is important because algorithms that achieve low loss function values under some network conditions but not others are not suitable candidates for real-world deployment. The testing of the algorithms' stable performance is discussed and presented in the case study.

Finally, the candidate algorithms need to be efficient. The on-line nature dictates that the proposed framework runs faster than real-time and still leave some time for prediction and route guidance generation. Algorithms that run at a very slow pace, e.g. in the order of hours, have no value for on-line deployment. The requirement

on speed is especially critical given that a single evaluation of the loss function is expensive, which is a common situation when simulating sufficiently large networks. The preferred candidate algorithms would thus require as little function evaluations as possible. A rough measure of algorithmic efficiency is the average function evaluations required for one state variable. This ratio is defined as:

$$ratio = \frac{\#Func}{N} \tag{3.5}$$

Where #Funcs refers to the Total number of functional evaluations for one interval and N refers to the Total number of model parameters to calibrate. The ratio should be as low as possible.

One limitation of using a single ratio like the aforementioned one is due to the structural difference among candidate algorithms (e.g. some methods require a single iteration and others require many iterations of its main procedure). For example, an algorithm that decays fast but has long running time tail tend to give a higher ratio compared to algorithms that decay slow and use less time, and one need to draw the complete decay diagrams when comparing alternatives. In this study, we will take a simplified approach. The running time of candidate algorithms will be limited to a fixed portion of the DTA system's operation interval (e.g. if DTA runs on a 10min interval, candidate algorithms can be limited to complete to 5min). We will then use this ratio to evaluate efficiency with this timing constraint enforced. The ratio thus gives a general flavor of candidates' running speed as well as their accuracy-speed trade-offs.

3.3 State-Space Formulation

3.3.1 Extended Kalman Filter Algorithm

The Extended Kalman Filter has been proposed as the base-line solution to the on-line calibration problem [Antoniou, 2004]. The Extended Kalman Filter recursively

solves the following state-space model:

$$X_{h+1} = F_h X_h + w_h \tag{3.6}$$

$$Y_h = H_h(X_h) + u_h \tag{3.7}$$

where w_h and u_h are assumed to be white. Equation 3.6 is referred to as the prediction step of the state vector X_h through function F_h, and 3.7 relates observation Y_h to state vector X_h through measurement function H_h.

Algorithm 1 The Extended Kalman Filter Algorithm

Initialization

$$X_{0|0} = X_0 \tag{3.8}$$

$$P_{0|0} = P_0 \tag{3.9}$$

for h = 1 to N do

Time update

$$X_{h|h-1} = F_{h-1} X_{h-1|h-1} \tag{3.10}$$

$$P_{h|h-1} = F_{h-1} P_{h-1|h-1} F_{h-1}^T + Q_h \tag{3.11}$$

Taking partial derivative

$$H_h = \left. \frac{\partial f(X^*)}{\partial X^*} \right|_{X^* = X_{h|h-1}} \tag{3.12}$$

Measurement update

$$G_h = P_{h|h-1} H_h^T \left(H_h P_{h|h-1} H_h^T + R_h \right)^{-1} \tag{3.13}$$

$$X_{h|h} = X_{h|h-1} + G_h \left[Y_h - h(X_{h|h-1}) \right] \tag{3.14}$$

$$P_{h|h} = P_{h|h-1} - G_h H_h P_{h|h-1} \tag{3.15}$$

End for loop

The EKF algorithm starts by initializing state vector X and the variance-covariance

matrix P. This is followed by a loop indexed on the time interval h during the DTA model simulation.

At each iteration, a number of tasks are performed in sequence. The time update models the transition process of the state vector, as in 3.10 and the variance-covariance matrix, as in equation 3.11. This is followed by the numerical approximation of the Jacobian Matrix for time interval h, computed in equation 3.12. Then the Kalman Gain G_h is calculated in equation 3.13. The state vector, as computed in equation 3.14, is thus the sum of two components - the vector output from time update (its time evolution) and the difference between sensory observation Y_h and its simulated counter-part $h(X_{h|h-1})$, but magnified by the quantity of the Kalman Gain. Finally, the variance-covariance matrix is updated as in equation 3.15.

Notes

The EKF algorithm is very efficient in terms of numbers of function evaluations. The computational burden of EKF comes in two folds: 1) The evaluation of the Jacobian at 3.12 and a matrix inversion at 3.13. With state vector of N unknowns, 3.12 requires at least N+1 function evaluations. With a moderate N, cost of 3.13 is small. The algorithm requires storage of the state vector and variance-covariance matrix P. 2) Matrix inversion.

3.4 Direct Optimization Formulation

In the followings, we present candidate algorithms to solve the direct optimization formulation of the on-line calibration problem. We will state our motivation for the candidacy selection first, since there are a large numbers of minimization algorithms that can be potentially tested.

3.4.1 Categorization on Minimization Algorithm

The direct optimization formulation allows us to cast of the DTA on-line calibration as a stochastic minimization problem (at equation 2.17). With the statement of the

normality and uncorrelated assumption, we can solve it by minimizing function 2.19. Function minimization is associated with a large area of numerical research. Over the years, many established algorithms have been invented. The algorithms have different characteristics in speed and memory complexity and impose different requirements on the loss function. For example, some algorithms require the derivative calculation while others only requires explicit function evaluations. Moreover, algorithms expose structural difference. The plain EKF algorithm is a *direct method*, meaning that it requires a fixed, finite number of operations to solve the problem. In contrast, many minimization algorithms are *iterative, opposite to direct*, meaning they attempt to minimize the loss function through successive iterations starting from the initial guess. The algorithm may be stopped prematurely and would still be able to report a complete solution.

The solution algorithms under the direct optimization formulation would thus be selected to represent these categories: 1) Derivative vs Derivative-free; and 2) Direct vs Iterative. Before presenting the algorithms, we first show how these categorizations impose implications on the performance of the corresponding solution algorithms.

Derivative vs Derivative-free

Not all minimization algorithms require explicit computation of derivatives of the loss function. However, algorithms that require derivatives tend to be more powerful, but sometimes are not powerful enough to justify the cost of additional effort in calculating the derivatives [Press et al., 2007]. Sometimes an analytical expression exists for the first derivative of the loss function. Sometimes such an analytical expression does not exist because of the complexity of the function shape. When no analytical formula exist and it is still necessary to use derivative minimization algorithms, numerical approximations of the derivative may be used [Antoniou, 2004]. The EKF, GD and CG are all derivative method. That is, their routines involve the derivative calculations. The PS algorithm, on the other hand, is entirely derivative-free. The drawback of the derivative method in the context of DTA on-line calibration is its intensive computational complexity. Without analytical formulation of the

derivative, the numerical derivative approximation itself demands many numbers of function evaluations.

Direct Method vs Iterative Method

The EKF and GD(0)[1] algorithms are direct methods. They use a finite number of steps to perform the minimization task and there will be no intermediate results made available unless the algorithms come to their completions. On the other hand, the CG and PS typically require many iterations of their main routines. They usually come to completion when the objective function converges or the step progress drops below a pre-specified threshold.

As CG and PS are iterative, they tend to be more computationally intensive, for many function evaluations are required. However, iterative methods hold an important advantage - they can return the best result found at any given point during the iteration. On the other hand, the EKF and GD cannot produce any results unless a full run of the algorithms are completed.

To illustrate the categorization of the candidate algorithms, we tabulate the four candidate algorithms in table 3.1.

	EKF	GD	CG	PS
State-Space Model	Δ			
Direct Optimization Model		Δ	Δ	Δ
Use Derivative	Δ	Δ	Δ	
Derivative Free				Δ
Iterative Method			Δ	Δ
Non-iterative Method	Δ	Δ		

Table 3.1: Classification of the four base-line candidate algorithms (EKF, GD, CG and PS) from framework design, use of derivative and iterative/direct

In the next, we shall present the PS, CG and GD algorithms in detail.

[1]GD algorithm that evaluates gradient only once then performs a single line search and return. In this thesis GD refers to GD(0).

3.4.2 Hooke-Jeeves Pattern Search Algorithm

The Hooke-Jeeves pattern search method, [Hooke and Jeeves, 1961] only requires function evaluations and requires no derivatives. The method is composed of a sequence of *exploration moves* and *pattern moves.* The goal of *exploration move* is to find the next lowest point at the vicinity of the current point. In doing so, the method perturbs each variable in turn. The perturbation that results in the lowest function value is recorded and used in the pattern move phase. During *pattern move*, the current point is moved towards the point outputted by the exploration step until further sliding along the direction does not reduce the function value anymore. In that case, the perturbation magnitude is reduced and the process starts again. The algorithm is given below:

Exploration Move

The exploration move takes an input point as the start and perturbation factor and outputs the point with the lowest loss function value at the locality of the initial point.

Pattern Search Algorithm

At each step, the Hooke-Jeeves method works as follows: First the method searches for the best point in the locality of the present point P^{base}. If no better point is found then the perturbation step size is reduced according to α. If the step sizes are sufficiently small, then the algorithm quits and returns the present point. If an improved point is found, denoting that exploration point as P^{exp}, then pattern movement is performed. Denoting the resulting point $P^{pattern}$ from this movement, an additional exploration search is conducted. The pattern moves continues until no more improvements can be made. In that case it reduces the perturbation factor and starts a new iteration of exploration move with the reduced perturbation factor.

Algorithm 2 The Exploration Move Algorithm

Inputs

Initial point P of dimension N

Perturbation Matrix A

$$A_{N,N} = \begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_N \end{pmatrix}$$

for i = 1 to N do

Perturb

$$\begin{aligned} P_i^{low} &= P - A[i] \\ P_i^{high} &= P + A[i] \end{aligned} \tag{3.16}$$

$$P^{best} \Big| f(P^{best}) = minimal\Big(f(P_i^{low}), f(P^{best}), f(P_i^{high})\Big) \tag{3.17}$$

End for loop

Return P^{best}

Notes

The search algorithm is extremely simple and intuitive. It has the nice property of extremely low memory requirement. At any time only a constant number of points need to be stored. The biggest drawback of this algorithm is its running speed with larger number of variables to minimize. The exploration algorithm requires exactly O(2N) times evaluations of the loss function to recommend a new point to the pattern search phase. In addition, perturbations along every dimension are repeated at each iteration of the algorithm that no knowledge of the perturbations from previous is re-used. These two observations, especially the exhaustive search in a point's vicinity makes Hooke-Jeeves pattern search less attractive for on-line calibration applications with large amount of DTA model parameters and OD pairs. Despite the disadvantages on the speed, the method is usually very reliable. This will be discussed again in the experiment chapter.

Algorithm 3 The Hooke-Jeeves Pattern Search Algorithm

Initialization

Initial point P^{base} of dimension N
Perturbation Matrix A
Perturbation reduction factor α

```
1: while 1 do
2:   P^exp = exploration_move(P^base, A)
3:   if entries in A sufficiently small then
4:     return P^base
5:   else if P^exp = P^base then
6:     A = αA
7:     continue
8:   else
9:     loop
10:      P^pattern = P^exp + (P^exp − P^base)
11:      P^patternB = exploration_move(P^pattern, A)
12:      if P^pattern = P^patternB then
13:        A = αA
14:        break
15:      else
16:        P^base = P^exp
17:        P^exp = P^patternB
18:      end if
19:    end loop
20:  end if
21: end while
```

3.4.3 Conjugate Gradient Algorithm

Background

The Conjugate Gradient method is the most prominent method for solving sparse systems of linear equations. Extension of the method can be made to solve non-linear optimization problems. The Conjugate Gradient method first minimize along the direction of the gradient around initial guess point. Subsequent minimization directions are carefully chosen so that they are as close to the direction of the greatest descent (negative local gradient) as possible, but observing the direction conjugacy property. That is, subsequent search is strictly orthogonal to any previous search

directions. The main routine of the conjugate gradient algorithm is presented at algorithm 4.

Non-Linear Conjugate Gradient with Polak-Ribiere

Algorithm 4 Non-Linear Conjugate Gradient Algorithm with Polak-Ribiere

1: **Initialization:**
2:
3: $g_0 = -\frac{\partial f(X^*)}{\partial X^*}\Big|_{X^*=X_0}$
4: $\lambda_0 = argmin\big(f(X_0 + \lambda g_0)\big)$
5: $X_1 = X_0 + \lambda g_0$
6: i = 1
7: **while** 1 **do**
8:

$$g_i = -\frac{\partial f(X^*)}{\partial X^*}\Big|_{X^*=X_i} \tag{3.18}$$

$$\gamma_i = \frac{(g_i - g_{i-1})g_i}{g_{i-1}g_{i-1}} \tag{3.19}$$

$$h_i = g_i + \gamma_i h_i \tag{3.20}$$

$$\lambda_i = argmin\big(f(X_i + \lambda_i h_i)\big) \tag{3.21}$$

9: **if** $\lambda_i < Epsilon$ **then**
10: Return X_i
11: **end if**

$$x_{i+1} = x_i + \lambda_i h_i \tag{3.22}$$

12: **end while**

Alternative γ Formulation

The formula of γ presented is the original Fletcher-Reeves version. Two other versions are:

Polak-Ribiere:

$$\gamma_i = \frac{(g_{i+1} - g_i)g_{i+1}}{g_i g_i} \tag{3.23}$$

Hestenes-Stiefel:

$$\gamma_i = \frac{(g_{i+1} - g_i)g_{i+1}}{g_i(g_{i+1} - g_i)} \tag{3.24}$$

These formulations can be easily incorporated into the main algorithm by implementing them into 3.19.

Notes

In an N unknown minimization problem, N conjugated search directions are needed for exploration. This amounts to O(N) iterations of the procedures in the while loop. Within each iteration numerical approximation of the Jacobian is performed as in line 3.18, and exactly one line search is performed as in line 3.21. The draw-back of CG is that the derivative calculation at line 3.18 is as expensive as normal function evaluations. It requires numerical approximation such as finite differential method. With N unknowns, this amounts to at least N+1 function evaluations. Although the while loop is executed for, at most, N times, derivative evaluation embedded within each iteration is computationaly costly. These observations make the CG less attractive for very large traffic networks.

3.4.4 Gradient Descent Algorithm

Gradient Descent (GD) is a collection of simple yet elegant ideas. The GD algorithm is also O(N) in speed complexity. However, GD is matrix free - no matrix inversion is performed. Thus it significantly outperforms EKF in memory complexity. GD requires slightly more function evaluations in its line search routine. This incremental cost is largely constant irrespective of the size of the state variables. The main routine of the algorithm is presented in algorithm 5.

Notes

The computational burden of GD comes from 3.25 - the derivative calculations. Similar to that of EKF, this procedure requires N+1 function evaluation for a vector of N

Algorithm 5 The Gradient Descent Algorithm

Initialization

Let P_0 be the starting point in the minimization routine
for time interval h = 0 to Infinity do

Direction Search

$$G = - \bigtriangledown f(X_{h|h-1}) \tag{3.25}$$

Step Size Search

$$(\lambda) = argmin\big((X_{h|h-1} + \lambda G)\big) \tag{3.26}$$

Update

$$X_{h|h} = X_{h|h-1} + \lambda G \tag{3.27}$$

End for loop

unknowns. Instrumental in the process is the determination of the optimal step size λ, which requires additional line search routines. Next, two line search algorithms are presented. They are the Bracketing Search method and the Brent's method.

Bracketing Search in One Dimension

The bracketing search method [Kiefer, 1953] does not require knowledge of the derivatives. It uses a three probe points to bracket the minimum loss function value. Suppose the loss function is bracketed by three points a, b and c, such that $f(a) < f(b) < f(c)$. Bracketing search introduces an additional probe point X, either between a and b, or between b and c. Without lose of generality, if a probe point X is chosen between b and c and $f(X) > f(b)$, then the new bracketing triplet is (a,b,x). If $f(X) < f(b)$, then the new bracketing triplet is (b,x,c). The algorithm reduces the range of the triplet and quits when they are sufficiently close to one another. The bracketing logic is given below in algorithm 6.

Brent's Parabolic Interpolation in One Dimension

The Brent method [Brent, 1973][Atkinson, 1989] also uses a three point triplet to bracket the minimal. However, rather than using probe points on either the left or right half of the bracket, the method fits an inverse parabola through the three points

Algorithm 6 The Bracketing Search Algorithm

Initialization

Let P_0^a, P_0^b, P_0^c be the initial bracketing triplet

for h = 0 to Infinity do

If $\left|P_h^c - P_h^a\right| < \epsilon$ **Break**

Probe X

$$X_{right} = \frac{P_h^c - P_h^b}{2} \tag{3.28}$$

$$X_{left} = \frac{P_h^b - P_h^a}{2} \tag{3.29}$$

if $X_{right} < X_{left}$

$$X = X_{right} \tag{3.30}$$

else

$$X = X_{left} \tag{3.31}$$

Update Bracket

If $f(X) > f(P_h^b)$

$$\begin{aligned} P_{h+1}^a &= P_h^a \\ P_{h+1}^b &= P_h^b \\ P_{h+1}^c &= X \end{aligned} \tag{3.32}$$

Else

$$\begin{aligned} P_{h+1}^a &= P_h^b \\ P_{h+1}^b &= X \\ P_{h+1}^c &= P_h^c \end{aligned} \tag{3.33}$$

End for loop

in the current bracket. [Press et al., 2007] presents the formula of calculating the minimal of a parabola. Given three point a, b and c, the minimum point X on the parabola that fits through the three points is:

$$X = b - \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)] - (b-c)^2[f(b)-f(c)]}{(b-a)[f(b)-f(c)] - (b-c)[f(b)-f(c)]} \tag{3.34}$$

The parabola's minimal is then used as the probe point. The method is more efficient than that of the bracketing algorithm. The argument is that when the loss function is sufficiently smooth near minimal, one of pass of the parabolic interpolation puts the probe point close to the minimum. The Brent's method is also derivative knowledge free.

Algorithm 7 The Brent's Parabolic Interpolation Algorithm

Initialization

Let P_0^a, P_0^b, P_0^c be the initial bracketing triplet
for h = 0 to Infinity do

Calculate Probe X

$$X = P_h^b - \frac{1}{2}\frac{(P_h^b - P_h^a)^2[f(P_h^b) - f(P_h^c)] - (P_h^b - P_h^c)^2[f(P_h^b) - f(P_h^c)]}{(P_h^b - P_h^a)[f(P_h^b) - f(P_h^c)] - (P_h^b - P_h^c)[f(P_h^b) - f(P_h^c)]} \tag{3.35}$$

If $\left|P_h^b - X\right| < \epsilon$ **Break**
Update Bracket
If $f(X) > f(P_h^b)$

$$\begin{aligned} P_{h+1}^a &= P_h^a \\ P_{h+1}^b &= P_h^b \\ P_{h+1}^c &= X \end{aligned} \tag{3.36}$$

Else

$$\begin{aligned} P_{h+1}^a &= P_h^b \\ P_{h+1}^b &= X \\ P_{h+1}^c &= P_h^c \end{aligned} \tag{3.37}$$

End for loop

3.5 Summary

In this section, we presented the algorithm selection criteria - accuracy, robustness and efficiency. Accuracy is the most important requirement. In addition, candidate algorithms need to be robust and reliable. Finally, the candidate algorithms need to be efficient. Algorithms that are accurate and robust but slow are less attractive for real-time deployment of the on-line calibration framework. A balance among the criteria is needed.

The second part of the section presented four candidate algorithms to solve the state estimation problem - PS, CG, EKF and GD. The EKF belongs to the state-space formulation and the other three algorithms belong to the direct optimization formulation. The PS is an iterative, derivative free method that consists of exploration moves and pattern moves. The CG method is an iterative and derivative-based line-search method. At each iteration, a new search direction is obtained. This direction is "non-spoiling" to previous all search directions and also as close to the gradient direction as possible.

The PS is simple to implement and requires little memory. However, a large number of function evaluations is required. The CG method on the other hand, requires as few as N iterations to solve an N unknown minimization problem. However its derivative is too costly to calculate in the context of DTA model on-line calibration. Both algorithms are iterative, meaning they can be terminated when their execution time goes beyond the maximum time limits. For problems with large traffic networks (therefore slower for single function evaluation and a larger number of unknowns), the full versions of both algorithms may be too slow for real-time deployment.

Contrary to the PS and CG methods, which require many iterations, EKF and GD methods are direct methods, perform derivative approximations only once plus some extra finite overhead in the remaining of their routines. Consequencely, EKF and GD tend to be much faster than the full version of PS and CG, considering the expensive cost of function evaluation in this context.

In the interests of efficiency, the differences between the EKF and GD algorithms

are discussed. Both algorithms require partial derivative estimation exactly once. After the derivative estimation, EKF and GD employ different approaches. The overheads for EKF, post derivatives, are in general, *dense* matrix-matrix multiplications and matrix inversions. The overheads for GD post derivatives is additional function evaluations in its line search routine. Also the overheads for EKF is a function of the size of these matrices. It is sensitive to the size of the problem. On the other hand, the overhead for GD is largely constant. Furthermore, from the structural perspective, GD is considered more attractive because its line-search routine uses an iterative method, while the matrix manipulations within EKF are strictly procedural, meaning it cannot output intermedium results. Last but not least, GD is inherently simpler than EKF. It is matrix-free and requires much less memory.

When it comes to comparing the speed of the two algorithms for practical applications, one needs to carefully weigh the overheads discussed above. Is the additional matrix manipulation considered more overheads than additional couple of function evaluations, or vice versa. The key is to bridge the complexity of DTA system and linear algebra operations. [Wen, 2009] argues that the complexity of the DTA system is O(V) for each estimation interval time step, where V is the number of vehicles in the network. Therefore, as the number of vehicles increase linearly, the overheads within the GD algorithm increase linearly. On the other hand, the relationship between the number of the vehicles and speed complexity of the overheads in EKF (primarily related to the state size) is not always so clear-cut. In practice, we would recommend both the EKF and the GD methods. We will reserve the choice to further empirical works.

Chapter 4

Scalable Framework Design

Contents

4.1 Overview **66**

4.2 Scalable Algorithm on Single Processor **66**

4.2.1 The Idea of Simultaneous Perturbation 67

4.2.2 The PSP-EKF Algorithm 68

4.2.3 The PSP-GD Algorithm 70

4.3 Scalable Algorithm on Multiple Processors **72**

4.3.1 Background 72

4.3.2 Architecture 74

4.3.3 Definition 75

4.3.4 The Para-EKF Algorithm 76

4.3.5 The Para-GD Algorithm 77

4.4 Summary **78**

4.1 Overview

Efficiency requirement of the on-line calibration approach entails system scalability. The framework must be able to operate in real-time when dealing with sufficiently complex networks. While the base-line algorithms in the previous sections may be sufficient for smaller problems, they are hardly scalable. The PS and CG requires many iterations in their main routines and each of the main iterations requires 2N number of function evaluations. On average, PS and CG demand more than one function evaluation per state variable. The EKF and GD are superior. They only require N+1 function evaluations, given an N dimensional problem. Although this amounts to approximately one function evaluation per state variable, they are still considered slow when dealing with large networks where a single evaluation takes a long time. In this section, we explore framework scalability along two orthogonal directions: Algorithmic advancement and implementation advancement. Both aim to enhance the speed performance of the proposed on-line calibration framework.

- On the algorithmic side, we develop extensions of the GD and EKF algorithms using the idea of *Simultaneous Perturbation*. We show this modification results into much lower function evaluation to variable ratio.

- On the implementation side, we show that distributed computation can be applied to facilitate the Jacobian calculation and thereby reduce the running time of the GD and EKF algorithms.

4.2 Scalable Algorithm on Single Processor

The application of the idea of Simultaneous Perturbation to the on-line calibration problem was first explored in [Antoniou et al., 2007b]. The algorithm only uses 2 function evaluations and has been shown to produce effective results. This is a remarkable advancement since the original EKF requires N+1 function evaluations with N state variables.

4.2.1 The Idea of Simultaneous Perturbation

The idea of simultaneous perturbation originates from the Simultaneous Perturbation and Stochastic Approximation (SPSA) method, a highly efficient gradient approximation and stochastic optimization algorithm that has been applied mostly in the area of off-line DTA model calibration. The key advantage of SPSA is that, instead of perturbing each N variable individually, it performs simultaneous perturbations to calculate function gradients. The concept of simultaneous perturbation (SP) leads to order of magnitude of savings in computation time.

In the case of both GD and EKF, gradient approximation is the leading computational burden. The classical approach has been to apply numerical differential on each of the state variables independently. This amounts to 2N (at least N+1 if forward differential) function evaluations for a state vector of size N. We show that, application of the Simultaneous Perturbation (SP) can also lead to potentially orders of magnitude of saving in computation time in the case of on-line calibration using GD and EKF. The gist of the idea can be summarized in the following lines.

At each time interval h, the Jacobian estimation H in the standard EKF algorithm is performed according to:

$$H_h = \frac{\partial f(X^*)}{\partial X^*}\bigg|_{X^*=X^*_{h|h-1}} = \begin{vmatrix} \frac{\partial f}{\partial X^0} \\ \frac{\partial f}{\partial X^1} \\ \vdots \\ \frac{\partial f}{\partial X^{N-1}} \end{vmatrix} \tag{4.1}$$

With central stochastic differential, this requires N function evaluations. For $i = 1, 2, ..., N$ we perform:

$$\frac{\partial f}{\partial X_h^i} = \lim_{c->0} \frac{f(X_h + cP) - f(X_h - cP)}{2c} \tag{4.2}$$

Where f reflects the DTA simulator and X_h is the initial state vector for time interval h. c represents a small scalar and P is a padding vector with all entries zero except one for the i^{th} one.

During SP, the padding vector P is randomly initialized with $+1$ and -1 variables. Instead of perturbing each individual variable, the entire padding vector P is used. The stochastic differential can then be computed in one step with only 2 function evaluations:

$$D_h = f(X_h + cP) - f(X_h - cP) \tag{4.3}$$

$$H_h = \begin{vmatrix} \frac{D_h}{2cP_0} \\ \frac{D_h}{2cP_1} \\ \vdots \\ \frac{D_h}{2cP_{N-1}} \end{vmatrix} \tag{4.4}$$

Notice in the SP gradient approximation, equation 4.3 incurs only 2 function evaluations. Equation 4.4 is trivial to compute. In addition, the gradient approximated in this way is a biased estimator of the true gradient. The bias is proportional to c^2. [Spall, 1992] and [Spall, 1994b] provide detailed descriptions.

4.2.2 The PSP-EKF Algorithm

Although the SP gradient approximation has prominent speed complexity, the performance of Jacobian approximation in 4.3, 4.4 is often poor and may need to be further enhanced. The drawback stems from the fact that the state variables that are perturbed together may have common affects on some sensor values, thus imposing common affects on the in-direct measurement portion of the objective function. A simple example illustrates the idea. Imagine a network that consists of only one segment and has only one flow sensor on the segment. The state variables consist of only 2 OD flows that share this common segment. A simultaneous perturbation of (positive, positive) or (negative, negative) gives *overestimated* partial derivatives for the two variables. A simultaneous perturbation of (negative, positive) or (positive, negative) gives at least one *underestimated* partial derivative. When two state variables both affect a sensor value, we say they are *co-reactive* and recommend that they be perturbed in separate runs, for the sake of retaining the accuracy of the Jacobian approximation. On the other hand, when two variables do not have common affects

on any sensors in the network, we say they are mutually **conjugate**. When the property holds for arbitrary pairs of variables in a set, we say the set is a **conjugate set**.

The PSP-EKF stands for Partitioned SP-EKF algorithm. The idea is to strike a balance between efficiency and accuracy. We still want as few perturbations as possible, however, we enforce optimal utilization of each perturbation by including only conjugate state variables. An example of a conjugate set is to include OD flows that cover different areas of the network, and parameters of segments that are geographically far apart from one another. Perturbing these variables together would be identical to perturbing individual variables and combining the results. To achieve this goal, the state variables are partitioned into disjoint conjugate sets. We then simultaneously perturb variables from each of the sets. This can be made more explicit as follows:

The Jacobian matrix obtained from the original EKF provides useful heuristics for partitioning. With a network of N sensors and M state variables, the Jacobian matrix, which we shall call the reactiveness matrix R, is of size N rows by M columns. The non-zero entries in each column signify the reactiveness of that variable on these sensors. The magnitudes of these entries reflect the degree of effectiveness of the variable to those sensors. We define small scalar ϵ_k for $k = 0, ..., N-1$ for each sensor k as a threshold for the degree of effectiveness and we say sensor k is affected by variable i if $R(k,i) > \epsilon_k$. We say two variable i and j are conjugate if $R(k,i) < \epsilon_k$ or $R(k,j) < \epsilon_k$.

The partition process divides M variables into conjugate sets. Because the variables might not be strictly non-co-reactive within a set (due to the threshold), we also prefer the sets to be of similar cardinality in order to minimize the number of variables to be perturbed in a single set. A randomized greedy algorithm can be used to solve this task. We maintain a list of previously created non-empty sets. At each iteration, we randomly shuffle the order of the list. The next variable X is added to the first set that has no variables in conflict with X. In case no such sets exist, a new set is created to hold a single element of X.

At on-line, we perturb all variables from one set at a time. By the property of conjugacy, the resulting sensor value at index k after perturbation is only due to the change in variable i which affects it in the set and not due to any other variables j in the set. Thus, the approximated Jacobian $H(k,i)$ is calculated according to 4.3 and 4.4 and $\forall j \neq i H(k,j) = 0$. In this way, we are able to compute partial derivatives for many multiple state variables from just two model evaluations.

4.2.3 The PSP-GD Algorithm

The application of simultaneous perturbation can also be applied to the direction optimization algorithms. For example, the most computationally expensive step of the GD method - the gradient approximation, may be calculated using simultaneous perturbation. In the case where only two functional evaluations are used, this is a variant of the SPSA algorithm.[1] Next, we describe a more general schema for the idea of simultaneous perturbation in the context of direct optimization frameworks - the Partitioned Simultaneous Perturbation GD (PSP-GD) algorithm.

Let H be size N vector (N sensors) of partial derivatives for all the variables at time interval h (We will drop subscript h from this point on for simplicity). Let the in-direct measurement part of the objective function be the sum of the Euclidean Distance squared between each observed and estimated sensor value. The routine calculation of H follows equation 4.1, which amounts to at least N+1 function evaluations. Suppose the state variables have been partitioned into conjugate sets, we want to show that it is possible to only use P+1 function evaluations to complete derivative approximations instead of M+1, where P is the number of conjugate sets and M is the size of the state variable vector. That is, we apply only one function evaluation for each of the conjugate sets, instead of for each of the state variables.

Let $X = \{v_1, v_2, ..., v_j\}$ be a conjugate set that holds j state variables. By conju-

[1]The objective function within the direct optimization framework is composed of three parts. The derivative of state vector with respect to X is thus the sum of the derivative with respect to X from each of the components. As the derivative calculation with respect to the direct measurement portion is straightforward to compute, we will focus our description of the application of SP to the in-direct measurement part of the objective function.

gacy, change of any simulated sensory value is because of the change from one and only one of the set variables. At operation, simultaneous perturbation of all j variables is performed. Again, let P be a padding vector randomly initialized with $+1$ and -1 for all the variables. Let f be a DTA simulator and let $S^+ = \{S_1^+, S_2^+, ..., S_N^+\}$ be the vector of the resulting upward perturbed simulated sensory value for $f(v_1 + cP_1), f(v_1 + cP_2), ..., f(v_1 + cP_j)$ and let $S^- = \{S_1^-, S_2^-, ..., S_N^-\}$ be the vector of the resulting downward perturbed simulated sensory value for $f(v_1 - cP_1), f(v_1 - cP_2), ..., f(v_1 - cP_j)$. Also assume that the reactiveness matrix R is given, and $R(i, j) = 1$ signifies sensor i is affected by variable j and 0 otherwise. The weights associated with each sensor observation is captured by the vector of W. Equation 4.1 will then be replaced by the following:

$$\frac{\partial f}{\partial X^i} = \lim_{c->0} \sum_{j=1}^{N} \frac{W_j (S_j^+ - S_j^-)^2 R(j, i)}{2cP_i} \tag{4.5}$$

Notice that only 2 function evaluations are sufficient for the partial derivative for all variables in one set. The matrix $R(j, i)$ behaves as a mask that filters out simulated sensor values that are changed by perturbations of other **irrelevant** variables. This is a remarkable improvement from the speed performance on the direct optimization algorithm. Similar to the PSP-EKF, by controlling the threshold, we can vary our preference on the speed vs accuracy. When the threshold is very low, this produces high number of partitions so that the algorithm will behave like the original GD. On the other hand, if the threshold is high, the algorithm will produce very low number of partitions. In its extreme, all variables are within a single partition and we now have the SPSA algorithm.

Note that to achieve the efficiency close to that of the SPSA algorithm, one would need a relatively high threshold. This is because each additional variable of a set must be checked with every other variable that is already in a set. (Recall that within a conjugate set, any two variables are conjugate.) In general, small thresholds signalize emphasis on accuracy and a bigger threshold signalizes emphasis on algorithm running speed.

4.3 Scalable Algorithm on Multiple Processors

We come back to the baseline GD and EKF algorithms. As discussed in the last chapter, different candidate algorithms have different characteristics in terms of speed complexity. While the design of efficient candidate algorithms reflects our vision of pursuing superior computational performance, the implementations of these methods are equally important. In this chapter, we explore the possibility of implementing the two most promising algorithms (EKF and GD) in the context of distributed computing.

Distributed computing refers to algorithms running on a set of machines connected by a network [Pereira et al., 2009]. In the context of this chapter, the objective is to implement the EKF and GD algorithms in this fashion across multiple machines, when these hardware resources are available.

4.3.1 Background

The most computationally burdensome part of the EKF and the GD algorithms lie in their derivative approximations. In the case of EKF, the burden lies in equation 3.12. In the case of GD, the overheads come from equation 3.25. With state vector of size n and assuming forward finite differential approximation, both equations require exactly $n+1$ function evaluations. (Each function evaluation incurs a full DTA model operation) As n grows, the total overheads are extremely significant.

However, the $n + 1$ evaluations can be distributed to multiple processors and thereby reduce the overal computation time in real-time operations. We notice the following characteristics of the derivative approximation routines, which corresponds to those identified in [Antoniou, 2004] :

- Independence - The $n+1$ function evaluations are strictly independent from one another. In addition, the order in which the evaluations are performed does not matter. The i^{th} evaluation may be performed before or after the j^{th} evaluation and vice versa.

- Similar Processing Time per Evaluation - The time required to perform one function evaluation is largely identical. Small fluctuations on the running time may exist due to the perturbation of the state vector in the finite differential approximation. This property is attractive since it allows for the design of much simpler load balancing schemes. Because each evaluation takes roughly the same amount of time, the load level can be simply determined by the processing speed of each processor. This idea will come back in the case study.

- Minimal Inter-Processor Communication (IPC) - The required communications between processors are low. Depending on the design of the scheme, IPC may occur at the end of the EKF algorithm to synchronize the state vector X and variance-covariance matrix P. The situation is even simpler for the GD algorithm. IPC may occur at the end of the GD algorithm to synchronize the state vector X. The amount of data transferred over the IPC is relatively moderate. With a state vector's size of N and the above scheme in operation, the amount of data transferred (payload) for the EKF algorithm would include $4N$ bytes for state vector X and $4N^2$ bytes for the variance-covariance matrix P (Using double precision. Lower precision maybe used for the scalability requirement). This amounts to approximately 4MBs of data with $N = 1000$. With 100Mbs LAN connection, this is not a significant overhead. If the synchronization occur at the Jacobian step in equation 3.12, the amounts of data reduces to $4NN_{ob}$ bytes, where N_{ob} is the number of surveillance sensors in the system (Usually $N > N_{ob}$). The case for the GD algorithm is similar to the analysis above. In general, with any moderate size of N, this process is entirely a trivial exercise.[2]

In the next section, we will describe the architecture of the distributed computing in detail. The implementation details will follow and a summary will be given at the last part of this section.

[2][Wen, 2009] described the techniques of IPC reductions in details.

4.3.2 Architecture

While there are many schemes for distributed computing, this thesis employs the most fundamental one - the *Server-Client* scheme (or sometimes called *Master-Slave* scheme). In this scheme, one machine takes the role of "Server" and is responsible for 1) Coordinating clients for their derivative computations 2) completing the rest of the algorithm routines before and after the computation of derivatives and 3) Initializing synchronization with clients at the appropriate point in time. The exact internal state for both Server and Clients are controlled by a finite state automaton.

The general architecture of the implementation is give below:

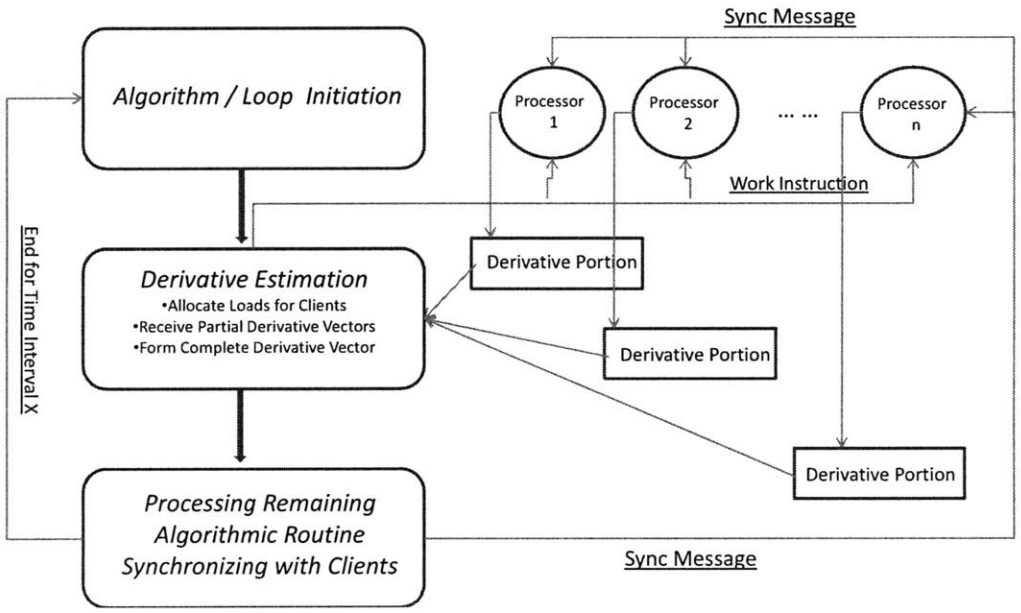

Figure 4-1: The generic server-client implementation of the Para-EKF and Para-GD algorithms

The master is responsible for distributing clients' work load, receiving derivative portions from clients, centralized processing on the remaining routines, and synchronizing clients at the end of each time interval.

At the beginning of a new time interval, the server first performs iteration/loop initializations. The derivative is then calculated in the distributed fashion. The server

allocates portions of the derivative calculations to every client. The derivative will be estimated using the finite differential method stated above. The server first calculates $H(X)$. Clients then calculate $\frac{\partial f}{\partial X_0}, \frac{\partial f}{\partial X_1}, ..., \frac{\partial f}{\partial X_{n-1}}$, respectively. The complete partial derivative can then be formed from derivative portions that are computed from individual clients.

Finally, the server completes the remaining part of the algorithm routines. At the end of the algorithm before proceeding to the next calibration time interval, it sends synchronization messages to each of the clients to align its internal state. In the case of EKF, the synchronization message contains the state vector X and the variance-covariance matrix P. In the case of GD, the synchronization message contains only the state vector X.

4.3.3 Definition

In this section, we present the distributed implementation of the two algorithms. We assume the state vector X_h for time interval h of size N. There are n processors/clients for distributed computing. The load balancing is archived by a prior definition of the boundary vector L which is composed of N 2-tuples of $(Processor_{ID}, Parameter_{ID})$.

$$L = \{(0,0),(0,1),...,(0,s_1),...,(n_{Server},N-3),(n_{Server},N-2),...,(n_{Server},N-1)\}$$

Processor #(0) will be responsible for $\frac{\partial f}{\partial X_0}, \frac{\partial f}{\partial X_1}, ..., \frac{\partial f}{\partial X_{s_1}}$. Likewise, processor #(n-1) will be responsible for $\frac{\partial f}{\partial X_{N-3}}, \frac{\partial f}{\partial X_{N-2}}, ..., \frac{\partial f}{\partial X_{N-1}}$.

The Para-EKF is presented in table 4.1. The Para-GD is presented in table 4.2.

4.3.4 The Para-EKF Algorithm

Server Side	Client J Side
1: **Time update**	
$X_{h\|h-1} = F_{h-1}X_{h-1\|h-1}$ $P_{h\|h-1} = F_{h-1}P_{h-1\|h-1}F_{h-1}^T + Q_h$	
4: **Taking partial derivative**	
5: **for** $i = 1$ to N-1 **do**	1: **Time update**
6: $H_h^i = \frac{\partial f(X_h^i)}{\partial X_h^i} = Clients\ [L[i][0]].H[L[i][1]]$	
7: **end for**	
8: **Measurement update**	$X_{h\|h-1} = F_{h-1}X_{h-1\|h-1}$ $P_{h\|h-1} = F_{h-1}P_{h-1\|h-1}F_{h-1}^T + Q_h$
$G_h = P_{h\|h-1}H_h^T\left(H_hP_{h\|h-1}H_h^T + R_h\right)^{-1}$	4: **Taking partial derivative**
$X_{h\|h} = X_{h\|h-1} + G_h\left[Y_h - h(X_{h\|h-1})\right]$	5: **for** $j = S_{J-1} + 1$ to S_J **do**
$P_{h\|h} = P_{h\|h-1} - G_hH_hP_{h\|h-1}$	6: $H_h^j = \frac{\partial f}{\partial X_h^j}$
12: **Synchronization**	7: **end for**
$\forall_{i:[1..n]}\ Clients[i].X_{h\|h} = X_{h\|h}$ $\forall_{i:[1..n]}\ Clients[i].P_{h\|h} = P_{h\|h}$	

Table 4.1: Distributed Extended Kalman Filter Algorithm - Para-EKF

4.3.5 The Para-GD Algorithm

Server Side	Client J Side
State Prediction	
$X_{h\|h-1} = X_h^a + \Delta X_{h-1\|h-1}$	
Direction Search	
for $i = 1$ to N-1 **do**	
$H_h^i = \frac{\partial f(X_h^i)}{\partial X_h^i} = Clients\,[L[i][0]].H[L[i][1]]$	**State Prediction**
end for	$X_{h\|h-1} = X_h^a + \Delta X_{h-1\|h-1}$
Step Size Search	**Direction Search**
$(\lambda) = argmin\big((X_{h\|h-1} + \lambda H)\big)$	**for** $j = S_{J-1} + 1$ to S_J **do**
Update	$H_h^j = \frac{\partial f}{\partial X_h^j}$
$X_{h\|h} = X_{h\|h-1} + \lambda H$	**end for**
$\Delta X_{h\|h} = X_{h\|h} - X_h^a$	
Synchronization	
$\forall_{i:[1..n]}\ Clients[i].X_{h\|h} = X_{h\|h}$	

Table 4.2: Distributed Gradient Descent Algorithm - Para-GD

Distributed implementation helps to reduce the amount of time spent on the derivative estimation. However, other parts of the algorithms (those that serialize in procedures), still need to be processed locally on the server. For EKF, the major overheads are the matrix inversion and multiplications. For GD, additional function evaluations are performed post-derivative estimation. Assuming that the processors are of equal speed and derivative calculation dominates speed complexity, with n processors, it is expected that the distributed implementation is approximately n times faster than the single-processor configuration.

4.4 Summary

In this section, practical considerations are given to target the scalability of the on-line calibration framework. This chapter proposes two strategies. On the algorithmic side, we propose the idea of SP for optimal algorithm scalability. This is later extended by more generalized methods: the PSP-EKF and PSP-GD algorithms for enhanced speed performance while retaining a high degree of accuracy. On the implementation side, distributed computing helps leverage the overheads in the EKF and GD algorithms. We provide implementations for both Para-EKF and Para-GD algorithms. The implementations are based on the master-client model. The master processor is responsible for running the overall algorithms, allocate jobs and receive results to/from clients, as well as synchronization between the server and clients. We conclude that the two strategies are orthogonal and complementary. It is possible to combine the techniques to form the hybrid Para-PSP-EKF/Para-PSP-GD algorithms - the PSP-EKF/PSP-GD that runs individual partitions on separate processors.

Chapter 5

Case Study

Contents

5.1 Objectives 80

5.2 Experiment Setup 80

5.2.1 Network and Sensors 80

5.2.2 DynaMIT-R 82

5.2.3 Parameters and variables 82

5.2.4 Experiment Design 83

5.3 Results 84

5.3.1 Offline Calibration 84

5.3.2 Base-line Algorithms Online Validation 86

5.3.3 Scalable Design Validation: PSP-GD and PSP-EKF 92

5.3.4 Scalable Design Validation: Para-GD and Para-EKF 97

5.4 Summary 101

5.1 Objectives

The objective of the case study is to demonstrate the feasibility of the proposed on-line calibration approach. In particular, **accuracy**, **robustness** and **speed** of the proposed candidate algorithms are discussed.

Accuracy refers to the ability of taking an array of sensor data to generate highly consistent network estimations and predictions. This translates directly to achieving low values for the performance indicators on current as well as predicted future network conditions.

However, being able to achieve good accuracy on non-fluctuating network scenarios is not sufficient. The ability to consistently achieve good accuracy across a number of different scenarios is desired. Robustness of the candidate algorithms will therefore be experimented and discussed.

Finally, the candidate algorithms need to be efficient. The real-time nature dictates that the proposed framework runs as fast as possible. Candidate algorithms with slow speeds (e.g. operate in the order of hours) have no practical value for on-line deployment.

5.2 Experiment Setup

In this section, the case study experiments are discussed in detail. The layout of the section is as follows. First the network and deployment of sensors are presented. We then describe the DTA system used in the experiments and the model variables for optimization. Finally, a description for each of the experiments is presented. The experiments are designed specifically to target the objectives described above.

5.2.1 Network and Sensors

The network used in this analysis is the Brisa A5 motorway. (A5 - Auto-estrada da Costa do Estoril) It is a 25-km inter-urban expressway section between Lisbon and Cascais. The motorway consists of 85 road segments (mostly highway segments)

and 56 nodes representing significant points in the network. The layout of the study network is given in figure 5-1. The A5 motorway is located in western Portugal. It links Cascais with Lisbon along a number of major destinations in the region.

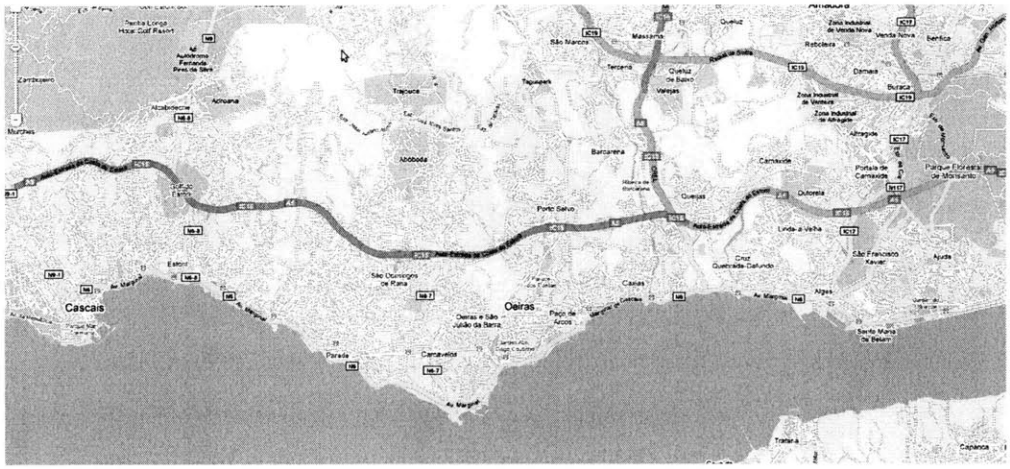

Figure 5-1: The study network - Brisa A5. Source: Google Map 2009

The motorway is primarily equipped with toll collection systems, surveillance camera counters and inductive loop detectors. All these sensors provide time dependent vehicle counts at specific parts of the network.[1] The network also deploys point-to-point sensors such as the *Via-verde* sensor. The sequential identification of vehicles between consecutive toll plazas using *Via-verde* offers measurements of average segment speeds in real-time.[2] The point-to-point sensors cover segments between 5 pairs of check stations: Pair 1: Estoril to Carcavelos; Pair 2: Carcavelos to Service Area; Pair 3: Service Area to Estadio; Pair 4: Estadio to Linda Velha, Pair 5: Linda Velha to Lisbon. The deployment schema of the fixed sensors and point-to-point *Via-verde* stations in the network are given in figure 5-2. Four types of fixed sensors are deployed: 1) Loop detectors 2)Toll Plaza Counter 3)Video Camera Counter 4) Point-to-point Sensor. The network management system also provides incident data, including type and location [Huang et al., 2009].

[1]The camera counters are still under development, thus they are omitted from the observations.

[2]It is a dedicated short range communication technology used to identify vehicles at toll plazas. When vehicles pass a pair of toll gates, their travel times are logged and average segment speeds can be estimated.

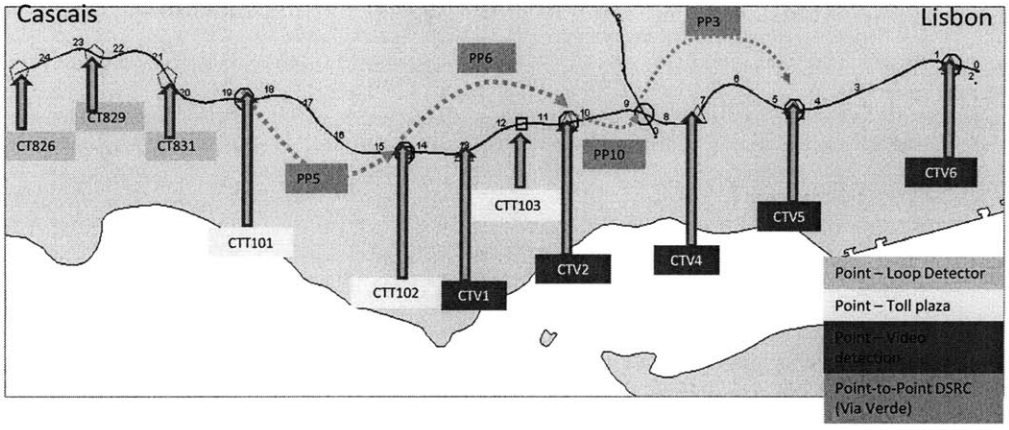

Figure 5-2: The sensor deployments on the Brisa A5 study network

5.2.2 DynaMIT-R

DynaMIT-R (stands for DynaMIT Real-time) is chosen as the candidate DTA system in the experiment. DynaMIT-R is a state-of-the-art DTA system for traffic network estimation and prediction developed by the MIT Intelligent Transportation Systems Laboratory. The system is composed of two components 1) Demand simulation and 2) Supply simulation. The demand simulation conducts pre-trip behavior modeling using historical OD demand matrix. This includes a simulation of the choice of departure time, routes and whether the trip is canceled, while considering the latest guidance information. The supply simulation takes the listed drivers with choices from the demand simulator, and conducts meso-scopic traffic simulation. While traveling, drivers' behaviors are constantly updated in response to the latest guidance. The outputs of DynaMIT-R are network characteristics such as link speed, average travel time, densities as well as drivers' benchmarks such as trip travel time, trip length, fuel consumption, etc.

5.2.3 Parameters and variables

The parameters in DynaMIT-R that are subject to scenario-to-scenario calibrations can be loosely grouped into three categories: 1) Parameters related to driver behaviors

2) Supply simulator parameters related to road segments and 3) Time dependent OD flow matrices and auto-regressive parameters.

The first group consists of parameters within the behavior choice models. These parameters, which generally reflect drivers' long-term decision-making philosophies, are for the purposes of this case study, assumed to be constant in the real-time calibration framework that deals with a much shortened time horizon. To that end, these parameters in the behavior choice model are usually estimated during the off-line process and are assumed to be constant in the on-line stage. [Antoniou et al., 1997] presents a complete list of the behavior parameters and their descriptions in DynaMIT-R.

The second group is composed of parameters that describe the characteristics of the network supply dynamics. These parameters are exemplified by the speed-density relationship model in DynaMIT-R's supply simulator. They consist of the maximum and minimum speeds and densities and, two model parameters α and β. These parameters will be included in the optimization framework. These parameters capture the short-term changes in network conditions.

The last group is composed of time dependent OD matrices. The OD flows are included in the optimization framework to account for short-term fluctuations in traffic demand. This is because the actual demand on the day of operation might be higher or lower than the historical value. Unlike the behavior parameters, deviations in demand level are likely to be significant between days. Proper adjustments of the demand levels are therefore, critical in reproducing consistent network conditions.

There are a total of 43 OD flow pairs specified, and 85 segments. The segments can be grouped into homogeneous groups and their supply parameters can be calibrated at the grouping level. The grouping of the segments' supply parameters is a common approach for reducing the dimension of the state vector [Aerde and Rakha, 1995].

5.2.4 Experiment Design

The experiment consists of two types of scenarios - the model is first calibrated off-line for the duration of a normal week day. The off-line calibrated model is then validated with data from a different week covers the same time period. The candidate algo-

rithms are tested during the validation experiment. The validation experiment tests the algorithms' accuracy (during the estimation and prediction stages), robustness (each experiment is repeated multiple times with different initial random seeds) as well as efficiency (algorithm running time). We use DynaMIT-R with one iteration of OD estimation as the base algorithm for comparison. During validation, we compare four candidate algorithms: PS, CG, GD and EKF with the base algorithm. Scalability is also investigated. Different sized state vector are used during validation. The initial comparison is carried out on a relatively small state vector (with the supply parameter combined into homogeneous groups). Later on, we will test scalable algorithms: PSP-GD, PSP-EKF, Para-GD and Para-EKF on the full state vector without groupings. The results and discussions are presented in the next section.

5.3 Results

5.3.1 Offline Calibration

The DynaMIT-R model is calibrated off-line for the day of 10-December-2009. OD flow levels, speed-density relationships and auto-regressive parameters are calibrated using an iterative Pattern Search method[3]. To facilitate the speedy comparison of large sets of candidate algorithms, the experiment period is selected to be 10 min intervals between 3:10pm-5:10pm, and during which 1) temporal traffic fluctuations exist and 2) traffic intensity is relatively mild so that comparisons among multiple algorithms can be made in shorter time duration. The count and speed estimates from the DynaMIT-R model after off-line calibration are closely aligned with the observations. The off-line calibrated RMSN for count is 0.087, and the off-line calibrated RMSN for speed is 0.109, during the experiment period. The off-line calibration result can best be presented using the 45 degree plot in figure 5-3 and 5-4.

The X axis represents the model's estimated speeds (mph) or counts (# vehicles)

[3]The calibrated set of ODs and model parameters from previous iterations is used as the starting values for their calibration in the next iteration. [Balakrishna, 2006] discussed off-line DTA calibration in detail.

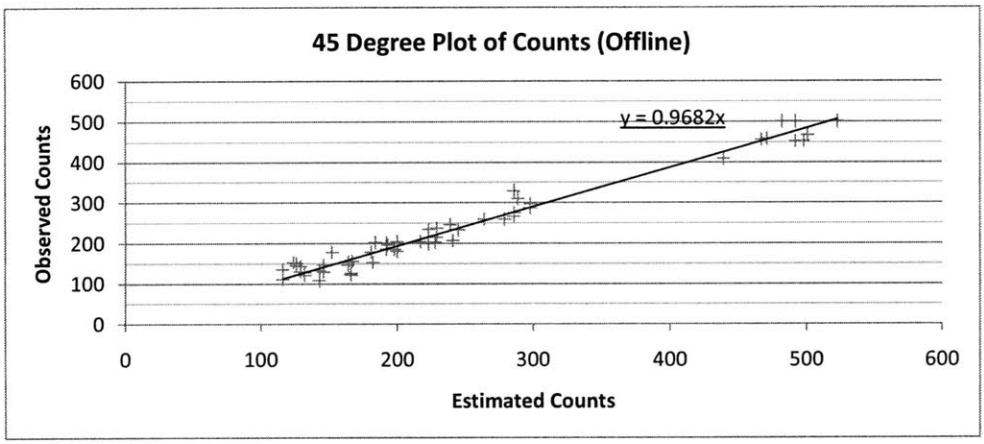

Figure 5-3: A 45 degree plot of estimated sensor counts against observed sensor counts during off-line calibration

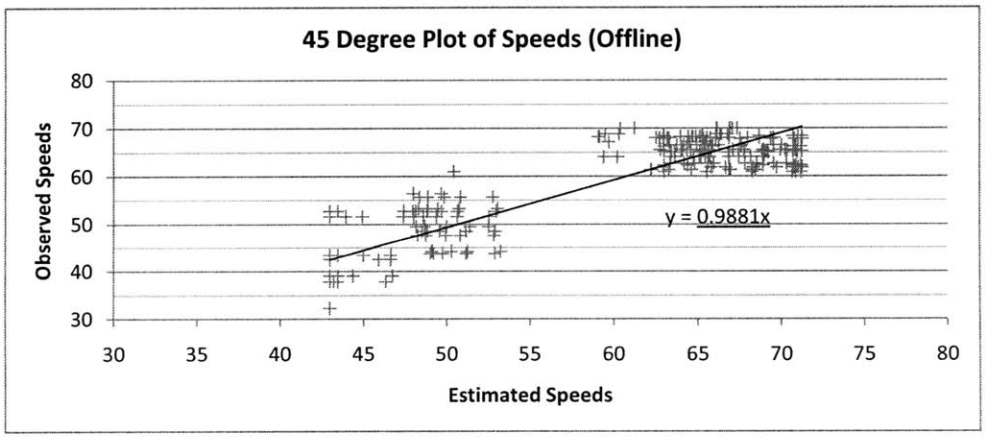

Figure 5-4: A 45 degree plot of estimated sensor speeds against observed sensor speeds during off-line calibration

and the Y axis represents the observed speeds (mph) or counts (# vehicles). A linear regression is run on the resulting data pair. The slopes of the lines for both counts and speeds are very close to 1.

Although the off-line calibrated model is capable of reproducing count and speed estimates that matche closely to sensory observations, off-line calibrated models without functionality of real-time adjustments cannot overcome traffic condition shifts between days. The shifts between two days may be due to weather conditions, incidents, special events, or simply the stochasticity of the traffic flows. For example, the difference between traffic sensory observation on 11-Jan-2010 (one month after 10-Dec-2009) and that of 10-Dec-2009 is significant. The fluctuation of counts across the two days are presented in figure 5-5. The systematic reduction of traffic volume may be caused by the demand sensitivity at this point in time in a year. The fluctuation of speeds across the two days are presented in figure 5-6. It is evident that significant changes in traffic patterns exist - while the off-line calibrated model is trained to cope with one scenario, it might not perform well for another scenario where unforeseen traffic patterns are present.

5.3.2 Base-line Algorithms Online Validation

The 11-Jan-2010 (Monday) data is used for on-line algorithm validations. Data is collected for the same period of time as that of the off-line calibration conducted for 10-Dec-2009 (Thursday). Four candidate algorithms are compared - PS, CG, EKF and GD. The base algorithm corresponds to DynaMIT-R running OD estimation for one iteration.[4] The parameters subject to optimization consist of all OD flows and supply parameters that are grouped into 24 portions. For each segment group we calibrate 6 parameters in real-time: Capacity, Max Speed, Min Density, Max Density, Alpha and Beta as defined by 2.20. This makes the state vector of length of

[4]The intuition of a base reference is to compare alternative on-line algorithms with an algorithm that has limited calibration capability. The emphasis is among candidate algorithms and between them as a group and the base. Although the base conducts only one iteration of OD, additional iterations maybe performed and the enhancement is only expected to contribute to the variability to the OD flows.

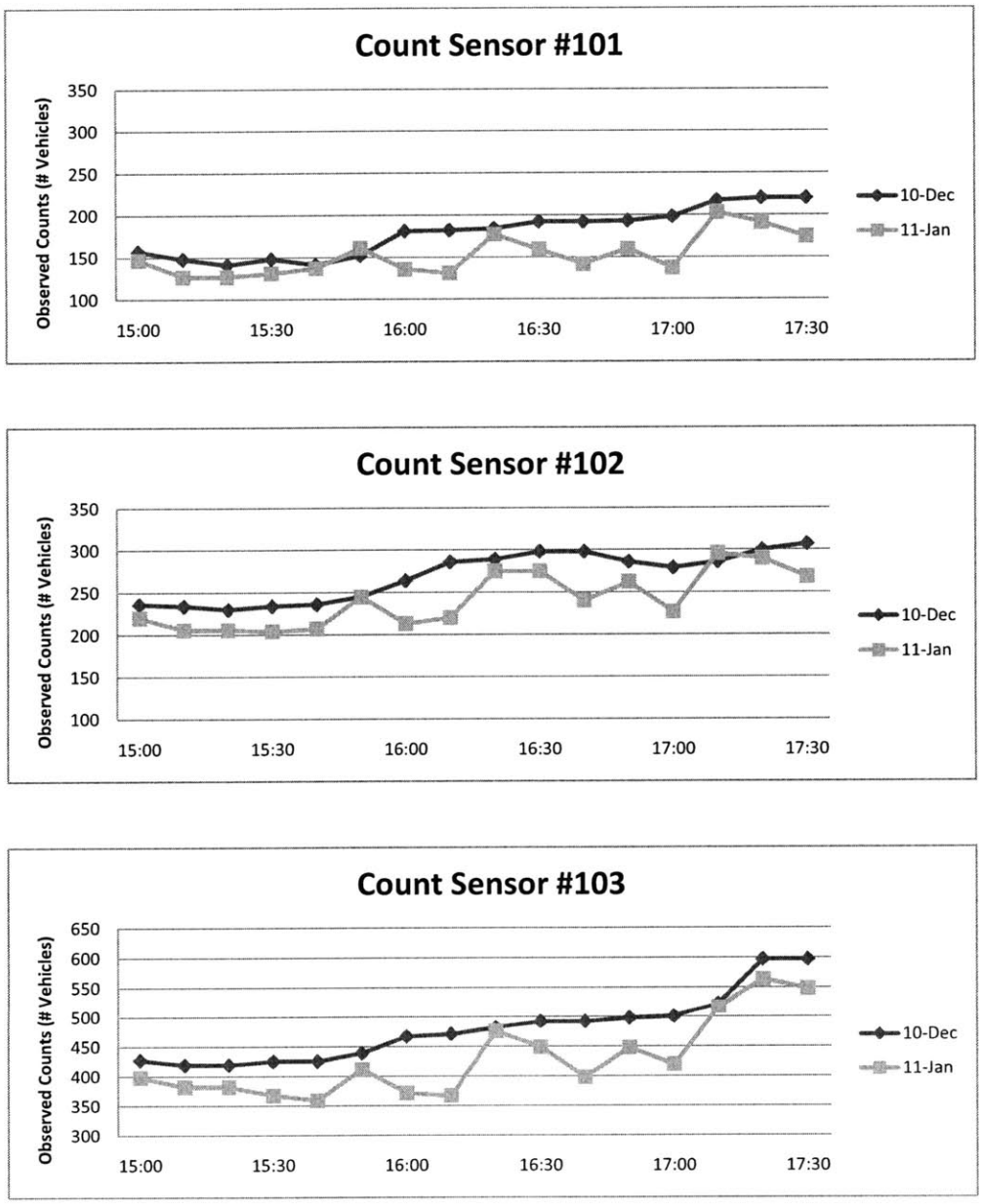

Figure 5-5: Comparison of traffic count observations between Dec-10-2009 and Jan-11-2010

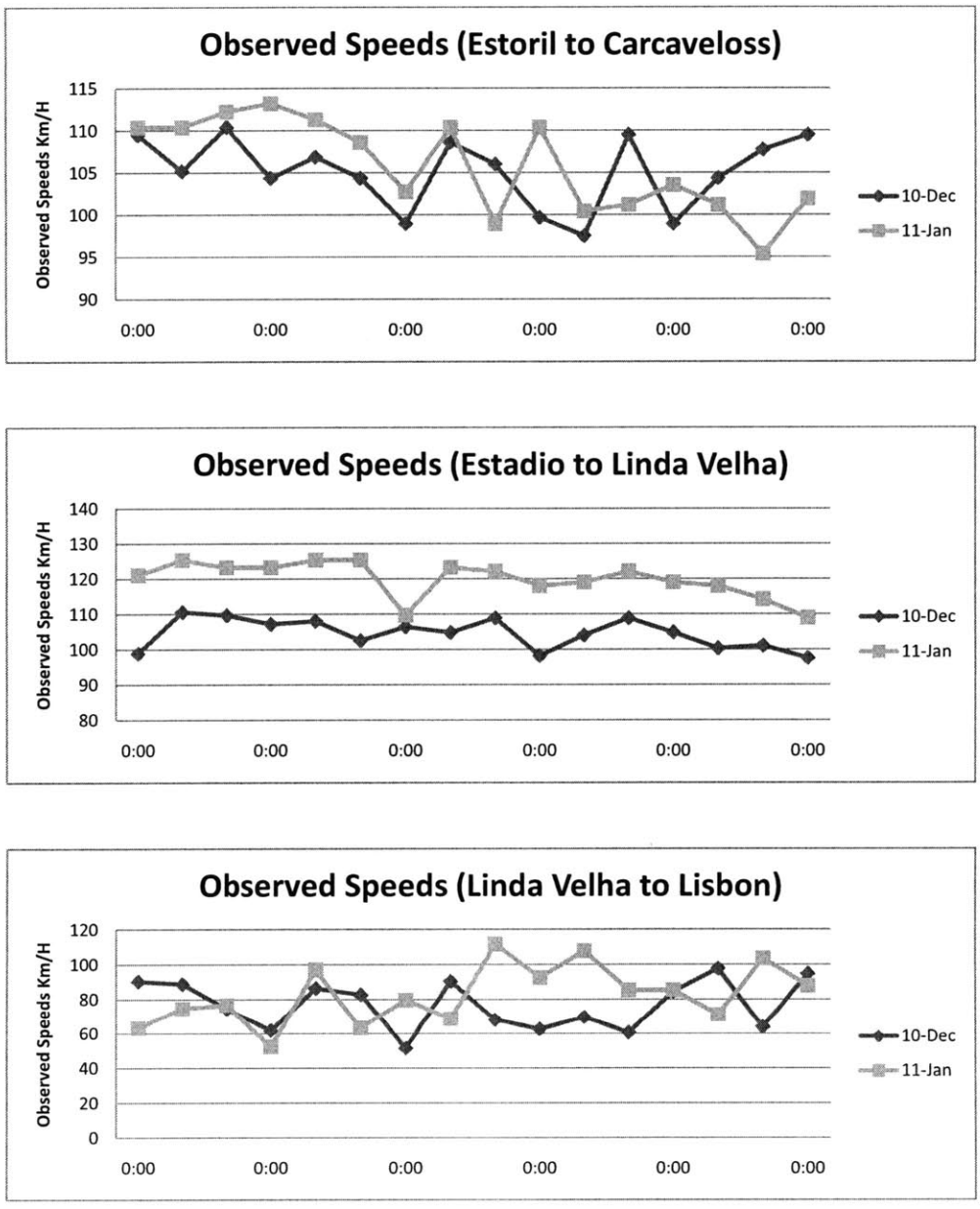

Figure 5-6: Comparison of traffic speed observations between Dec-10-2009 and Jan-11-2010

$43 + 24 \times 6 = 187$

The PS and GD algorithms are expected to be slow, thus operational constraints are specified instead of running the full versions till their convergence. The PS and GD are limited to run the outer loops only twice. This limits the running time of the algorithms to less than 5 mins during a 10 mins calibration interval. The implementation of PS is based on [Deb, 2005]. The implementations of CG and Brent's in GD are based on [Press et al., 2007]. The linear algebra library used in the implementations is the *C++ Boost* library with *Atlas Numerical Bindings*. The matrix inversion is implemented using the LU decomposition techniques. Benchmarking shows that our implementation of matrix inversion for a dense matrix initialized with random entries of size 500×500 takes sub-second, and less than 40 seconds for size of 5000×5000.[5] All comparisons are performed 5 times. For comparison purposes, identical sets of 5 random seeds are used across the algorithms. Average RMSNs are reported for all algorithms. *GNU gprofiler* is used for speed benchmarking. All algorithms are tested on the same Linux machine running Ubuntu 9.04 with a CPU configuration identical to that of the matrix inversion test. DynaMIT-R is compiled using G++-4.2 with -O3 flagged, debugging disabled and terminal/disk I/O suppressed.

The performance of estimation as well as 3-step predictions for both counts and speeds are tabulated below:

Table 5.1: RMSN for the online validation for counts for the four candidate algorithms as well as improvement over the base algorithm

	Estimation	**1-Step**	**2-Step**	**3-Step**
Base	0.174	0.174	0.175	0.175
CG	0.142 (18.4%)	0.156 (10.3%)	0.165 (5.71%)	0.167 (4.57%)
PS	0.122 (30.0%)	0.154 (11.5%)	0.164 (6.30%)	0.168 (4.00%)
GD	0.143 (17.8%)	0.160 (8.05%)	0.165 (5.71%)	0.167 (4.57%)
EKF	0.118 (32.2%)	0.150 (13.8%)	0.161 (8.00%)	0.164 (6.29%)

As it can be seen, all four candidate algorithms significantly outperform the base algorithm. The improvements over the base peaks for the estimation state and grad-

[5]On Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz

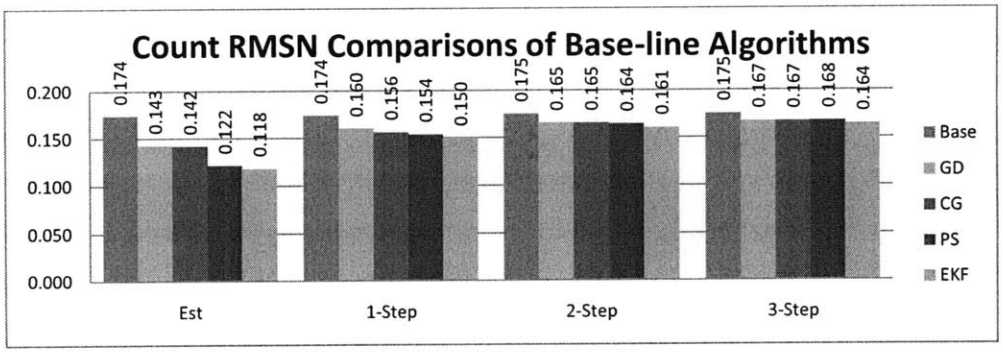

Figure 5-7: Comparison of traffic count RMSN among base-line algorithms

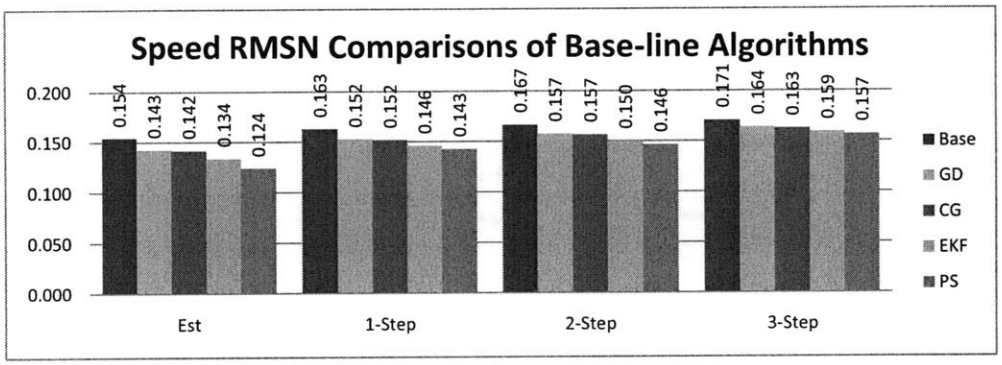

Figure 5-8: Comparison of segment speed RMSN among base-line algorithms

ually decrease as more prediction steps are taken. In particular, the PS and EKF are superior to the CG and GD algorithms under both the estimation and prediction stages.

Table 5.2: RMSN for the online validation for speeds for the four candidate algorithms as well as their improvment over the base algorithm

	Estimation	**1-Step**	**2-Step**	**3-Step**
Base	0.154	0.163	0.167	0.171
CG	0.142 (7.79%)	0.152 (6.75%)	0.157 (5.99%)	0.163 (9.53%)
PS	0.124 (19.5%)	0.143 (12.3%)	0.146 (12.6%)	0.157 (8.19%)
GD	0.143 (7.14%)	0.152 (6.75%)	0.157 (5.99%)	0.164 (4.09%)
EKF	0.134 (13.0%)	0.146 (10.4%)	0.150 (10.2%)	0.159 (7.02%)

In the above, we compared the four candidate algorithms' accuracy performance

and we have demonstrated the feasibility of our framework using the state-space and the direct optimization frameworks. We notice that the base algorithm performed less well compare to the off-line testing case. This is expected as the matching of the traffic pattern and surveillance data are inexact between the two dates. As a result of the miss-matching, the real-time model calibration approach is needed. Although numerically promising, it can be shown that none of these algorithms are sufficiently fast for large networks - they all require many number of function evaluations. Among the four algorithms, CG and PS have much higher computational complexity, therefore they more function evaluations than that of EKF and GD.

Table 5.3: Algorithm running speed comparison for CG, PS, EKF and GD

	CG		**PS**		**EKF**		**GD**	
	Tot Expr Running Time (sec)	Tot #Func Evaluations (Master)	Tot Expr Running Time (sec)	Tot #Func Evaluations (Master)	Tot Expr Running Time (sec)	Tot #Func Evaluations (Master)	Tot Expr Running Time (sec)	Tot #Func Evaluations (Master)
Trial 1	137.51	6371	219.28	9855	37.89	1728	40.33	1895
Trial 2	115.07	5131	178.91	8032	36.66	1728	40.38	1877
Trial 3	156.51	6801	180.24	7997	36.38	1728	40.51	1908
Trial 4	155.38	6801	189.55	8378	36.66	1728	39.05	1902
Trial 5	157.00	6586	192.81	8691	35.55	1728	42.15	1914
Average	**144.29**	**6338**	**192.16**	**8590**	**36.6**	**1728**	**42.48**	**1899**

The computational performance of the validation experiment is presented in table 5.3. Both total wall clock running time and number of function evaluations per estimation interval is presented. The PS requires the most number of function evaluations. This is followed by CG, requiring, on average, 3.20 function evaluations per unknown. The most efficient algorithm is EKF. The GD requires a small overhead

in its line search routine, thus requires on average 1.12 function evaluations per unknown. Both EKF and GD employ O(N) speed complexity with respect to state vector of N unknowns. Notice that the overall running time for GD is still superior to EKF. This is due to the simplicity of the GD algorithm - no dense linear algebras.

Although PS achieved good accuracy as measured by its RMSN statistics, its running speed is the worst among all candidates. The running time for CG is slightly better than that of PS. The total number of function evaluations CG spent is about 3.5 times that of the EKF and GD. Although PS and CG methods are highly accurate and robust, they nevertheless do not meet the efficiency requirement. Their high speed complexities make them inappropriate for large network deployments.

On the other hands, EKF and GD achieved orders of magnitude lower running time. The benchmark showed average total experiment running time of 36.6 seconds for the EKF and 42.5 seconds for the GD. In addition, the number of function evaluations per state variable for EKF is 1.0 and 1.1 for GD. These results show that EKF outperformed GD in terms of running speed on this validation experiment. The fact that GD spent, on average, more function evaluations is because of its overheads in its line search routine. Although this speed advancement is quite remarkable, compared to PS and CG, their workability when challenged with large and complex networks, is still questionable. Next, we present the validation for the scalable algorithms.

5.3.3 Scalable Design Validation: PSP-GD and PSP-EKF

For the same validation data set, we extend the state size. The state now includes 43 OD flows as well as parameters for all mainline segments in the network. There are 85 mainline segments. The segment parameters now include the 7^{th} parameter - *minimal segment speed.* This, therefore, amounts to a total of $43 + 85 \times 7 = 638$ unknowns for on-line calibration. Since the base-line algorithms are to slow, they are left out of the comparison. Thus, this validation experiment will focus on the performance of the scalable solution approaches. In particular, we will focus on the single processor configuration algorithms: the PSP-GD and PSP-EKF algorithms. The objective is to showcase their workability when dealing with large state size and demonstrate: 1)

Significant improvement over the base (preferably as close to their non-SP counter-part as possible) and 2) Significant improvement on algorithm efficiency (ratio of function evaluations over state size) over their non-SP counter-part.

Table 5.4: Count RMSN Performance of PSP-GD and PSP-EKF algorithm with extended state unknowns

	Estimation	**1-Step**	**2-Step**	**3-Step**	**# F**
Base	0.174	0.174	0.175	0.175	-
PSP-GD	0.162 (6.6%)	0.161 (7.3%)	0.168 (4.1%)	0.168 (3.7%)	44 [6]
PSP-EKF	0.142 (18.4%)	0.160 (7.8%)	0.165 (5.8%)	0.167 (4.7%)	44 [7]

Table 5.5: Speed RMSN Performance of PSP-GD and PSP-EKF algorithm with extended state unknowns

	Estimation	**1-Step**	**2-Step**	**3-Step**	**# F**
Base	0.154	0.163	0.167	0.171	-
PSP-GD	0.148 (4.1%)	0.158 (3.2%)	0.161 (3.4%)	0.167 (2.4%)	44 [6]
PSP-EKF	0.131 (15.3%)	0.150 (7.4%)	0.151 (9.7%)	0.162 (4.9%)	44 [7]

In this experiment, PSP-GD, PSP-EKF are implemented and tested against the large state data set. The partition of the state variables is done off-line. 43 partitions of the entire state spaces are made based on the heuristics from an EKF Jacobian matrix. This amounts to a total of 44 function evaluations for the entire Jacobian approximation while on-line. The line search after gradient approximation is limited to 10 function evaluations. If we ignore the constant portions of algorithmic procedures in PSP-GD (in its line searches) and PSP-EKF (due to the matrix manipulations), this partition schema will translates to function evaluation to state size ratio of 0.069, approximately 15 times faster than the ordinary GD and EKF algorithms used in the previous validation experiment. Both PSP-GD and PSP-EKF are implemented based on the same partition schema. The comparison result for both counts, speeds as well as number of function evaluations are presented in tables 5.4 and 5.5

[6]This excludes function evalutions in the line search process.

[7]Additional calculations are required for matrix manipulations.

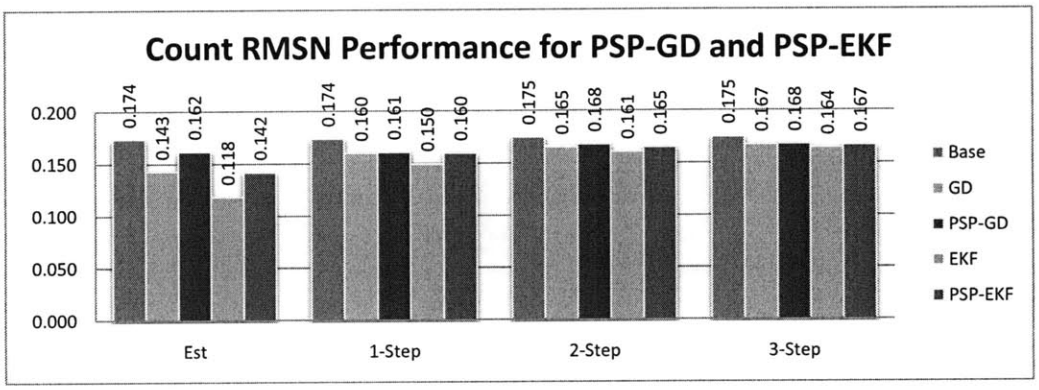

Figure 5-9: Comparison of traffic count RMSN among GD, PSP-GD, EKF and PSP-EKF algorithms

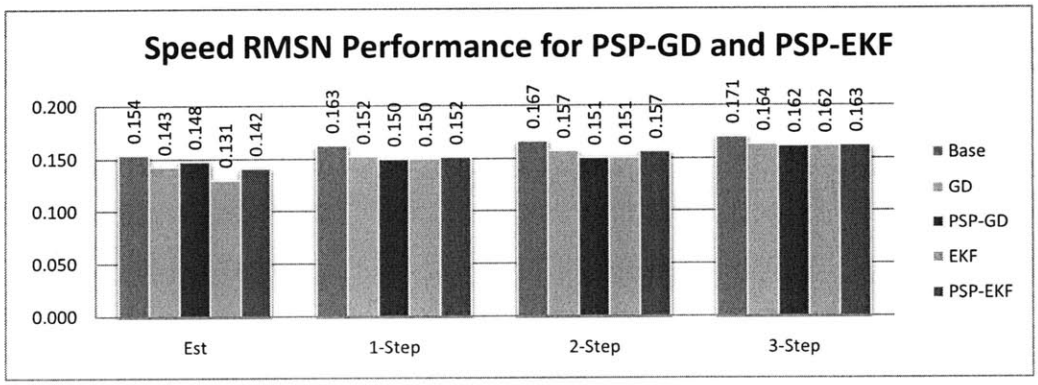

Figure 5-10: Comparison of segment speed RMSN among GD, PSP-GD, EKF and PSP-EKF algorithms

The key observation here is the significant improvement on the number of function evaluations for PSP-GD and PSP-EKF. While we maintain 638 variables, the computational complexity for PSP-GD and PSP-EKF are orders of magnitude better than the base line algorithms. Besides, the accuracy performance, signified by their RMSNs matches the performance of their base-line family members, GD and EKF, respectively. This maybe best revealed by examining figures 5-9 and 5-10. In the plots, we compare the base algorithm, GD and PSP-GD, as well as EKF and PSP-EKF.

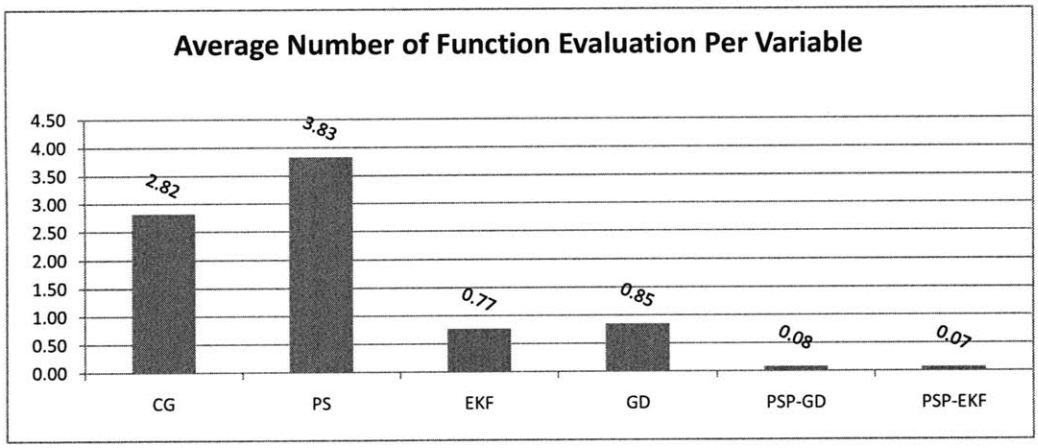

Figure 5-11: Comparison of algorithms' scalability using the average number of function evaluations per state variable per estimation interval

Finally, we summarize the speed performance for the single processor on-line calibration algorithms. We plot the function evaluation to state size ratio for all four base-line algorithms as well as the PSP-EKF and PSP-GD. The repeated findings of search directions in both PS and CG have resulted in a large degree of inefficiencies. The EKF and GD have improved speed performance. Both algorithms required approximately N function evaluations with N state unknowns. This is further improved by a fold of 10 by the application of the idea of Partitioned Simultaneous Perturbation. On average, the PSP-GD and PSP-EKF only require approximately 0.07 function evaluations per state variable. In addition, it has been shown that PSP-GD and PSP-EKF achieved a high level of efficiency while maintaining prediction

accuracy that is comparable to that of GD and EKF.

We also plot a speed/accuracy compensation diagram. The average number of function evaluations required per state variable for each algorithm is ploted against its prediction RMSN accuracy.

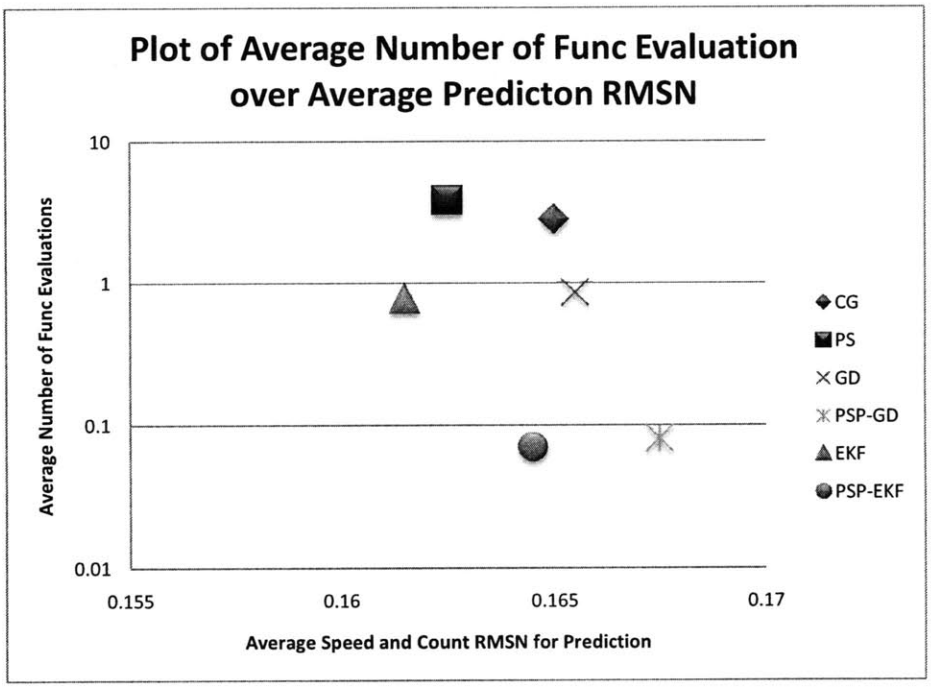

Figure 5-12: Comparison of algorithms' trade off of scalability against accuracy. X-axis: algorithm's prediction RMSN (calculated roughly as $\frac{1}{2}RMSN_{spd} + \frac{1}{2}RMSN_{count}$), Y-axis: algorithm's average function evaluations per state variable

The optimal algorithm in the plot would position itself as close to the (0,0) as possible. As one can see, the most accurate and the most efficient algorithms are the EKF and the PSP-EKF respectively. As we move away from the original point, we encounter different trade-offs among candidates. For example, the PS is accurate but very slow. The PSP-GD is faster compared to the rest of the candidates, but it is less accurate.

The trade-off between the accuracy, as reflected by the prediction RMSN and algorithm running speed, as reflected by the average number of function evaluations required per state variable, is important from a practical point of view. Algorithms

that require many function evaluations tend to be more accurate. On the other hand, algorithms that are faster, due to the application of PSP, tend to be less accurate. The balance between the two factors needs to be achieved and we shall reserve the choice of the most appropriate algorithm to case-by-case empirical analysis.

5.3.4 Scalable Design Validation: Para-GD and Para-EKF

We present the validation experiment for the multi-processor scalable design - distributed implementations. We come back to the base-line GD and EKF algorithm with their speed performance presented in table 5.3. As the validation state size increased from 187 to 638, the corresponding average running time of the two algorithms also increased. The total running times for EKF and GD for the large state experiment are 151.52 and 131.82 seconds. Although the increase on the overhead is almost linear, it can become intractable as the state size becomes large enough. One important observation is that the calculation of one variable's partial derivative is strictly independent of one another and there is no reason one needs to serialize the procedure. Another important observation is that the distributed approach is lossless - that it produces identical results compared to the serialized approach.

This section tests the distributed implementation of the two algorithms, aiming at verifying the feasibility of the approach. As the distributed computation is lossless, we will focus our analysis on the running speed aspects. In this experiment, four processors are connected via wired Internet and one processor is connected through wireless. *Internet Communication Engine* (ICE) is used to provide communication between processors via Internet access. ICE is a modern object-oriented middle-ware with support for many languages including C++. We ran the distributed implementations of the two algorithms with configurations of 2 processors, 3 processors and 5 processors and we logged the speed performance of the master processor. For the purposes of this experiment, identical random seeds are seeded at initialization among the processors. For each configuration, we repeat 5 times for both algorithms. In table 5.6, we summarize total running time, total function evaluation time, average time spent per single function evaluation and total time spent on internal function

evaluations for the *server processor.*

The statistics suggest effective reduction of the operational time spent in the function evaluation, and as a result, reduction on the overall algorithm running time. The total running time for EKF dropped from 151.5 seconds to 55.5 seconds. The total running time for GD dropped from 131.8 seconds to 34.8 seconds. As expected, the time incurred from the serial part for both algorithms remained largely constant. The time spent on algorithm internals for EKF and GD are 27.2 seconds and 0.1 seconds for the single processor configuration. This did not change significantly for the 5 processor configurations. As the "fixed cost" for GD is much lower than EKF, the distributed computing strategy tends to be more effective for GD than for EKF. The overall running time reduction is 73.6% for GD and 63.4% for EKF. In addition, GD outperforms EKF in terms of computational performance in every trial. The comparisons can be further comprehended from figure 5-13. The empirical data supports our theoretical expectations. Although the Para-EKF and Para-GD are still very slow under the 5 processor configuration, it nevertheless tells us that distribution of the Jacobian computation is a viable approach. Last but not least, the finding might also help to further improve the PSP-GD and PSP-EKF, resulting in their respective parallel versions.

		Extended Kalman Filter (EKF)				Stochastic Gradient Descent (GD)			
		A	B	C	D	A	B	C	D
1 CPU	Trial 1	149.21	5751	2.595×10^{-2}	87.05	126.42	5961	2.121×10^{-2}	91.11
	Trial 2	149.73	5751	2.604×10^{-2}	88.58	131.34	5946	2.209×10^{-2}	93.83
	Trial 3	152.49	5751	2.651×10^{-2}	89.72	132.90	5935	2.235×10^{-2}	95.67
	Trial 4	151.95	5751	2.642×10^{-2}	90.42	131.10	5970	2.196×10^{-2}	94.23
	Trial 5	152.38	5751	2.650×10^{-2}	90.47	137.33	5966	2.302×10^{-2}	94.55
	Mean	**151.52**	**5751**	$\mathbf{2.628}\times10^{-2}$	**89.25**	**131.82**	**5956**	$\mathbf{2.213}\times10^{-2}$	**93.88**
2 CPU	Trial 1	86.58	2817	3.073×10^{-2}	44.66	60.93	2822	2.159×10^{-2}	44.00
	Trial 2	83.93	2817	2.979×10^{-2}	43.71	61.57	2841	2.167×10^{-2}	44.82
	Trial 3	85.81	2817	3.046×10^{-2}	44.26	60.86	2839	2.144×10^{-2}	45.29
	Trial 4	86.05	2817	3.055×10^{-2}	44.34	59.72	2828	2.111×10^{-2}	43.12
	Trial 5	85.14	2817	3.022×10^{-2}	44.19	68.51	2834	2.417×10^{-2}	49.39
	Mean	**85.5**	**2817**	$\mathbf{3.035}\times10^{-2}$	**44.23**	**69.32**	**2832**	$\mathbf{2.200}\times10^{-2}$	**45.32**
3 CPU	Trial 1	65.59	1962	3.343×10^{-2}	29.71	46.08	2147	2.146×10^{-2}	32.88
	Trial 2	66.85	1962	3.407×10^{-2}	30.09	47.17	2166	2.178×10^{-2}	34.39
	Trial 3	66.85	1962	3.407×10^{-2}	30.53	47.15	2164	2.179×10^{-2}	34.26
	Trial 4	67.64	1962	3.448×10^{-2}	31.36	45.28	2153	2.103×10^{-2}	32.32
	Trial 5	65.71	1962	3.349×10^{-2}	29.79	51.42	2159	2.382×10^{-2}	37.80
	Mean	**66.53**	**1962**	$\mathbf{3.391}\times10^{-2}$	**30.30**	**47.42**	**2157**	$\mathbf{2.198}\times10^{-2}$	**34.33**
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5 CPU	Trial 1	55.63	1377	4.040×10^{-2}	20.89	33.70	1562	2.157×10^{-2}	24.01
	Trial 2	55.31	1377	4.017×10^{-2}	20.56	34.73	1581	2.197×10^{-2}	24.29
	Trial 3	54.15	1377	3.932×10^{-2}	20.85	34.04	1579	2.156×10^{-2}	24.64
	Trial 4	56.80	1377	4.125×10^{-2}	22.24	33.33	1568	2.126×10^{-2}	23.49
	Trial 5	55.76	1377	4.049×10^{-2}	21.66	37.98	1574	2.413×10^{-2}	27.55
	Mean	**55.53**	**1377**	$\mathbf{4.033}\times10^{-2}$	**21.24**	**34.76**	**1572**	$\mathbf{2.210}\times10^{-2}$	**24.80**

Table 5.6: Computational statistics for distributed implementations of EKF and GD algorithms. Processor configurations of 2, 3 and 5 are compared each is repeated for 5 random seeds. Category A: Total experiment running time (sec); B: Total number of function evaluations; C: Running time per single function evaluation (sec); D: Total time of function evaluations (sec)

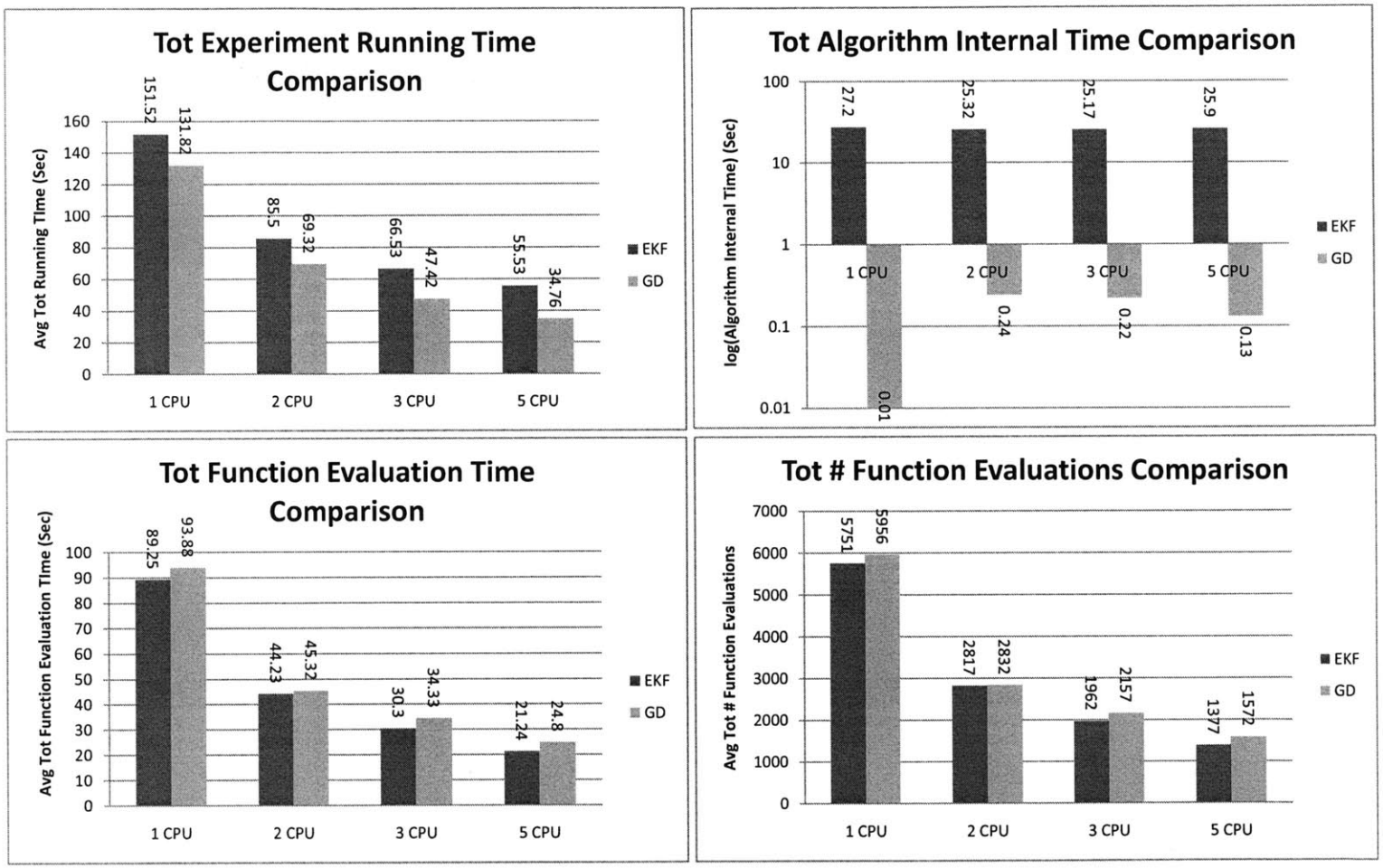

Figure 5-13: The speed performance comparison between EKF and GD in their distributed implementations

5.4 Summary

This section presented a real-world case study that exemplifies the effectiveness of the proposed on-line calibration framework and characteristics among different candidate algorithms and their implementations. We show that off-line calibration is not sensitive to traffic pattern fluctuations and may perform poorly without adequate real-time adjustments. The framework is verified on the Brisa A5 motorway in Western Portugal. Four candidate algorithms are tested in the validation experiment. The PS and EKF show most promising numerical results. The other algorithms obtained comparable performance on both counts and speeds. We show that PS and CG require many function evaluations and run much slower than EKF and GD, therefore they are not suitable for framework deployment on large-scale networks. The recommendations for the most promising algorithm on a single processor using sequential gradient approximation, are the GD and EKF algorithms.

The next part of the case study demonstrates the effectiveness of the proposed scalable framework design. We then extended our parameter state size from 187 to 638. We showed that we are able to achieve improvements over the base case using much fewer function evaluations with the application of PSP. Using PSP-EKF and PSP-GD, we are able to achieve function evaluation to number of state variable ratio of as low as 0.07. We show that the resulting prediction performance is largely comparable to that of the original EKF and GD algorithms.

On the side of distributed implementations of EKF and GD, 2, 3 and 5 CPU configurations were tested with multiple random seeds. We show that with distributed computing, we are able to reduce the *variable cost* for both algorithms considerably. However, both algorithms incur their own *fixed cost* portion that are relatively stable across different configurations. For this particular case study, we show that the GD is superior than EKF in terms computational performance. In the case of EKF, the computational burden in its matrix manipulation is still high. On the other hand, the *fixed cost* portion for GD is more than 200 times smaller. Although GD requires additional number of function evaluations, the advantage from their small algorithm

internal cost is so huge that the losts of its incremental function evaluations are almost negligible.

Chapter 6

Conclusion

Contents

6.1 Summary and Findings 104

6.2 Thesis Contribution 105

6.3 Future Research 105

6.1 Summary and Findings

In this thesis, algorithmic and implementation aspects of the DTA model on-line calibration are addressed.

The on-line calibration of DTA systems is formulated under the state-space as well as the direction optimization framework. Using simulation-based DTA models, the frameworks seek to minimize 1) derivations of unknown parameters from their a priori and 2) inconsistencies between sensory observations and their DTA-simulated counter-parts. The framework is flexible. It does not impose any restrictions on specific DTA models nor the types of sensory networks.

We outline four base-line solution algorithms aimed at verifying the feasibility of the on-line calibration framework. The state-space formulation is realized through the application of the EKF algorithm. The direct optimization formulation is realized through the application of the PS, CG and GD algorithms. Although numerically promising, we find that PS and CG are computationally burdensome. Although on average, EKF and GD requires only one function evaluation per one state variable, they become intractable when dealing with sufficiently large network, for which, one function evaluation is already very costly.

We propose two orthogonal and complementary strategies to speed up the operations of the framework. We take the two most promising candidates, the EKF and GD from the base-line algorithms, and we present their extensions to address the issue of scalability.

On a single processor, we propose the PSP-GD and PSP-EKF algorithms. The algorithms are capable of producing accurate predictions that are comparable to their base-line counter-parts: GD and EKF. However, their running time is orders of magnitude lower.

With multiple processors, we propose the Para-GD and Para-EKF algorithms. We show, empirically, that distributed computation considerably reduces the computational time compared to that of the single-processor configuration for both EKF and GD algorithms. We conclude that distributed implementation effectively reduces

the computational time for large-scale network deployment.

6.2 Thesis Contribution

This thesis makes several concrete contributions to the state-of-the-art, specifically,

1. The feasibility and effectiveness of the on-line calibration framework in a real-world application using both state-space and direct optimization formulations are demonstrated.

    - The framework is tested and successfully verified on a medium network with state variables exceeding 600. Previous on-line calibration studies used state size around 100.
    - Multiple sensory technologies are used in the testing, including Via-Verde and automatic Toll Gate Counter technologies. This demonstrates the flexibility of the framework. It does not impose any constraints on the types of sensors that it handles.

2. A scalable framework design with single processor: the PSP-EKF and PSP-GD algorithms are presented. The algorithms achieve a high degree of accuracy while maintaining low computational complexity.

3. A scalable framework design with multiple processors: Complete C++ implementation of the distributed version of the EKF (Para-EKF) and GD (Para-GD) algorithms within DynaMIT-R. A detailed computational performance analysis is presented along with the Brisa case study.

6.3 Future Research

The limitations of this study are discussed and algorithmic and application-related considerations for future research are presented in this section.

- **Additional Model Parameters:** In this thesis, the parameters for on-line calibration only includes a subset of all the parameters from a DTA model. In particular, the behavioral model parameters are left out of the on-line calibration framework. The assumption is that for real-time applications, the underlying behavior model parameters do not fluctuate rapidly within a small time period. However, this may not always be the case. [Antoniou, 2004] argued that variations in the effective behavioral parameters can be observed because of other reasons such as variations in the traffic mix. One of the examples given in [Antoniou, 2004] is the situation during major special events such as sporting events or a concert, where large amounts of traffic arrive and depart from the same location at the same time. Future research may investigate and study the flexibility of including behavior model parameters into the online calibration framework.

- **Further Experimental Analysis:** The framework is demonstrated on a non-trivial but relatively small real-world network. Further verification of the effectiveness and computational performance may need to be performed on larger networks. Moreover, the case study is conducted during non-peak hours. Additional analysis during peak traffic hours will further add credibility to this work.

- **Accurate and Robust Algorithms:** Real-time workability of the framework dictates accurate and robust algorithms. We show that accuracy and robustness are the fundamental requirements in selecting candidate algorithms. Future research will continue the selection of the most promising algorithms that are able to produce accurate and robust performance. We used the ratio between the number of function evaluations and the total number of unknown variables as an indicator for the algorithmic speed. As discussed in chapter 3, this is a very simplified approach and further analysis can adopt a comprehensive set of performance descriptors.

- **Parallelization:** While accuracy and robustness are important, the efficiency

of the framework is the key to framework scalability. When the problem is sufficiently small, we have shown that the base-line candidate algorithms are able to solve it well in a moderate amount of time. However, this may not be the case as the program grows larger. Here we differentiate two parallel schemes: 1) *Parallel implementation of calibration algorithms* and 2) *Parallel computation within the DTA model itself* [Wen, 2009]. This thesis investigates the design and realization of the former. However, a combination of the two approaches could potentially lead to extended scalable solutions. Future research might study these strategies.

- **Hybrid Parallelization and PSP:** As the state becomes large, there might be potentially many conjugate sets generated by the off-line partition process. When the number of sets becomes large, one might use parallelization techniques to process them from multiple processors and thereby reduce the total computation time required.

Bibliography

[Aerde and Rakha, 1995] Aerde, M. V. and Rakha, H. (1995). Multivariate calibration of single-regime speed-flow-density relationships. *Vehicle Navigation and Information Conference (VNIS), Piscataway, N.J.*, pages 334–341.

[Antoniou, 1997] Antoniou, C. (1997). Demand simulation for dynamic traffic assignment. Master's thesis, Massachusetts Institute of Technology.

[Antoniou, 2004] Antoniou, C. (2004). *On-line calibration for dynamic traffic assignment.* PhD thesis, Massachusetts Institute of Technology.

[Antoniou et al., 1997] Antoniou, C., Balakrishna, R., and Ben-Akiva, M. (1997). Dta system enhancement and evaluation at traffic management center, mit its laboratory. June.

[Antoniou et al., 2008] Antoniou, C., Balakrishna, R., and Koutsopoulos, H. N. (2008). Emerging data collection technologies and their impact on traffic management applications. *Proceedings of the 10th International Conference on Application of Advanced Technologies in Transportation, Athens, Greece.*

[Antoniou et al., 2004] Antoniou, C., Ben-Akiva, M., and Koutsopoulos, H. (2004). Incorporating automated vehicle identification data into origin-destination estimation. *Transportaion Research Record: Journal of the transportation research board*, 1882:37–44.

[Antoniou et al., 2005] Antoniou, C., Ben-Akiva, M., and Koutsopoulos, H. (2005). On-line calibration of traffic prediction models. *Transportation research record*, 1934:235–245.

[Antoniou et al., 2006] Antoniou, C., Ben-Akiva, M., and Koutsopoulos, H. (2006). Dynamic traffic demand prediction using conventional and emerging data sources. *IEEE proceedings intelligent transportation systems*, 153(1):97–104.

[Antoniou et al., 2007a] Antoniou, C., Ben-Akiva, M., and Koutsopoulos, H. (2007a). Non-linear kalman filtering algorithms for on-line calibration of dynamic traffic assignment models. *IEEE transactions on intelligent transportation systems*, 8(4):661–670.

[Antoniou et al., 2007b] Antoniou, C., Koutsopoulos, H. N., and Yannis, G. (2007b). An efficient non-linear kalman filtering algorithm using simultaneous perturbation and applications in traffic estimation and prediction. *Proceedings of the IEEE Intelligent Transportation Systems Conference, Seattle, WA, USA.*

[Ashok, 1996] Ashok, K. (1996). *Estimation and Prediction of Time-Dependent Origin-Destination Flows.* PhD thesis, Massachusetts Institute of Technology.

[Ashok and Ben-Akiva, 1993] Ashok, K. and Ben-Akiva, M. (1993). Dynamic o-d matrix estimation and prediction for real-time traffic management systems. *Transportation and traffic theory.*

[Ashok and Ben-Akiva, 2000] Ashok, K. and Ben-Akiva, M. (2000). Alternative approaches for real-time estimation and prediction of time-dependent origin-destination flows. *Transportation Science*, 34(1):21–36.

[Atkinson, 1989] Atkinson, K. (1989). *An Introduction to Numerical Analysis (2nd Edition).* John Wiley and Sons.

[Balakrishna, 2002] Balakrishna, R. (2002). Calibration of the demand simulator in a dynamic traffic assignment systems. Master's thesis, Massachusetts Institute of Technology.

[Balakrishna, 2006] Balakrishna, R. (2006). *Off-line calibration of Dynamic Traffic Assignment models.* PhD thesis, Massachusetts Institute of Technology.

[Ben-Akiva. et al., 2002] Ben-Akiva., M., Bierlaire, M., Koutsopoulos, H. N., and Mishalani, R. (2002). Real-time simulation of traffic demand-supply interactions within dynamit. *Transportation and network analysis: current trends*, pages 19–36.

[Ben-Akiva et al., 1991] Ben-Akiva, M., DePalma, A., and Kaysi, I. (1991). Dynamic network models and driver information systems. *Transportation Research A.*

[Bierlaire and Crittin, 2004] Bierlaire, M. and Crittin, F. (2004). An efficient algorithm for real-time estimation and prediction of dynamic od tables. *Operations Research*, 52(1).

[Bottom, 2000] Bottom, J. A. (2000). *Consistent Anticipatory Route Guidance.* PhD thesis, Massachusetts Institute of Technology.

[Brent, 1973] Brent, R. (1973). *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, NJ.

[Deb, 2005] Deb, K. (2005). *Optimization for engineering design: algorithms and examples.* Prentice-Hall.

[FHWA, 2001] FHWA (2001). Managing our congested streets and highways. *Technical report, Federal Highway Administration, US Department of Transportation, Washington, D.C. Publication No. FHWA-OP-01-018.*

[FHWA, 2005] FHWA (2005). Highway statistics 2005. *Office of Highway Policy Information - Federal Highway Administration.*

[FHWA, 2007] FHWA (2007). Highway statistics: Annual summary and reports. *Office of Highway Policy Information - Federal Highway Administration.*

[Hooke and Jeeves, 1961] Hooke, R. and Jeeves, T. (1961). direct search solutions of numerical and statistical problems. *Journal of the Association for Computing Machinery*, 8(2):212–229.

[Huang et al., 2009] Huang, E., Antoniou, C., Wen, Y., Ben-Akiva, M., Lopes, J., and Bento, J. (2009). Real-time multi-sensor multi-source network data fusion using dynamic traffic assignment models. In *proceedings to the 12th International IEEE Conference on Intelligent Transportation Systems.*

[Kiefer, 1953] Kiefer, J. (1953). Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4:502–506.

[Laborczi et al., 2002] Laborczi, P., Nowotny, B., Linauer, M., Schneider, M., Karim, R., and Leihs, D. (2002). Optimal route guidance based on floating car data. *Proceedings of the 10th world conference on transport research (WCTR), Istanbul.*

[Nooralahiyan et al., 1998] Nooralahiyan, A., Kirby, H., and McKeown, D. (1998). Vehicle classification by acoustic signature. *Mathematical and computer modeling*, 27(9):205–214.

[Oh et al., 2002] Oh, S., Ritchie, S. G., and Oh, C. (2002). Real time traffic measurement from single loop inductive signatures. *Proceedings of the 81st annual meeting of the transportation research board, Washington DC.*

[Pack et al., 2003] Pack, M. L., Smith, B. L., and Scherer, W. T. (2003). An automated camera repositioning technique for integrating video image vehicle detection systems with freway cctv systems. *Proceedings of the 82th annual meeting of the transportation research board, Washington DC.*

[Pereira et al., 2009] Pereira, J., Almeida, P., Souto, P., and R, O. (2009). Ditributed computing: Proposal for a map-i optional curricular unit on theory and foundations. Technical report, University of Minho and Univeristy of Porto.

[Pindyck and Rubinfeld, 1997] Pindyck, R. S. and Rubinfeld, D. L. (1997). Econometric models and economic forecasts. *4th edition. Irwin McGraw-Hill.*

[Press et al., 2007] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical recipes: the art of scientific computing.* Cambridge University Press.

[Quiroga et al., 2002] Quiroga, C., Perez, M., and Venglar, S. (2002). Tool for measuring travel time and delay on arterial corridors. *In ASCE 7th international conference on applications of advanced technology in transportation.*

[Spall, 1992] Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341.

[Spall, 1994a] Spall, J. C. (1994a). Developments in stochastic optimization algorithms with gradient approximations based on function measurements. *Proceedings of the 1994 Winter Simulation Conference.*

[Spall, 1994b] Spall, J. C. (1994b). A second order stochastic approximation algorithm using only function measurements. *In Proceedings of the 33rd Conference on Decision and Control, Lake Buena Vista, FL.*

[Spall, 1998a] Spall, J. C. (1998a). Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):813–823.

[Spall, 1998b] Spall, J. C. (1998b). An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest*, 19(4):482–492.

[Spall, 1999] Spall, J. C. (1999). Stochastic optimization, stochastic approximation and simulated annealing. *Wiley Encyclopedia of Electrical and Electronics Engineering*, pages 529–542.

[Srinivasan et al., 2004] Srinivasan, S., Latchman, H., Shea, J., Wong, T., and MacNair, J. (2004). Airborne traffic surveillance system - video surveillance of highway traffic. *Proceedings of the ACM VSSN 04 conference, New York.*

[Toledo et al., 2003] Toledo, T., Koutsopoulos, H. N., Davol, A., Ben-Akiva, M., Burghout, W., Andreasson, I., Johansson, T., and Lundin, C. (2003). Calibration and validation of microscopic traffic simulation tools: Stockholm case study. *Transportation Research Record*, 1831:65–75.

[van der Zijpp, 1987] van der Zijpp, N. J. (1987). Dynamic od-matrix estimation from traffic counts and automated vehicle identification data. *Transport Research Board*, 1882:37–44.

[Vaze et al., 2009] Vaze, V. S., Antoniou, C., Yang, W., and Ben-Akiva, M. (2009). Calibration of dynamic traffic assignment models with point-to-point traffic surveillance. *Transportation Research Board Annual Meeting.*

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

[Wang et al., 2007] Wang, Y., Papageorgiou, M., and Messmer, A. (2007). Real-time freeway traffic state estimation based on extended kalman filter: A case study. *Transportation Science*, 41:167–181.

[Wang et al., 2008] Wang, Y., Papageorgiou, M., and Messmer, A. (2008). Real-time freeway traffic state estimation based on extended kalman filter: Adaptive capabilities and real data testing. *Transportation Research Part A*, 42:1340–1358.

[Wen, 2009] Wen, Y. (2009). *Scalability of Dynamic Traffic Assignment.* PhD thesis, Massachusetts Institute of Technology.

[Zhou and Mahamassani, 2006] Zhou, X. and Mahamassani, H. S. (2006). Dynamic origin destination demand estimation using automatic vehicle identification data. *IEEE transactions on intelligent transportation systems*, 7(1):104–114.