Calibration of Dynamic Traffic Assignment Models with Point-to-Point Traffic Surveillance

by

Vikrant Vaze

Bachelor of Technology in Civil Engineering

Indian Institute of Technology, Bombay, India (2005)

Submitted to the Department of Civil and Environmental Engineering

In partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

Author .....
Department of Civil and Environmental Engineering
May 30, 2007

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

Calibration of Dynamic Traffic Assignment Models with Point-to-Point Traffic Surveillance

by
Vikrant Vaze

Submitted to the Department of Civil and Environmental Engineering
on May 30, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

Accurate calibration of demand and supply simulators within a Dynamic Traffic Assignment (DTA) system is critical for the provision of consistent travel information and efficient traffic management. Emerging traffic surveillance devices such as Automatic Vehicle Identification (AVI) technology provide a rich source of disaggregate traffic data. This thesis presents a methodology for calibration of demand and supply model parameters using travel time measurements obtained from these emerging traffic sensing technologies.

The calibration problem has been formulated in two different frameworks, viz. in a state-space framework and in a stochastic optimization framework. Three different algorithms are used for solving the calibration problem, a gradient approximation based path search method (SPSA), a random search meta-heuristic (GA) and a Monte-Carlo simulation based technique (Particle Filter). The methodology is first tested using a small synthetic study network to illustrate its effectiveness. Later the methodology is applied to a real traffic network in the Lower Westchester County region in New York to demonstrate its scalability. The estimation results are tested using a calibrated Microscopic Traffic Simulator (MITSIMLab). The results are compared to the base case of calibration using only the conventional point sensor data. The results indicate that the utilization of AVI data significantly improves the calibration accuracy.

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

Acknowledgements

I am highly grateful to a lot of people for making my stay at MIT such a memorable experience. I would like to thank my advisor Prof. Moshe Ben-Akiva for his guidance and inspirational presence throughout the course of my work at MIT. I would also like to express my gratitude towards Constantinos Antoniou and Yang Wen for providing incessant help related to all my transportation and IT questions. I am especially thankful to Constantinos for spending so much effort and time to ensure that I understand the concepts. I am highly indebted to Prof. Nigel Wilson for his guidance and friendly advice from time to time regarding many academic issues.

I express my deep sense of gratitude to all the ITS lab members, for being so helpful and friendly and making the stay in the lab a thoroughly enjoyable experience. I would like to especially thank Vaibhav and Varun for always being there in every good and bad experience over the last two years. Senior students in the lab including Rama, Charisma, Maya and Yang have been great sources of knowledge and helped me at every step on issues related to research, coursework and thesis writing. Charisma with her PhD comics perspective on virtually every topic, Maya with her socialization fundaes and Yang with his '*Jianzi*' coaching have been amazing. Ramachandran Balakrishna, Charisma Choudhury, Yang Wen, Anita Rao, Gunwoo Lee, Vaibhav Rathi, Varun Ramanujam, Maya Abou Zeid, Shunan Xu, Casper Chorus, Emma Frejinger and of course, Leanne Russell, have all made the ITS lab a vibrant, lively place.

I would like to thank Tanachai Limpaitoon and Onur Aydin for their friendship and support during difficult days. I am thankful to my MST classmates including Carlos, Celia, Grace, Hong Yi, Joanne, Nobuhiro, Ritesh, Tanachai and Valerina for making this journey truly *world class*. It's impossible to forget the junta2005. I would like to thank Chetan, Saumil, Parikshit, Deep, Deepanjan, Shrikanth, Ashish, Ram, Srujan, Nidhi, Harish, Sudhir and Sukant who have made the MIT life a fantastic experience.

Most importantly I would like to thank my parents and sister, without whose unwavering support, nothing could have been possible. Finally I thank god for giving me this opportunity.

Contents

List of Figures

List of Tables

1. Introduction

In the recent years, transportation problems have been attracting a lot of attention. It is becoming increasingly important to ensure that the available means of transportation continue to function efficiently with minimal impact on the environment. Road transportation is one of the most critical means of transportation today, due to its pervasiveness as well as direct impacts on its surroundings. While the traffic load on highway transportation system is increasing at a tremendous rate, the growth in highway infrastructure is far slower. Construction of new highway capacity has not kept pace with increases in population and car use and the resulting increase in demand for highway travel. In the United States, between 1980 and 1999, the total length of highways increased by 1.5 percent, while the total vehicle-miles traveled during the same period increased by 76 percent (FHWA, 2006). Building new road infrastructure is becoming increasingly difficult while the congestion costs are growing rapidly. Therefore, in the recent years, the focus has shifted from building new infrastructure to efficient utilization of existing infrastructure. Innovative means are sought to solve the growing congestion problem. Intelligent transportation system is one of the technology-oriented solutions to traffic congestion.

Intelligent Transportation System (ITS) technologies aim to make use of recent advances in the field of communication, sensing and computation to better manage the available roadway infrastructure. Traffic sensing devices such as loop detectors, video cameras, infrared sensors, Global Positioning System (GPS), radio frequency identification (RFID) tag readers etc have made the task of traffic measurement much more tractable. Traffic information devices such as Variable Message Signs (VMS), Highway Advisory Radio (HAR), in-vehicle navigation systems are now being used much more widely than before. Advances in semiconductor technology have made it possible to perform complex computations for traffic decision support at a great speed. Optimally managed highway traffic system, far from being a distant dream, is a foreseeable possibility today. Due to these technological advances, the development of highly accurate traffic prediction models and decision support systems is no longer just an academic exercise, but has the

potential to produce significant traffic improvements. Dynamic Traffic Assignment (DTA) systems are viewed as the solution to the problem of accurate traffic estimation and prediction.

1.1 Dynamic traffic assignment systems

Dynamic Traffic Assignment (DTA) systems are mathematical tools that are designed to provide decision support for traffic management and information provision. DTA systems aim to improve traffic conditions and alleviate congestion through accurate proactive forecasts of likely problems and future bottlenecks in the network. The traffic estimation and forecasts can be used by the traffic managers in two major ways. ATMS and ATIS are the tools available to traffic managers to manage traffic close to optimality.

Advanced Traffic Management Systems (ATMS) involve traffic control devices such as ramp meters, signal timings, variable speed limit signs and lane use signs to manage the flow of traffic through the supply side measures. On the other hand, Advanced Traveler Information Systems (ATIS) such as variable message signs, in-vehicle guidance units, traffic advisory through radio, television and internet can provide useful traffic information to roadway users and guide them to make more informed travel decisions for themselves, regarding destination, departure time, travel mode and route. Thus the ATIS manages the demand side of roadway networks. The DTA systems form the basis of these tools for traffic management through the management of demand and supply.

State-of-the-art DTA models have been developed in the past decade, for a variety of traffic network design, planning and operations management situations. These models employ sophisticated algorithms and detailed microscopic, macroscopic and mesoscopic simulation techniques to estimate current network performance, predict future conditions and generate traffic guidance.

Microscopic models capture the traffic dynamics through detailed representations of individual drivers and vehicular interactions. Popular commercial microscopic software

packages include CORSIM (FHWA, 2005), PARAMICS (Smith et al., 1995), AIMSUN2 (Barcelo and Casas, 2002), MITSIMLab (Yang and Koutsopoulos, 1996; Yang et al., 2000), VISSIM (PTV, 2006) and Trans-Modeler (Caliper, 2006). These models, while being highly accurate, suffer from computation-intensiveness and that obviates the possibility of employing them for real-time traffic management purposes. At the other end of the spectrum are the macroscopic models such as METANET (Messmer and Papageorgiou, 2001), EMME/2 (INRO, 2006), VISUM (PTV, 2006) and the cell transmission model (CTM, Daganzo (1994)). They model traffic flow through road segments as fluid flow through pipes. The aggregate nature of traffic modeling makes them inappropriate for accurate modeling of traveler behavior in congested situations. Mesoscopic models try to strike a balance between the computational constraints for making them useful in real-time applications, and accuracy requirements to be able to describe the network conditions with adequate level of details. Examples of such systems include Dynamic Network Assignment for the Management of Information to Travelers (DynaMIT, Ben-Akiva et al. (2001, 2002)) and DYnamic Network Assignment-Simulation Model for Advanced Road Telematics (DYNASMART, Mahmassani (2002); UMD (2005)).

1.2 Need for DTA calibration

In order to make effective traffic management decisions, the traffic managers need to know not only the current state of the system but also the predicted future states on a continuous basis. The better the knowledge about current and future state, the higher is the likelihood of effective decisions. However, in spite of the vast improvements in traffic sensing, it is impossible to measure each and every variable related to the state of the system at any point of time. While the sensors and historical knowledge about the network conditions do provide useful information, a substantial effort is necessary to estimate and predict various useful traffic performance indices which cannot be measured directly.

The DTA models provide a useful way to model the highway transportation system. However, the effectiveness of DTA models depends on the ability to replicate the network conditions accurately based on the available data. Various inputs and model parameters within the DTA system need to be set to appropriate values. Calibration is the process of assigning values to model parameters so as to replicate the traffic measurements closely.

1.3 Thesis motivation and problem statement

There is an abundance of literature related to calibration of DTA models, both the calibration of supply parameters as well as estimation of origin-destination flows. However, most of the studies in this area treat each model's parameters separately and focus on utilization of particular type of sensor data, most commonly link-flow counts. The reader is referred to Balakrishna (2006) for a detailed overview of literature related to the previous studies on calibration of specific demand or supply parameters using specific types of sensor data.

Several new types of traffic data collection technologies are being deployed in recent years, with a focus on disaggregate data collection. Different types of calibration approaches are necessary to suitably incorporate such wide variety of information. Hence, it is of value to focus on generic approaches that focus on calibration of various types of model parameters using a variety of traffic measurement technologies. A large set of these emerging technologies can be categorized as Automatic Vehicle Identification (AVI) technologies, which form the basis of this study. Note that the terms point-to-point sensor data and AVI data are sometimes used interchangeably. The two are not exactly the same and the readers are referred to section 3.1 for a discussion of the distinction between them. For the sake of simplicity, the term AVI data has been frequently used in this thesis.

Optimization formulation and state-space formulation are the two alternative frameworks that provide generalized formulation for the calibration problem and have the flexibility

to accommodate the calibration of any type of DTA model parameters using a variety of conventional and emerging traffic sensor measurement data. The calibration problem can be formulated as an optimization problem with the objective function representing a goodness-of-fit measure between observed (or apriori) values and the simulated values. The important constraints include those which express the fitted sensor measurements as a function of the calibration parameters and network characteristics. There may be other constraints providing lower and upper bound on the allowable values of the demand and supply parameters. Alternatively, the calibration problem can be formulated in a state-space framework which is described as the problem of estimating the state-vector using the measurement and transition equations. The calibration parameters constitute the state-vector describing the network conditions at any time interval. The sensor measurements provide the information necessary to estimate the state vector. The sensor measurements are functions of the state-vector values and the two are related by the indirect measurement equations. The apriori values are related to the state-vector by direct measurement equation. Further, the state-vectors across time periods are related to each other through the transition equations.

This thesis focuses on the calibration of DTA models using AVI data, under each of these two generic frameworks. A mesoscopic DTA system called DynaMIT is used to demonstrate the usefulness of the proposed methodology.

1.4 Implementation framework

As mentioned earlier, this thesis focuses on offline calibration of models and algorithm parameters within DynaMIT. A comparative analysis of three different algorithms is performed on a hypothetical network. The traffic sensor data is generated synthetically using a microscopic traffic simulator called MITSIMLab. However, in order to justify the usefulness of the approach, its effectiveness needs to be illustrated for calibration of a real traffic network with complex demand-supply interactions and large set of parameters. Therefore, a network from the Lower Westchester County (LWC) in New York State is chosen for demonstration. The LWC network has a set of loop detectors and toll booths

that together provide classified link flow counts. However, flow of AVI sensor data from the New York State Department of Transportation has yet not been established due to technical difficulties. Therefore, MITSIMLab is used as a proxy for this case study which can generate any necessary surveillance data that can then be made available for calibration of DynaMIT.

MITSIMLab is a microscopic traffic simulator that models traffic flow at the level of every individual driver. It uses behavioral algorithms and discrete choice models (Ben-Akiva and Lerman, 1985) to model decisions taken by an individual while traveling from his/her origin to his/her destination. Upon proper calibration with a network, MITSIMLab can effectively replicate the traffic characteristics. MITSIMLab is therefore a good candidate to perform the role of the simulator which is to be used as proxy for reality in the calibration of DynaMIT.

As a first requirement for this process, it is necessary for MITSIMLab, to be calibrated effectively with the traffic network under study so that it can mimic the traffic behavior effectively. DynaMIT calibration approach is then to be tested using MITSIM as a platform. The models and algorithms of DynaMIT need to be calibrated with respect to the data originating from MITSIMLab's calibrated models. MITSIMLab has been calibrated using data from the Lower Westchester County network, and the calibration process and results are described in the "Technical Memorandum on The Calibration of MITSIMLab for the Lower Westchester County Network" submitted to NYSDOT (Antoniou et al., 2006).

1.5 Thesis contribution

This thesis focuses on calibration of demand and supply parameters of DTA systems using point-to-point sensor data. Both the demand and supply parameters of the DTA system are calibrated using link counts data with and without the additional travel time information available from AVI data. Three different calibration algorithms are evaluated in terms of comparative performance. The three algorithms are Simultaneous

Perturbation Stochastic Approximation (SPSA), Genetic Algorithm (GA) and Particle Filter (PF). The application of this methodology is demonstrated using a mesoscopic DTA system called DynaMIT. The three algorithms are compared for calibration of DynaMIT using a synthetic study network. Finally, one algorithm is chosen for study of scalability to a real, large-scale, complex traffic network in the Lower Westchester County in New York State.

For both network, two calibration experiments are performed. In the first experiment, only the demand parameters are calibrated while holding supply parameters constant. In the second experiment, the demand and supply parameters are simultaneously calibrated. The following are the main findings of this thesis:

Demand calibration, by itself, was found to improve the calibration accuracy considerably as compared to the starting values. Simultaneous demand-supply calibration was found to be superior compared to the demand-only calibration and further improved the calibration accuracy. Comparison of calibration results using combination of loop detector and AVI data with the calibration results using only loop detector data indicated that the AVI data is useful to improve the calibration accuracy.

For the hypothetical network, the sensitivity analysis suggested that the relative weight given to AVI measurements is critical in determination of trade-off between the link count accuracy and travel time accuracy. While AVI data helps improve the travel time accuracy significantly, it tends to decrease the sensor count accuracy slightly. In case of the LWC network, the AVI data was found to improve the calibration accuracy both in terms of matching the sensor counts as well as travel times. This can be attributed to the fact that the sensor coverage is relatively low in large-scale network and measurement accuracy for loop detectors is lower than the AVI sensors. Hence the loop detector data by itself cannot provide all the necessary information to calibrate the large set of parameters. However, the addition of accurate AVI data aids the calibration process to move towards the true network state more efficiently, hence improving the overall calibration performance.

Based on the algorithm comparisons carried out with the small network, SPSA and GA were found to be more effective than PF algorithm. GA's calibration accuracy was comparable to SPSA. However, it came at the cost of additional computational effort. SPSA was found to be slightly better than GA in terms of accuracy and far more efficient in terms of computational effort. Therefore, SPSA was chosen for calibration of large-scale network. The calibration results with the large network reinforced the usefulness of AVI data. They also indicated that simultaneous demand supply calibration large number of parameters could be efficiently carried out using SPSA. Validation results were found to be consistent with the calibration results, which further reinforces the effectiveness of the employed methodology.

1.6 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 provides a review of previous studies related to use of AVI data in calibration as well as a review of generic DTA calibration approaches. Chapter 3 discusses various important characteristics of the information collected from AVI data sources and how it could be incorporated into DTA calibration. Chapter 4 describes the two alternative problem formulations. Chapter 4 provides a detailed discussion of various alternative solution approaches. Based on the specific characteristics of the problem and experiences from prior studies, three different algorithms are chosen as candidate methods for the calibration problem at hand. Chapter 6 demonstrates the usefulness of proposed calibration approach with an application to a hypothetical study network using synthetic data. Chapter 7 applies the framework to calibrate a large scale traffic network and illustrates the scalability of the methodology using SPSA algorithm. Chapter 8 concludes with discussion of important results and indicates some directions for future work.

2. Literature Review

This thesis focuses on the calibration of DTA models using AVI sensor data. AVI data is fundamentally different from conventional point sensor data because of the disaggregate nature and a variety of information that can be collected from it. Therefore, special methods are necessary to incorporate the AVI data into calibration effort. The first section of literature review focuses on previous calibration studies involving the use of AVI data. While the overall pool of DTA calibration literature is large, a majority of it describes methods of calibration for specific model parameters using link counts data. They are not flexible to incorporate AVI data into calibration. Therefore, the second and third section reviews previous studies to calibrate DTA models using generic traffic data types. The final section provides a summary of literature discussion in this chapter.

2.1 Use of AVI data in previous studies

Since AVI technologies are relatively recent, the previous studies using AVI data are less abundant as compared to the overall pool of DTA model calibration literature. Zietsman and Rilett (2000) proposed a disaggregate travel time estimation approach using AVI data. A fourteen-mile stretch of I-90 in Houston TX is used as the test bed. Five AVI-stations divide the freeway stretch into 4 links. The analysis period included no incidents for simplicity of analysis. The researchers identified a set of regular commuters on this highway stretch. A commuter-based disaggregate approach was compared with the aggregate approach of travel time estimation. The disaggregate approach estimated travel times and travel time variability separately for individual days and entry times. It was found that aggregation based on historic approach without considering the effect of individual days leads to considerable error when compared with individual travel times. Aggregate technique was found to be 35% less accurate than the technique that considers the effects of individual days. The authors conclude that the travel time variability should be determined on individual basis. The link travel times were found to be more variable than the corridor travel time. Significant amount of cancellation variability occurs between links across the corridor. Therefore the authors conclude that it is not necessary

to disaggregate the corridor into links if only the corridor travel times are sought. This case study was a simplified approach to using direct travel time measurements to improve the travel time estimation. But it incorporated no network effects as it used a stretch of single freeway. Most important limitation is that the analysis approach is not flexible enough to estimate any state parameters other than travel times. Nevertheless, it emphasizes the importance of disaggregate travel time information.

Van der Zijpp (1997) has proposed a constrained optimization formulation to jointly estimate the unknown OD-demand flows and identification rates. AVI data is incorporated into demand estimation in the form of partial vehicle trajectory observations. Therefore the formulation is affected by the AVI penetration rates and vehicle identification rates. Problem formulation is general enough to allow for random errors in traffic counts measurements as well as misrecognition or identification errors at AVI stations. Study network includes a single highway corridor in which no route choice alternatives exist. Case studies with synthetic data sets are performed that show a reduction in estimation error due to the usage of AVI data.

Dixon and Rilett (2000) have incorporated AVI data into the estimation of OD flows. Different methods are presented for utilizing AVI data in OD estimation and evaluating their respective benefits. They were evaluated using Generalized Least Squares method and the Kalman Filtering method. The authors evaluated sixteen different cases with the goal of evaluating the OD split proportion estimates to determine the benefits of different types of information. The cases were constructed through various combinations of method of estimation i.e. GLS or Kalman Filter and type of available data i.e. link volumes, historical OD values and AVI travel times. OD split proportions vector $b(t)$ is defined as the column vector of size equal to the number of ODs whose elements are $b_{ij}(t)$ proportions of trips departing at time t whose destination is j given that their origin is i. These OD split proportions are estimated and their closeness to the true value is used as the measure of effectiveness of each of the 16 cases. The procedure assumes that the origin flows, i.e. the total number of trips beginning from each origin is known. The OD flows are obtained by multiplying the origin flow with appropriate OD split proportion.

The study network comprised of a 20 kilometer section of the eastbound I-10 corridor leading into downtown Houston, TX together with 19 on-ramps and 22 off-ramps with no route choice involved. The results in the paper indicate that the incorporation of AVI data with link volumes is feasible and beneficial. However, the assumption of known origin flows and the lack of route choice in the simple freeway network limit the applicability of these conclusions.

Asakura et al. (2000) have provided an off-line least-squares model to simultaneously determine the OD demand and the location dependent identification rates. The AVI identification rates determine how much part of the vehicle's trajectory is identified based on the AVI observations. Assume that each link has an AVI reader. A vehicle, which travels links 1, 2, 3... 10 but is identified at links 3, 5, 6, 7 and 9 only, forms part of observed flow from link 3 to link 9. However, in reality it has its origin at the beginning of link 1 and destination at the endpoint of link 10. The following notations are used.

$Y_{1,10}$ = OD flow from link 1 to link 10

$X_{3,9}$ = observed flow from link 3 to link 9

$a_1, a_2 \ldots a_{10}$ = identification rates at the 10 AVI readers

$A_{3,9}^{1,10}$ = Percentage contribution of the true OD flow between link 1 and 10 to the observed flow between link 3 and 9 such that

$$A_{3,9}^{1,10} = (1-a_1) * (1-a_2) * a_3 * a_9 * (1-a_{10})$$

The observed and actual OD flows are related as,

$$X_{ij} = \sum_{r} \sum_{s} A_{ij}^{rs} Y_{rs} \qquad (2.1)$$

Thus the true OD matrix is estimated as follows,

$Y = A^{-1}X$

Where

$X = \{ X_{ij} \}$

$Y = \{ Y_{rs} \}$

$A = \{ A_{ij}^{rs} \}$

This least-squares model formulation was applied to Han-Shin expressway network extending 221.2 kilometers between Osaka and Kobe. This model is estimated using ordinary least-squares method. The estimated OD matrix was found to be consistent with the explicit OD survey conducted via mail.

Zhou and Mahmassani (2005) have identified two classes of demand estimation problems using vehicle identification data: 1) the estimation of tagged vehicle demand and 2) the estimation of population demand while acknowledging the fact that several of the previous attempts of utilizing AVI data focused on the first class of problems. A dynamic OD estimation method has been proposed to extract split fraction information from AVI counts without estimating the market penetration rates and identification rates of the AVI tags. The authors used a non-linear ordinary least-squares estimation model to combine AVI counts, link counts and historical demand information and solved this as an optimization problem.

If the tagged vehicles are assumed to be the representative of the whole vehicle population, then the OD split fractions calculated from the AVI data can be used as a direct measurement of the overall population's OD split values subject to a sampling error. Further, the measurement equation corresponding to the link volume measurements relates the OD flows to the link flows through link flow proportions i.e. the assignment fractions.

$$\frac{d^{tg}_{(i,k)}}{\sum_j d^{tg}_{(i,j)}} = \frac{d_{(i,k)}}{\sum_j d_{(i,j)}} + \eta_{(i,k)} \tag{2.2}$$

and

$$c_{(f)} = p_{(f),(i,k)} d_{(i,k)} + \varepsilon_{(f)} \tag{2.3}$$

Where

$d_{(i,k)}$ = Split fraction for all vehicles, for OD pair (i, k).

$d^{tg}_{(i,k)}$ = Split fraction for tagged vehicles, for OD pair (i, k).

$\eta_{(i,k)}$ = Sampling error corresponding to split fraction for OD pair (i, j).

$c_{(f)}$ = Number of vehicles on link f.

$p_{(f),(i,k)}$ = Proportion of flow for OD pair (i, j) contributing to link flow for link f.

$\varepsilon_{(f)}$ = Combined error in estimation of link flow of link f.

These equations are formulated for each time interval to create a system of nonlinear equations to estimate the unknown population OD flows $d_{(i,k)}$. A DTA simulation program called DYNASMART-P was used to first simulate the link counts and AVI counts as the 'ground-truth' and then the same simulation program was used to test the effectiveness of calibration methodology. A simplified Irvine test-bed network with around 16 OD zones, 31 nodes and 80 directed links was used for OD estimation. AVI readers were assumed to cover all entry-exit links of each OD-demand zone so that OD AVI counts were available for each OD-pair. It was noted that it is highly advantageous to place the AVI detectors so as to cover major O-D flows. Also sufficient market penetration is found to be very critical for obtaining reliable information from AVI counts.

Antoniou et al. (2004, 2006) have employed the state-space formulation for calibrating the micro-simulation model called MITSIMLab using the loop detectors as well as probe vehicle data. In this case, simple linear measurement equation is utilized assuming the apriori knowledge of true assignment vector. Also the transition equation is assumed to be linear. The estimation problem is solved using the basic Kalman filter algorithm.

Antoniou et al. (2006) have proposed a state-space formulation for incorporating the sub-path flow information into OD estimation problem. The sub-path flows are modeled as linear functions of the OD flows. The state vector includes the set of OD flows for each time interval. The Measurement equations include the measurement of link sensors counts, the measurements of sub-path flows based on the AVI data and the historical estimates of OD flows themselves. The transition equation is described by an autoregressive process where the state at a given interval depends on and is a linear function of a series of states from several previous intervals. As a result, all the measurement and transition equations are linear. Further, the measurement and estimation errors are assumed to be identical and independently normally distributed. Therefore, the Kalman Filtering method (Kalman, 1960) could be used for state estimation.

In this paper, an application of the methodology is presented through a case study. Synthetic data was generated using a microscopic traffic simulator called MITSIMLab. A simple network having 10 links and 6 OD pairs was used to demonstrate the methodology. The authors have reported that the additional AVI sub-path flow information improves the ability to accurately predict the OD flows and hence the prevalent traffic conditions. One limitation of this methodology is that it requires an assumption about the assignment fraction i.e. the assignment matrix for the network which, in itself, is difficult to estimate and is a function of several factors including travel times and OD flows themselves.

The next section describes some of the earlier efforts that formulate the calibration problem in a simulation based optimization framework and/or use optimization algorithms to solve the problem.

2.2 Stochastic optimization based studies

Mahanti (2004) has focused on the calibration of both the demand and selected supply parameters for the MITSIMLab microscopic simulator by formulating the overall optimization problem in a Generalized Least Squares (GLS) framework. The approach divides the parameter set into two groups: the OD flows and the remaining parameters (including a route choice coefficient, an acceleration/deceleration constant in the car-following model, and the mean and variance of the distribution of drivers' desired speeds relative to the speed limit). An iterative solution method is implemented, with the OD flows estimated using the classical GLS estimator, and the parameters are estimated by Box-Complex iterations. The details of the algorithm can be found in Box (1965). Balakrishna (2002) also formulated the off–line calibration framework as a large scale optimization problem where the final objective is to match simulated and observed quantities.

Gupta (2005) has demonstrated the calibration of mesoscopic DTA model called DynaMIT, wherein he uses separate methodologies to calibrate the demand and supply parameters sequentially. The supply parameters are calibrated first using the speed

density relationship for 5 separate groups of segments. Subsequently the OD estimation is performed. The OD estimation problem is formulated in a generalized least squares framework where the objective function is to minimize the sum of squares of difference between the simulated and observed quantities. The optimization problem is solved by iteratively estimating the assignment matrix and the OD flow estimates until convergence is reached between the two.

Kunde (2002) reports calibration of supply models within a mesoscopic DTA system. A three-stage approach to supply calibration is outlined, in increasing order of complexity. At the lowest disaggregate level, the individual speed-density relationship parameters for each segment are calculated using curve fitting using actual speed and density data. At the middle level, calibration is performed using sub-network where the OD flows can be accurately estimated using sensor count values, and there is little or no route choice. At the highest level full network is used for calibration. Calibration problem is formulated as a stochastic optimization problem at this level and is solved using the box-complex and SPSA algorithms. Clearly, the first stage is specific to the calibration of supply parameters in the speed-density relationship and the second stage is specific to the usage of the sensor counts data for calibration. The thesis states that the results in the third stage of calibration show that the SPSA algorithm provides comparable results to box-complex algorithm using much lesser number of function evaluations and requires much lesser run-time.

Zhou and Mahmassani (2005) used a non-linear ordinary least-squares estimation model to combine AVI counts, link counts and historical demand information and solved this as an optimization problem. The methodology they used was described in section 2.3 earlier. This approach is more flexible than other rigid sensor-counts-based approaches. However, it still incorporates AVI sub-path flows as the only additional type of traffic measurements.

Recently, Balakrishna (2006) has developed an offline DTA model calibration methodology for simultaneous demand and supply parameter estimation. This thesis used

loop detector data and historical OD flow estimates. However, the approach is easily extendable to incorporate other types of traffic sensor data. A minimization formulation is adopted and solution algorithms suitable to solve the resulting non-linear, stochastic optimization problem are identified and evaluated through detailed case studies. Two algorithms are used for optimization, viz. SPSA and SNOBFIT. The effectiveness of the approach is demonstrated using a large real traffic network. It is concluded that the two algorithms estimate comparable parameters though SPSA does so at a fraction of the computational requirements. Concerns are raised about the scalability of the SNOBFIT algorithm, while SPSA is found to be much more scalable approach. The results indicate that the simultaneous demand and supply calibration approach is much more effective than the sequential approach.

Section 2.1 has already described some earlier studies that have formulated the calibration problem in a state-space framework, especially in the context of AVI data. The next section summarizes the some further calibration studies that used the state-space formulation.

2.3 State-space formulation based studies

Ashok (1996) used the state-space formulation to model the OD estimation problem. The significant improvement over previous approaches was the use of deviations to describe the network state rather than the actual OD values. The deviations were calculated with respect to the historical OD values. The use of deviations instead of actual OD flows has two advantages. The historical estimates of OD flows subsume a wealth of information about the spatial and temporal variation of OD flows. Therefore the usage of deviations instead of actual OD flows as state vector indirectly takes into account all the experience gained from prior estimation efforts. Thus the OD estimates calculated based on the deviations retain this invaluable information. Further, the use of deviations allows the state to be represented through symmetrical distribution, especially normal distribution, which possesses desirable estimation properties.

Antoniou (2004) has calibrated the problem of online calibration of DTA models as a state–space model comprising the transition and measurement equations. Apriori values are used as direct measurements of unknown parameter such as OD estimates, mesoscopic supply parameters such as speed-density parameters and segment capacities. Surveillance information such as link counts, speeds and densities is incorporated as indirect measurement equations. A deviations based approach is used wherein the state vector is defined as the deviation of the parameters and DTA inputs from the available estimates. The measurement equation for the sensor counts is non-linear in the state variables. This obviates the usage of basic Kalman filter for calibration. Therefore extended Kalman filter, limiting extended Kalman filter and the unscented Kalman filter algorithms are used to solve the state estimation problem. This approach is demonstrated for the calibration of DTA model called DynaMIT for a real traffic network. The comparison of extended Kalman filter with the limiting version of EKF shows that the limiting EKF performed almost as effectively as the ELF but with a substantially lower computational effort. The results also indicated that the extended Kalman filter outperforms the unscented Kalman filter algorithm. However, this author has suggested that a simulation-based extension of unscented Kalman filter called particle filter could potentially provide improvement over unscented Kalman filter results.

Antoniou et al. (2004, 2006) propose a general flexible methodology for OD estimation using the new and emerging data sources. The authors try to address the problem of developing a general framework for incorporating the various types of emerging traffic sensing technologies in addition to the traditional link traffic counts. Incorporating these different technologies in estimation poses unique challenges due to their different technical characteristics including the type of collected data, measurement accuracy, levels of maturity, cost, feasibility and network coverage.

The authors employ the classic state-space technique for modeling of dynamic systems. The modeling framework includes a set of measurement equations that map the state vector on the direct or indirect measurements of the state and a set of transition equations that capture the evaluation of the state vector over successive time intervals. The state

vector is defined as the "minimal set of data sufficient to uniquely describe the dynamic behavior of the system at a time interval". However, the state vector does not include the state variable directly. This is so because often good estimates of the state variables are available apriori based on historical data or previous analysis results that embody a large amount of useful information about the state. So the authors resort to the usage of the deviations of the OD flows from the best available apriori estimates of the ODs to describe the state.

The combined measurement equation is formulated as follows,

$$Y_h = \bar{A}_h X_h + U_h \tag{2.4}$$

Where

$$Y_h = \begin{bmatrix} X_h^H \\ Y_h \\ X_h^{probe} \\ Z_h \end{bmatrix}, \quad \bar{A}_h = \begin{bmatrix} I \\ A_h \\ E_h \\ G_h \end{bmatrix}, \quad U_h = \begin{bmatrix} u_h \\ v_h \\ u_h^* \\ \eta_h \end{bmatrix}$$

Where the variable are defined for time interval h as,

X_h^H = vector of deviations of OD flows departing from their corresponding historical values.

Y_h = vector of deviations of average link flows from their best estimates.

X_h^{probe} = vector of deviations of observed probe vehicle flows from their best estimates.

Z_h = vector of deviations of sub-path flows.

I = identity matrix.

A_h = assignment matrix

E_h = diagonal 'expansion' matrix to account for the fact that the probe vehicle constitute only a fraction of the total number of vehicles in the network

G_h = matrix that maps the sub-path flows to the OD flows

u_h, v_h, u_h^*, η_h = vectors of Gaussian, zero mean, uncorrelated errors

The transition equation can be represented in matrix form in terms of deviations by an autoregressive process of degree q as follows,

$$\hat{X}_{h+1} = \sum_{p=h-q}^{h} f_h^p X_p + w_h \tag{2.5}$$

Where,

f_h^p = matrix of effects of X_p on $\hat{X}_{h+1}$

$\hat{X}_{h+1}$ = estimate of X_{h+1}

w_h = vector of Gaussian zero mean uncorrelated errors

The assignment matrix A, which maps the OD flows into the link counts, is critical for OD estimation. It depends on a large number of factors including the travel times, route choice model parameters, supply parameters such as speed density relationship parameters etc.

2.4 Summary

In summary, there is a paucity of literature regarding the utilization of AVI travel time in model calibration and the available studies mostly focus on highly simplistic models for estimating prevalent link travel times using probe vehicle travel time data. Most studies use AVI split fractions and a large majority of them restrict the estimation efforts only to origin-destination flow estimation. Majority of case studies involve OD estimation in freeway-ramp networks with very little or no route choice. Some studies have focused on the estimation of penetration and identification rates. State space formulation has been frequently used for formulating the OD estimation problem using predominantly linear models and generalized least squares and Kalman filter techniques are used for estimation.

Past research suggests that calibration problem can be formulated as an optimization problem that can incorporate a variety of traffic measurements. However, except Balakrishna (2006), all the prior studies have involved calibration of only a specific class of DTA model parameters under this framework. SPSA algorithm has been found to be promising in terms of accuracy of calibration as well as computational efforts.

The calibration problem has also been modeled under the state-space formulation. Majority of previous efforts calibration efforts have involved the usage of state-space formulation only in the context of link counts data. The only state-space formulation that incorporated AVI data (Antoniou et al., (2004, 2006)) made linearizing assumption for representation of indirect measurement equation. Extended Kalman filter is found to be an effective solution methodology. While the unscented Kalman filter did not produce

superior performance it has been suggested that the particle filter technique could potentially yield better results.

3. Automatic Vehicle Identification Technology

AVI technology is fundamentally different from the traditional loop detectors which are the most common type of traffic measurement sensors. There are various types of technologies that can be classified under the generic type called automatic vehicle identification technologies. They have varying technological attributes and accuracy and reliability characteristics. They often differ in the types of data collected. This chapter provides an overview and classification of AVI technologies from the viewpoint of DTA model calibration. The first section introduces the AVI technology. The subsequent section provides an overview of the types of data that can be utilized for calibration. The final section concludes the chapter with a summary of AVI technology characteristics.

3.1 Introduction

AVI stands for Automatic Vehicle Identification. There are several types of technologies being deployed for traffic data collection, which could be classified under the general term, AVI technologies. Majority of the discussion in this section has been derived from Antoniou et al. (2004), who provide a detailed overview of various types of technologies as well as the underlying data considerations. AVI technologies differ from other traffic data collection methods because of the disaggregate nature of collected data. Due to the identification of individual vehicles, AVI technologies capture vehicle trajectories at several points in the network.

In general, the type of data collected from an AVI technology includes 3 types of information,

a) Unique Vehicle Identifier: This is the identification code for each vehicle by which the same vehicle can be identified at various data collection points. Depending on the privacy issues involved, this identifier may sometimes be

scrambled so that it cannot be traced back to any information about the original vehicle (Antoniou et al., 2004).

b) Location: This is the location in the network e.g. link ID or road name and distance along its length, at which the vehicle was identified.

c) Time Stamp: This is the time at which the vehicle was identified.

AVI technologies can be classified based on each of these types of information although these classifications may often be overlapping. The unique vehicle identifier could be any one of the following:

a) the transponder tag installed on the vehicle, which communicates with the roadside detectors

b) license plate number, which is identified by the roadside video cameras

c) on-board GPS unit

d) cell phone of the vehicle occupant

The location of data collection could be the locations of the video cameras in case of license plate recognition or the location of RFID readers in case of the transponder tags. However, in case of GPS or cell phone based data collection, the data can be collected anywhere over the entire network, because they do not require any short range communication between the vehicle and infrastructure. For the same reason, the time of data collection for license plate recognition or transponder tags identification is the time at which the vehicle passes the location of the tag reader or video camera. However, for GPS and cell phones, the data can be collected much more frequently.

The proportion of vehicles getting identified by the AVI technology depends on the type of technology. GPS based data collection depends on the existence of on-board GPS units. Although, nowadays large proportion of vehicle occupants carry mobile phone devices, the cell-phone based data collection depends on the vehicle occupants' willingness to participate in the data collection process. Similarly, the transponder tag based data collection depends on the fraction vehicles that have the transponder tags. The video camera based license plate recognition should, at least in theory, capture the disaggregate data on all the vehicles in the network. But in practice, many of these technologies have less than 100 percent identification rates. This means that only a fraction of the vehicles, equipped with the necessary identifiers, are actually identified due to various reasons. There is another issue that could result in low identification rates. Asakura et al. (2000) have reported studies where only one of the two freeway lanes was equipped with video cameras, implying that vehicles on the other lane could not be identified.

Finally, it must be noted that the GPS and cell-phone based technologies are area-wide whereas the transponder tags and license plate recognition are point-to-point data sources. But because of practical limitations of high cost and requirement of driver participation, the area-wide techniques have so far not been practically viable on a large scale. Most of the discussion in this thesis focuses on the information collected from point-to-point data sources which has been successfully implemented in various real roadway networks. Also the data requirements for the analysis that follows are more modest than the data collected by area-wide technologies. Therefore, the discussion in this thesis is still applicable to area-wide sensors while only using a subset of the collected data.

3.2 AVI information types

Various useful measurements of traffic network performance can be derived directly from the AVI data. These include travel times, route-choice fractions, origin-destination flows, sub-path flows and actual paths used by the vehicles. It must be noted that any vehicle which is identified at only one location carries no useful information. Only the vehicles which are identified at two or more locations should be included in the analysis.

The RFID detectors and the video detection cameras are referred to by the generic term 'sensors' in the following discussion.

a) Travel times:

For the vehicles which are identified at two or more locations, the time stamps and sensor IDs of successive identifications provide the travel times between those two locations in the network.

b) Route choice fraction:

Investigation of the way that vehicles detected by one sensor are distributed among downstream sensors can provide useful information about the route choice fractions. This is especially relevant for vehicles that are identified at minimum 3 locations. For example, consider a set of vehicles that are identified at two fixed locations A and C. Some of these vehicles are identified at an intermediate location B_1 and others at another intermediate location B_2. In that case, the proportion of all the vehicles identified at A and C, which is identified at B_1 (or B_2), provides a direct estimate of the corresponding route choice fractions.

c) Origin-destination flows:

Direct estimates of origin destinations flows may be obtained provided there are sensors located sufficiently close to the origins and destinations so as to ensure that all the vehicles identified by the sensor belong to the same origin or destination.

d) Sub-path flows:

Even if the sensors are not located close to individual origins or destinations, the AVI data can provide information on how many vehicles' paths contained the sub-path between two successive sensors at which they were detected.

e) Actual paths:

AVI data can aid the generation of actual paths prior to the route choice. It gives an indication of what paths the drivers actually took.

Each of these has different implications for DTA calibration and in particular, estimation of origin-destination flows. The actual paths only aid the path generation process and cannot be directly incorporated into any numerical estimation. The travel times are a function of origin-destination flows. As the OD flows increase, the links get more and more congested and therefore the travel times increase. However, the relationship is highly complex and nonlinear. The direct measurements of OD flows and the sub-path flows can be modeled as linear functions of the actual OD flows. The route choice fractions also depend indirectly on the OD flows. However the relationship is highly nonlinear and difficult to express analytically.

Various efforts to incorporate the AVI data into OD estimation have focused on these different types of information extracted from the AVI data as has been described in section 2.1. Most of these earlier efforts focus on the sub-path flows and the direct OD measurements. But these approaches have limitations. For the direct measurement of OD flows, we need to have the AVI sensors located sufficiently close to the origins and destinations, which is often not feasible in reality. Also, in order to capture all OD flows the minimum number of required AVI sensors should equal the sum of the number of origins and the number of destinations in the network.

In case of route choice fractions, the AVI sensors need to be located in such a way that one sensor is near the beginning and the other is close to the end of the possible sub-paths

in the network under consideration. The other sensors (at least two) must be located on different possible paths between the beginning and end point. This implies that the number of sensors on each location of possible route choice necessary for the collection of any useful information must be four. This again demands a large number of sensors.

In comparison, the sub-path flows can capture information about several OD flows. As an illustration, consider the network in the figure. There are 3 origins and 3 destinations and 8 OD pairs in this simple network viz. 1-4, 1-5, 1-6, 2-4, 2-5, 2-6, 3-5 and 3-6. The two AVI sensor locations are indicated by the blue ovals. The indicated AVI sensors capture useful sub-path flow information which is a function of OD flows 1-5, 1-6, 2-5 and 2-6 but is unaffected by OD flows 1-4, 2-4, 3-5 and 3-6.

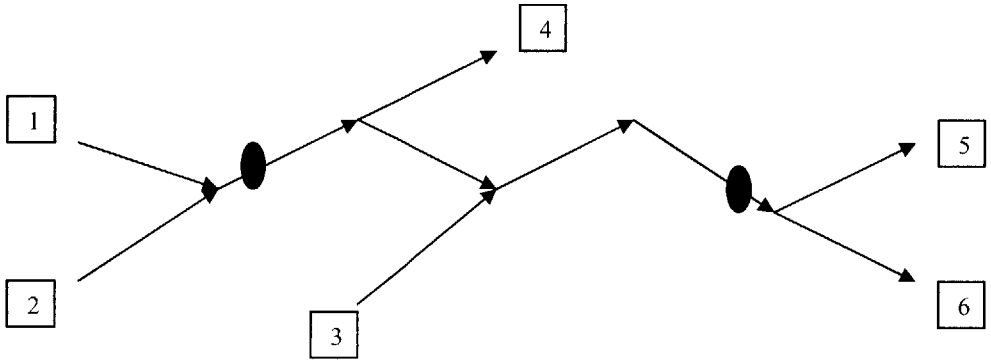

Fig 3.1: Effectiveness of AVI information

The travel time information captures the effects of an even larger set of ODs. Consider the same example again. The travel time data that can be extracted from the indicated AVI sensor pair are affected by each of the eight OD pairs in the network.

There is another advantage of travel time information over the other types of information. The usage of direct OD measurement in the estimation of population OD flow values depends on the penetration of AVI equipped vehicles, which is called the penetration rate, as well as on the identification rates of the sensors. In general, it is very difficult to predict these identification and penetration rates. Besides, they may be dependent on the origin, destination, and/or departure time intervals. Similar problem exists with the usage of sub-path flow information. However, the travel times and the route choice fractions are unlikely to be affected by the fraction of vehicles that are identified by the AVI sensors. Although in case of high penetration and identification rates, the estimates will be more reliable, the measurements of travel times and route choice fractions can be assumed to be independent of the penetration and identification rates. Note that we have ignored the possibility that there might be some correlation between the vehicles speed or route choice and whether or not it is equipped with AVI tags.

3.3 Summary

Automatic Vehicle Identification technology is a broad class of technologies involving point-to-point and area-wide technologies. Disaggregate nature of the collected data and high degree of measurement accuracy, are important features characterizing them. The AVI data contains various types of useful information about the network traffic flow including travel times, route choice fractions, OD flows and sub-path flows. Out of them, travel time is a useful and unexplored dimension of the information which is not affected by penetration and identification rates.

4. Problem Formulation

The previous chapters emphasized on the specific features of information collected by AVI technology and summarized previous calibration efforts in the literature. This chapter provides two alternative formulations of the DTA calibration problem. Each of the formulations is described in the first two sections. The final section provides a summary of the discussion related to problem formulation.

4.1 Optimization formulation

The general calibration problem involves estimation of OD flows as well as various model parameters using variety of data from sensor measurements and apriori values of OD flows and model parameters. The calibration problem can be described as an optimization problem with the objective of minimizing the goodness-of-fit measure comparing the observed and fitted measurement values. The next sub-section describes the notation to be used for describing the optimization formulation and the subsequent sub-section describes the formulation itself.

4.1.1 Notation

The following notation will be used to formulate the DTA estimation problem in the optimization framework:

T = Number of time intervals

T = 1,2, … T

x: OD flows; $x = \{x_t\} \ \forall t \in \{1, 2, \ldots T\}$

β: Model parameters; $\beta = \{\beta_t\} \ \forall t \in \{1, 2, \ldots T\}$

M^o: Observed sensor measurements; $M^o = \{ M_t^o \} \ \forall t \in \{1, 2, \ldots T\}$

x^a: Apriori OD flows; $x^a = \{ x_t^a \} \ \forall t \in \{1, 2, \ldots T\}$

β^a: Apriori model parameter values; $\beta^a = \{ \beta_t^a \} \ \forall t \in \{1, 2, \ldots T\}$

G: Road network characteristics; $G = \{G_t\} \ \forall t \in \{1, 2, \ldots T\}$

M^s: Simulated sensor measurements; $M^s = \{ M_t^s \} \ \forall t \in \{1, 2, \ldots T\}$

z: Function representing the goodness-of-fit between the observed or apriori values and the measured or true values

4.1.2 Formulation

Using these notations, the general calibration problem can be expressed as minimization of the goodness-of-fit measure z subject to constraints as follows:

$$\underset{x,\beta}{Minimize}\, z(M^s, x, \beta, M^o, x^a, \beta^a)$$

Subject to the constraints:

$$M^s = f(x, \beta, G)$$

$$l_x < x < u_x$$

$$l_\beta < \beta < u_\beta$$

The model parameters may include the trip choice model parameters of the route, mode, departure time and destination choice models on the demand side as well as the supply side parameters. The supply side parameters to be calibrated depend on the type of supply simulator. In the case of microscopic supply simulator such as MITSIMLab (Yang and Koutsopoulos, 1996 and Yang et al., 2000), they include the parameters in microscopic driver behavioral models such as acceleration-deceleration, lane changing, gap acceptance, merging models etc. In the case of a mesoscopic supply simulator such as the one in DynaMIT (Ben-Akiva et al., 2001), the supply parameters include the parameters of the speed-density curves of each segment in the network as well as segment capacities.

The abovementioned general objective function can be split into three parts. The goodness-of-fit is often calculated separately for each of the three entities i.e. the apriori model parameters, apriori OD flows and the observed sensor measurements and the overall goodness-of-fit measure is expressed as an additive function of individual fit values as follows:

$$\underset{x,\beta}{Minimize}\, z_1(\mathrm{M}^o, M^s) + z_2(x, x^a) + z_3(\beta, \beta^a)$$

The functions z_1, z_2 and z_3 represent the goodness-of-fit. The goodness-of-fit is often described by a sum of squared deviations for sensor measurements, OD flows and model parameters respectively. Note that the first part in this additive objective function can include any type of sensor measurement without any restriction. The relationship of the simulated (or fitted) sensor measurements to the calibration parameters and OD flows is highly non-linear and often closed form expressions are not available. This presents a major hurdle in solving the optimization problem described above.

4.2 State-space formulation

State-space modeling is a classic technique to handle dynamic systems. Traffic network is also a dynamic system which can be described with a set of variables that evolve over

time. The next sub-section provides background discussion about state-space formulation for dynamic systems and the subsequent sub-section illustrates how the same discussion is applicable in the specific case of a dynamic traffic network.

4.2.1 State estimation for dynamic systems

Many real-world data analysis tasks involve estimation of unknown quantities from given set of observations. These unknown quantities are usually difficult or impossible to observe directly. However, they are fundamental to the understanding and description of the system being studied. So their estimation is critical. These unknown quantities are together called as the *'State'* and an array or vector containing the values of these entities is called the *'State Vector'*. The state of a system evolves with time and hence each state vector is associated with a particular instance of time. Time may be discretized into several intervals of fixed length for ease of analysis. In general, the state vector at time *t+1* depends on state vector at time 0, 1, 2... t, along with possibly some other factors. Thus an estimate of state at previous time interval adds valuable information to the estimation of state in next interval.

The observations are measurements of entities that we can directly observe. So they are called the measurements or the indirect manifestations of the state vector. In most of these situations, some prior knowledge about the phenomenon being modeled is available. Often the observations arise sequentially in time and it is necessary to update our apriori knowledge of the unknown quantities based on the observations. These observations are dependent on the underlying unknown quantities in some way. So they provide useful information about the quantities to be estimated. However, these observations are often ridden with random noise or other factors out of our control.

4.2.2 State-space formulation for the traffic network

The problem of estimation of origin-destination flows in a traffic network can be represented as a special case of a general state-space formulation. Under the assumption that the network traffic supply parameters (such as segment capacities, speed-density curve parameters) and the route choice model parameters can be considered to be fixed, the network performance depends on the origin-destination flows. Observing the origin-destination flows is difficult because it involves either detailed OD surveys or highly sophisticated vehicle tracking technologies for observing the path of each vehicle in the network. On the other hand, what is easier to observe is a set of sensor measurements of segment flow, average speed, density and limited measurements of point-to-point flows. These measurements depend on the underlying origin-destination flows and are subject to measurement errors due to different levels of reliability of different sensor technologies. The estimation problem involves estimating the unknown OD flows from the sensor measurements. Historical OD flow information is available that serves as the apriori estimates of the state. Thus the traffic network state can be considered as a special case of the state-space formulation described above.

This framework can be extended by relaxing the assumption of fixed supply parameters and route choice parameters. The network state can be generalized to include not only the OD flows but also the speed-density relationship parameters, segments capacities and route choice parameters. In this way, a general DTA model calibration problem can be described by the state-space formulation.

The state-vector is the minimal set of data that is sufficient to uniquely describe the dynamic behavior of a discrete, stochastic, dynamic system during a time interval. Antoniou (2004) has used a state-vector comprising of origin-destination flows (x_t), parameters of the speed-density relationship model (p_t) and segment capacities (c_t) during interval t.

$$\pi_t = [x_t\ p_t\ c_t]^T = [x_t\ y_t]^T \tag{4.1}$$

The evolution of state is captured by the transition equation as which relates the state vector at interval t+1 to the state vectors of previous q intervals as:

$$\pi_{t+1} = F(\pi_t, \pi_{t-1}, \ldots, \pi_{t-q}) + \eta_t' \tag{4.2}$$

Where,

π_t : state vector at interval t

y_t : the combined vector of supply parameters; $y_t = [p_t\ c_t]^T$

η_t' : the vector of random error in respective state variables

The direct measurement equation relates the sensor measurements M_t to the state-vector π_t through the function S, which is represented by the simulator.

$$M_t = S(\pi_t) + u_t \tag{4.3}$$

Where,

u_t is the vector of random errors

Finally, the apriori estimates (π_t^a) of state variables can be included in the formulation as direct measurement equations:

$$\pi_t^a = \pi_t + v_t \tag{4.4}$$

v_t is the vector of random errors

The general form of a state-space formulation is described concisely as follows,

$$X_{t+1} = f_t(X_t) + w_t \tag{4.5}$$

$$Y_t = h_t(X_t) + u_t \tag{4.6}$$

Where,

X_t is the state-vector and Y_t is the measurement vector at time t, f_t and h_t are the transition and measurement functions while w_t and u_t are the random error components.

4.3 Summary

Calibration problem can be modeled under two alternative formulations. Optimization formulation is flexible enough to incorporate any type of traffic measurements as well as apriori estimates of model parameters and inputs, without any extra efforts. The main challenges in solving the optimization problem are the non-analytical, simulation based nature of the calibration problem as well as large size of parameter set. Alternatively, the problem could be modeled under state-space framework where the OD flows and model parameters constitute the network state at any time, to be estimated. While this approach can incorporate any type of model parameters and any type of traffic data, the complexity, and non-analytical nature of measurement equation necessitates application non-linear solution techniques.

5. Solution Algorithms

The previous chapter described each of the two general frameworks for formulation of the calibration problem. This chapter provides and overview of important solution algorithms useful for solving the calibration problem using either of these two frameworks. The first half of the chapter focuses on solution algorithms under optimization formulation. The next section begins with a description of the stochastic nature of the simulation based optimization problem at hand. Then the types of most important solution approaches are explained. The three subsequent sub-sections describe each type of solution approaches in details.

The second part of this chapter describes various solution approaches to solve the calibration problem under state-space formulation. The first section in this part focuses on the basic Kalman filter algorithm for estimation of state under the linear and Gaussian assumptions. Subsequent section describes several extensions of the basic Kalman filter that can be implemented even when the assumptions about linearity and/or Gaussian distribution are relaxed. Final section provides a summary of the discussion in this chapter and discusses the chosen approaches in more details.

5.1 Stochastic optimization approaches for simulation based systems

Any optimization problem can be represented as a minimization of an objective function subject to a set of constraints. Well-established methods and algorithms are available for solving linear optimization problems to optimality including simplex algorithm (Dantzig, 1963) and interior point methods (Karmarkar, 1984). Most of the algorithms that solve nonlinear optimization problems such as Newton's Method (Ypma, 1995), Steepest Descent Method (Arfken, 1985), Conjugate Gradient Method (Hestenes and Stiefel, 1952) make use of gradient vector of objective function with respect to the decision variables as the gradient provides useful information about the direction of cost reduction. These algorithms make simplifying assumptions about the problem structure including

analytical representation of objective function, ability to calculate the gradient vector of objective function and a deterministic setting.

Most real life problems involve a degree of uncertainty where the objective function and constraints can be represented by probabilistic distribution rather than as a single set of deterministic equations. The stochasticity exists due to stochastic models of human choice behavior, randomness in movement of stock prices, probabilistic nature of demand for a product in market etc. In many cases, the objective function to be optimized cannot be represented as an analytical closed-form function of the decision variable, e.g. dynamic traffic assignment models. Even if it is possible to express objective function analytically, it may not be possible to calculate the exact gradient vector. Stochastic optimization refers to the optimization of systems that produce output with inherent system noise. A simulation optimization problem is where the objective function and/or the constraints are implicit stochastic functions of decision variables and hence can only be evaluated by computer simulation.

This non-analytical nature of the problem precludes the possibility of differentiation or exact computation of local gradients of objective functions or constraint expressions, rendering most common optimization algorithm inadequate for solving these problems. Each evaluation of the simulation function involves considerable computational burden. Further, the best point in the solution space cannot be decided by evaluating the objective function only once at that point making the evaluation of objective function computationally very demanding. This makes the solution of stochastic optimization problems involving simulation systems a highly challenging task.

5.2 Classification of optimization algorithms

Some of these difficulties in applying the algorithms based on direct gradient measurements can be solved by using the algorithms based on gradient approximation using multiple evaluations of objective function (Spall, 1998b). Finite Difference Stochastic Approximation and Simultaneous Perturbation Stochastic Approximation are

two notable gradient approximation based optimization algorithms that search the problem space by recursively calculating the descent direction based on gradient approximation. Another class of global optimization techniques try to replicate natural processes to reach a very good although not necessarily the optimal solution, and therefore are called meta-heuristics. These techniques include Genetic Algorithms, Simulated Annealing and Ant Colony Optimization etc. Many of these meta-heuristics have been summarized in Spall (1999). Apart from these there are some methods that require neither gradient estimates nor randomization procedures. Instead they use some characteristics or pattern of the function evaluations at different points in the search space to obtain an improved point. The next three sections summarize some of the important candidate algorithms belonging to each of these types.

5.3 Path search methods

All the path search methods need an initial point in the search space to begin the search. The algorithm keeps moving the current point in a certain direction with the aim of improving the value of the objective function. Usually the gradient of the function is used directly or indirectly to determine the direction of movement. Response Surface Methodology (RSM) and Stochastic Approximation (SA) are two important families of path search methods.

5.3.1 Response surface methodology

Response surface methodology (RSM) (Kleijnen, 1987) begins at a starting point and keeps moving from point to point in the search space. It involves a polynomial approximation of the optimization surface at the current point in each iteration. The objective function is approximated by a local polynomial evaluated in the vicinity of the current parameter vector. The direction of the gradient of resulting function is calculated. The algorithm moves along this direction by some step size. The points at which the function must be evaluated are determined systematically, such as through an experimental design. Typically, a linear or quadratic response surface is chosen.

RSM is applied to the problem of simulation optimization in either in the form of meta-models or sequential procedures.

Sequential approach:

Sequential approach involves two phases. The first phase involves fitting a linear regression model around a given point, defining a linear response surface using ordinary least squares method. Then the algorithm keeps moving the point in the direction of gradient until the surface stops improving the response function value. Second phase is then implemented by fitting a quadratic model to the response. Again the algorithm keeps moving the point in the direction of gradient until the magnitude of gradient becomes sufficiently close to zero. If required, even higher order polynomials can then be utilized in analogous manner.

Meta-models:

Another implementation of RSM involves meta-models. First the set of observation points around the current solution have been identified. Then the function values at these points are used to fit a response curve or meta-model. Deterministic optimization methods are then employed to generate the meta-model's gradient and update the parameter. The reader is referred to Kleijnen (1987) for more details about the algorithm.

SNOBFIT (Huyer and Neumaier, 2004) algorithm is a recently developed extension of the RSM type algorithms. The algorithm begins with a population of several points in the search space instead of a strict path search from one starting point. The starting population is chosen based on the lower and upper bounds on decision variable values and hence it is unlikely to be affected by the choice of starting point(s) too much. This is also useful for avoiding the problem of getting stuck at a local optimum, which several of the path based algorithms may face. However, convergence rate results about SNOBFIT algorithm are currently unavailable. Balakrishna (2006) has recently used the SNOBFIT

algorithm for calibration of DTA model parameters. The reader is referred to this thesis for exact implementation details of this algorithm. The author used SNOBFIT and SPSA algorithm, which we discuss in the next section, for calibration. This thesis concludes that the two algorithms estimate comparable parameters though SPSA does so at a fraction of the computational requirements.

5.3.2 Stochastic approximation

Stochastic Approximation (SA) is another family of path search algorithms. These algorithms trace a sequence of points in the search space that ultimately converges to the point of zero gradient of the objective function. The general expression for the solution vector θ_{i+1} at the beginning of i+1th iteration of the algorithm is given by the following:

$$\theta_{i+1} = \theta_i - a_i \hat{g}(\theta_i) \tag{5.1}$$

Where,

θ_i = the solution vector at the beginning of ith iteration.

$\hat{g}(\theta_i)$ = the approximation of gradient at the ith iteration.

a_i = Step size parameter at the ith iteration. The sequence of a_i values is also known as the gain sequence. It is non-negative sequence of real numbers.

Robbins-Monro Stochastic Approximation:

The original core stochastic approximation technique that is based on direct measurement of gradient vector is known as the Robbins-Monro Stochastic Approximation (RMSA). It

is the direct analogue of the deterministic gradient based algorithms applicable for the stochastic objective functions case.

$$\theta_{i+1} = \theta_i - a_i Y(\theta_i) \tag{5.2}$$

Here, the function Y is an instance of the stochastic gradient of objective function f. So the expected value of this function corresponds to the function gradient in deterministic sense.

$$E(Y(\theta_i)) = \frac{\partial f}{\partial \theta}(\theta_i) \tag{5.3}$$

This method, as stated above, does not incorporate the constraints in the optimization problem. In other words, there is no guarantee that the new point θ_{i+1} will lie in the feasible space. The Kushner and Yin (1998) discuss about the projection operator Π_c that maps the solutions outside the constraint set C back to the nearest point inside the constraint set.

$$\theta_{i+1} = \Pi_c[\theta_i - a_i Y(\theta_i)] \tag{5.4}$$

Spall (1999) has discussed the theoretical conditions for convergence of RMSA algorithm and practical implementation details of faster convergence.

There are other techniques that do not require gradient measurements and are computationally much less burdensome. They are the Finite Difference Stochastic Approximation (FDSA) and Simultaneous Perturbation Stochastic Approximation (SPSA).

Finite Difference Stochastic Approximation (FDSA):

It involves perturbation of each component of the parameter vector separately and calculating the corresponding component of gradient using finite difference method. Each component of gradient vector is calculated by differencing the f values and then dividing by the corresponding difference in the θ_i values. Typically a two-sided gradient approximation is employed. This approach is motivated by the interpretation of gradient vector as the vector of partial derivatives of the function. Indeed, the limiting value of this gradient vector as the difference in θ_i values tends to zero equals the true gradient in the deterministic case. The j^{th} component of the gradient approximation vector is given by,

$$\hat{g}_{ij}(\hat{\theta}_i) = \frac{f(\hat{\theta}_i + c_i.e_j) - f(\hat{\theta}_i - c_i.e_j)}{2.c_j} \tag{5.5}$$

Where e_j is the vector having j^{th} component equal to 1 and every other component equal to 0.

Clearly, the calculation of each component of gradient approximation vector requires the function evaluation at 2 points. Hence the total gradient vector approximation process requires 2n functional evaluations, where n is the dimension of parameter vector. Each objective function evaluation can be highly expensive especially in case of simulation based stochastic optimization problems. Upon evaluation of function gradient, the new estimate of solution vector is computed using the equation (5.6), which is the same as equation (5.4). This requires another function evaluation.

$$\theta_{i+1} = \Pi_c[\theta_i - a_i Y(\theta_i)] \tag{5.6}$$

Thus, each iteration of FDSA requires 2n+1 function evaluations.

Simultaneous Perturbation Stochastic Approximation (SPSA):

As opposed to FDSA, SPSA involves all elements of the parameters vector being perturbed simultaneously to obtain two measurements of objective function. Then each component of the gradient approximation vector is calculated as the ratio of the difference between the two function measurements and each individual component of the perturbation vector.

$$\hat{g}_{ij}(\hat{\theta}_i) = \frac{f(\hat{\theta}_i + c_i.\Delta_i) - f(\hat{\theta}_i - c_i.\Delta_i)}{2.c_j.\Delta_{ij}} \tag{5.7}$$

Where Δ_i is the perturbation vector at the i^{th} iteration.

Again the new estimate of parameter vector is computed as before:

$$\theta_{i+1} = \Pi_c[\theta_i - a_i Y(\theta_i)] \tag{5.8}$$

Thus SPSA and FDSA differ only in the procedure of calculation of gradient. The iterate update procedure is the same. However, the gradient calculation for SPSA requires only 2 function evaluations as against 2n for the FDSA. But, as a result, the gradient approximation in SPSA is not as accurate as that in FDSA requiring additional iterations for convergence (Balakrishna 2006). However, the great computational saving in each iteration means that SPSA is computationally far more efficient. In fact, under minor conditions, Spall (1999) has shown that the overall convergence rate of SPSA tends to n times that of FDSA, for large number of iterations.

Spall (1988, 1992) has discussed the conditions for convergence of SPSA algorithm extensively. Spall (1998a) states the three important convergence conditions as follows,

1) The step sizes for both the gradient calculation and successive iterate calculations must go to zero at rates neither too fast nor too slow.

2) The objective function should be sufficiently smooth i.e. differentiable several times close to the optimal solution.

3) The components of the perturbation vector are independent and symmetrically distributed around zero with finite inverse moments.

Under these conditions Spall (1999) has proved that the best case convergence rate $i^{-1/3}$ can be obtained for SPSA.

In summary, FDSA and SPSA are the two possible candidate algorithms for solution of calibration problem using gradient approximation technique and they differ only in terms of the way the gradient is calculated. FDSA requires n times the computational effort required for SPSA per iteration (where n is the parameter vector size), which more than compensates for additional iterations required for SPSA convergence. Therefore, SPSA reaches convergence with much lower computational effort.

5.4 Pattern search methods

Pattern Search Methods do not require any gradient calculations. Hence they are termed as direct search methods. Since they do not require derivatives, they are also suitable for discrete optimization to certain extent. Some characteristic or pattern is used to obtain an improved solution at each iteration.

5.4.1 Hooke and Jeeves method

This method starts from a point in the search space and keeps moving in the direction that produces a favorable change in the objective function value. Each iteration of Hooke and Jeeves algorithm involves several sub-steps whose number equals the dimension of the parameter vector. One component of the parameter vector is perturbed in either direction by some predefined amount and the function is evaluated by holding all the other parameters constant. If a sufficient improvement is not obtained in either direction, then

the step-size is reduced and the process is repeated until a local descent direction is identified. Then the sub-step moves the current solution to this new location. This process is repeated until a search direction is identified for each component of the parameter vector. The final search direction at each iteration is a combination of each of these intermediate sub-steps.

The reader is referred to Kolda et al. (2003) for extensive description of this method. Main advantage of this method is that it does not make use of the gradient values and hence is suitable for simulation optimization problems. But there is a strong possibility of the algorithm not being able to reach the global optimum as it always looks for the local descent direction. Its strong focus on finding a pattern that reduces the objective function value at each iteration may render it susceptible to getting stuck in local optima separated by ridges. Besides, empirical evidence shows that the method performs poorly even when applied to small- or medium-sized simulation optimization problems (Kolda et al., 2003).

5.4.2 Downhill simplex method

The Nelder and Mead's Downhill Simplex Method (Nelder and Mead, 1965) is another pattern search method. A non-degenerate simplex is a geometrical figure consisting of N+1 vertices in N dimensions, where the N+1 vertices span a N-dimensional vector space. The diagram shows a two-dimensional space with a 3 dimensional simplex. The initial simplex is generated by randomly selecting N+1 points in the feasible space. The algorithm maintains a set of N+1 points at every iteration. The first step is to evaluate the objective function at each of these N+1 points. The new simplex is calculated by replacing the worst point with a new point using either reflection extension or contraction operation.

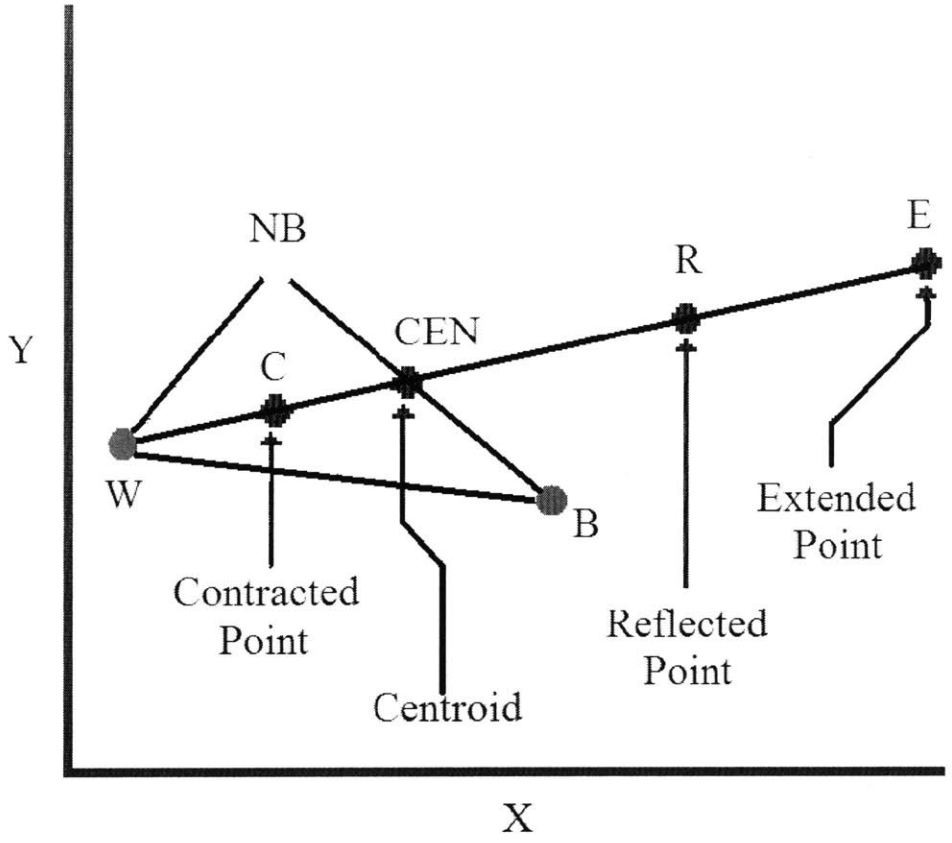

Fig 5.1: An iteration of the downhill simplex method

Consider the two-dimensional case shown in the figure. Points B, NB and W represent the best, next best and the worst point in the simplex. Let $\vec{x}_B, \vec{x}_{NB}, \vec{x}_W, \vec{x}_{CEN}, \vec{x}_R, \vec{x}_E$ and $\vec{x}_C$ be the position vectors of the points B NB, W, CEN, R, E and C respectively. Each iteration includes the following steps,

1. The position of midpoint of line segment B - NB (point CEN in the midway between points B and NB) is calculated as:

$$\vec{x}_{CEN} = \frac{\vec{x}_B + \vec{x}_{NB}}{2} \tag{5.9}$$

2. A reflection of the worst-response point W with respect to CEN is performed and the objective function is evaluated at the reflected point R. The position of the reflected point R is given by:

$$\vec{x}_R = 2 * \vec{x}_{CEN} + \vec{x}_W \tag{5.10}$$

3. If R is within the bounds of acceptable range of parameters and its function value is better than that of W but worse than that of B, then a new simplex is formed by replacing W with R and the iteration ends here.

4. If the value of objective function at R is even better, i.e. better than that of B, then this is an indication that simplex is moving in the correct direction, therefore an extension to point E is tried where point E is twice as far from CEN as R is in the same direction.

$$\vec{x}_E = 2 * \vec{x}_R + \vec{x}_{CEN} \tag{5.11}$$

5. If E is within the acceptable parameter limits and its response is better than that of R then W is replaced with E, otherwise W is replaced with R. And that particular iteration ends here.

6. If the initial reflection fails, i.e. function value at R is worse than that of W or R is not within the acceptable limits of parameters, then a contraction is performed. The contracted point C is the midpoint of line segment joining W and CEN. In this case the point C replaces W in the simplex and that iteration ends.

$$\vec{x}_{C} = \frac{\vec{x}_{W} + \vec{x}_{CEN}}{2} \tag{5.12}$$

In this way, at the end of every iteration, a new simplex, i.e. a set of (n+1) points (which in this case is a triangle) is generated and the process repeats as this set successively approaches the optimum value. The iterations are terminated when no more significant improvement of the response is observed on moving from one simplex to the other or the displacements are insignificant between successive iterations.

However, there is a dearth of strong convergence results for the downhill simplex method especially for higher dimensional cases. Proving the algorithm convergence is very difficult analytically. Besides, empirical evidence suggests that the algorithm is likely to terminate at a sub-optimal point especially in case of noisy objective function (Lagarias et al.,1998).

Box (1965) has proposed an extended version of the Nelder-Mead downhill Simplex algorithm which also maintains a set of points in N dimensional space at every iteration. However, it includes at least N+2 points in the set instead of exactly N+1 points as in the case of downhill simplex method. The use of a larger set can potentially increase the speed and accuracy of the search, and also guard against the possibilities of numerical instabilities with the Nelder-Mead approach. The reader is referred to Balakrishna (2006) for detailed exposition of the numerical problems associated with the Nelder-Mead approach. Mahanti (2004) provides a detailed description implementation details of Box-Complex algorithm for the calibration of DTA models.

However, the Box-Complex approach itself may run into some problems while dealing with simulation optimization problems. Model stochasticity may result in an apparently worst point being eliminated from the complex, when it should have been retained. Another potential drawback is related to its focus on the worst point in the complex. While the algorithm repeatedly expends effort to improve the points with the highest objective function value, an improvement to the best point is not guaranteed at every

iteration. Also, in the case where the worst point repeats itself, multiple function evaluations may be required per iteration. Hence the algorithm tends to display extremely slow convergence rates as the optimization proceeds. Therefore it is not very suitable for the purpose of simulation optimization.

In summary, a useful feature of pattern search methods such as downhill simplex method and Hooke and Jeeves method is that they do not require an evaluation or even an approximation of gradient of objective function. There is a lack of strong convergence results in case of downhill simplex while the Hooke and Jeeves method is likely to suffer from the problem of getting stuck in local optima. Besides, the danger of unintentional elimination of the best point due to stochasticity. For these reasons, the aforementioned pattern search methods may not be particularly suitable for simulation based optimization problems.

5.5 Random search methods

Random search methods do not follow a single search path. They typically maintain a large set of points at each iteration and often do not utilize all the information from previous iterations. Instead these methods adopt probabilistic mechanisms to randomly select updated parameter vectors with the hope of improving towards optimality. We review two important random search methods i.e. simulated annealing and genetic algorithms. Both are meta-heuristics which mans that they try to imitate some natural process in its search for a good solution and are not theoretically guaranteed to reach the global optimum.

5.5.1 Simulated annealing

Simulated Annealing (SA) (Metropolis et al., 1953; Corana et al., 1987) is a meta-heuristic that tries to mimic the natural process of cooling of metals. The name comes from the annealing process in metallurgy, a technique which involves heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.

The heat causes the atoms to get released from their initial positions which correspond to a local minimum of the internal energy and wander randomly through states of higher energy. A slower rate of cooling gives them more chances of finding configurations with lower internal energy than the initial one. By analogy with this physical process, each step of the SA algorithm replaces the current solution by a random nearby solution, chosen with a probability that depends on the difference between the corresponding function values and on a global parameter T (called the *temperature*). The temperature parameter is gradually decreased during the process. The dependency is such that the current solution changes almost randomly when T is large, but goes increasingly downhill as T goes to zero. The early iterations therefore allow for random jumps to escape from local optima. The allowance for uphill moves saves the method from getting stuck at local minima.

Each iteration of Simulated Annealing algorithm involves the following steps:

- Starting at x, select a random neighbor x in the neighborhood structure with probability q_{xy}: $q_{xy} > 1$; $\sum_{y \varepsilon N(x)} q_{xy} = 1$

- Move to y if $c(y) \leq c(x)$.

- If $c(y) > c(x)$, move to y with probability $e^{-(c(y)-c(x))/T}$; else stay at x only.

The abovementioned procedure is one specific way of implementing the SA algorithm. Specific details of the implementation of simulated annealing methods vary widely in the literature. However, the need for the pre-selection of a large number of tuning parameters implies that significant effort may be required in identifying their optimal values for each application. These parameters include (a) the initial temperature, (b) the distribution of the perturbation applied to randomly generate updates, (c) the cooling schedule that determines the sequence of temperatures, and (d) the criteria for lowering the temperature

(typically tied to the number of function evaluations of randomly perturbed parameter vectors allowed at each temperature setting).

The main advantage of simulated annealing is the ability to reach a global optimum due to high energy initial movements that increase the probability of reaching better solution by chance. The method is found to be effective for combinatorial optimization with discrete variables. However there are scalability issues. Goffe et al. (1994) have shown that the convergence in case of very small continuous optimization problem was found to be very slow and a very high number of function evaluations are required to achieve the optimum. Thus the performance of simulated annealing with continuous variables is not encouraging. Besides it has also been found problematic to deal with noisy measurements in the implementation of the simulated annealing algorithm.

5.5.2 Genetic algorithms

Genetic algorithms are categorized as global search heuristics. Genetic algorithms belong to a class of evolutionary algorithms that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. A population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

The chromosomes in a GA population typically take the form of bit strings i.e. sequences of 0 and 1. Each chromosome can be thought of as a point in the search space of candidate solutions. The GA processes populations of chromosomes, successively replacing one such population with another. The GA uses the objective function of the optimization problem as the fitness function that assigns a score (fitness) to each chromosome in the current population. The fitness of a chromosome depends on how well that chromosome solves the problem at hand.

GA operators

The simplest form of genetic algorithm involves three types of operators: selection, crossover, and mutation.

Selection: During each successive generation, a proportion of the existing population is selected to breed a new generation. This operator selects chromosomes in the population for reproduction. The fitter the chromosome, the more times it is likely to be selected to reproduce. Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

Crossover: This operator randomly chooses two chromosomes and exchanges the subsequences of bit strings between two chromosomes to create two offspring. For example, the strings 10000100 and 11111111 could be crossed over after the third bit in each to produce the two offspring 10011111 and 11100100. The crossover operator roughly mimics biological recombination between two single-chromosome organisms. There are several crossover techniques that are used for different applications. Apart from the one-point crossover technique indicated by the above example, there are other popular techniques including two-point crossover and "cut and splice" crossover. Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering

two child organisms. The "cut and splice" approach, results in a change in length of the children strings. The reason for this difference is that each parent string has a separate choice of crossover point.

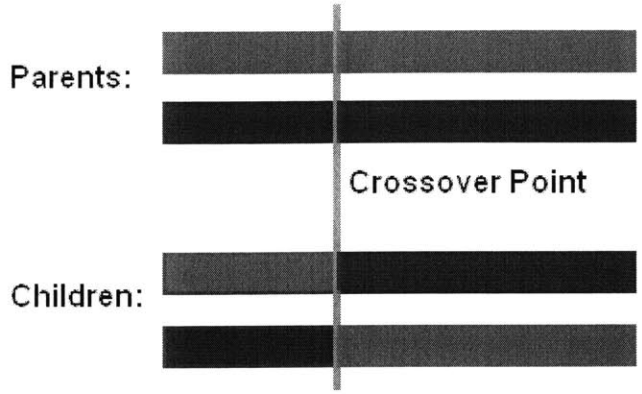

Fig 5.2: Single point crossover

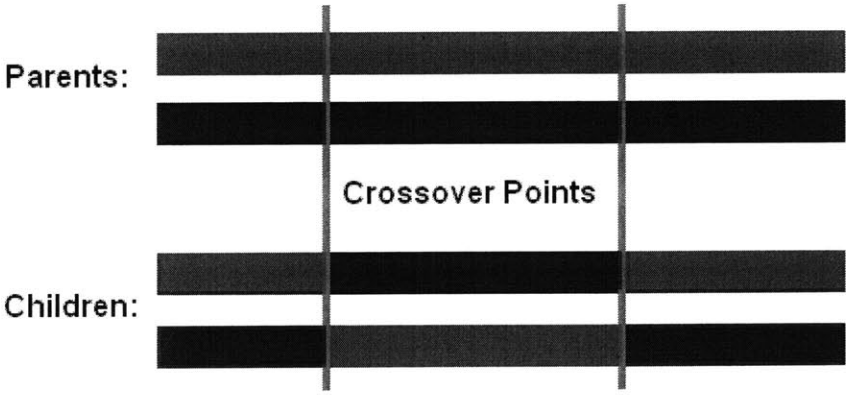

Fig 5.3: Two point crossover

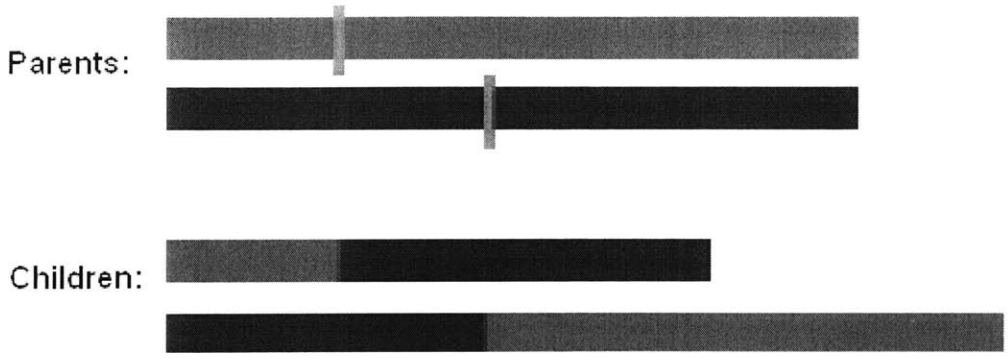

Fig 5.4: Cut and splice crossover

Mutation: This operator randomly flips some of the bits in a chromosome. For example, the string 00000100 might be mutated in its second position to yield 01000100. Mutation can occur at each bit position in a string with some probability, usually very small (e.g., 0.001). The purpose of mutation in GAs is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution.

Apart from these three operators, there exist some other operators such as inversion which are not common in practice. For detailed exposition of various types of genetic algorithm methods, implementation details and convergence issues, the reader is referred to Mitchell (1996).

The main strength of genetic algorithms is that they can easily avoid getting stuck in a local optimum. They are naturally suitable for integer optimization, especially when there are a large number of variables with small range of possible values for each. Previous studies on use of GA for transportation problems (Abdulhai et al., 1999; Lee et al., 2001; Kim, 2002; Kim and Rilett, 2003) report successful implementation of GA for calibration of various microscopic traffic simulation tools. Kim and Rilett (2004) used have employed GA for the calibration of driver behavior parameters in microscopic models CORSIM and TRANSIMS. However, the number of parameters to be calibrated was not large. Previous studies involving the usage of genetic algorithms for solving small

transportation problems (Kim and Rilett, 2004; and Henderson and Fu, 2004) indicate that a major difficulty arises from the requirement of coding the continuous variables into discrete strings. While it is difficult to code the speed-density curve parameters and segment capacities into discrete variables, the origin destination flows can be easily treated to be integers. However, this does not solve the problem completely since the number of possible integer values that each of the origin-destination flow can take is very large.

Scalability is another important issue especially when function evaluations are expensive. Henderson and Fu (2004) reviewed an application of GA for maximum likelihood estimation and explored the effect of population size and the number of generations indicating that the two quantities varied greatly depending on the search space and the nature of the objective function, and that large populations are necessary if high levels of accuracy are desired.

An important issue is the choice of suitable parameters for the genetic operators such as selection, crossover and mutation. De Jong (1975) performed an early systematic study of how varying parameters affected the GA's performance. De Jong's experiments indicated that the best population size was 50–100 individuals, the best single-point crossover rate was ~0.6 per pair of parents, and the best mutation rate was 0.001 per bit. Thierens (1999) has shown that as the problem dimension increases, the required population size to maintain similar convergence properties is found to increase at exponentially. The paper emphasizes on the fact that the choice of GA operators and the corresponding parameters is highly critical for GA's success.

In summary, simulated annealing and genetic algorithm both have the ability to reach the global optimum efficiently. Simulated annealing is noted to suffer from slow convergence even for small scale problems especially in case of noisy function measurements. Previous literature indicates instances of successful implementation of Genetic Algorithms for calibration problems in the context of transportation, where stochasticity is inherent in the problem structure. It must be acknowledged that there is no

sufficient evidence that either of these algorithms can perform efficiently for large scale problems.

5.6 Solution algorithms for state-space formulation

The solution of calibration problem modeled as a state-space formulation involves estimation of state vector. The dimension of the state vector is an important attribute of the model, largely governing the computational properties of any solution approach. The dimension of the state vector is the sum of the number of OD pairs, the total number of speed–density relationship parameters and the number of segment capacities. Most of the solution techniques for the state-space models belong to the family of Kalman Filtering Techniques. Therefore, before we proceed ahead, it would be useful to review the most basic Kalman Filtering Algorithm.

5.7 Kalman filtering algorithm

Kalman filters are based on linear dynamical systems discretized in the time domain. They are modeled on a Hidden Markov chain built on linear operators perturbed by Gaussian noise. Thus Kalman Filtering technique can be used only to solve a special case of equation (4.5) and (4.6), where the function f_t and h_t are linear given by the multiplicative matrices F_t and H_t. Also the components of the error vectors w_t and u_t follow normal distributions with zero mean and are uncorrelated to each other.

$$X_{t+1} = F_t . X_t + w_t \tag{5.13}$$

$$Y_t = H_t . X_t + u_t \tag{5.14}$$

Let Q_t and R_t be the covariance matrices of w_t and u_t respectively.

The Kalman filter has two distinct phases: prediction and update. The prediction phase uses the estimate from the previous time step to produce an estimate of the current state. In the update phase, measurement information from the current time step is used to refine this prediction to arrive at a new, more accurate estimate.

Each time interval in the estimation process corresponds to one iteration. Each iteration begins with the final estimate from the previous time interval, denoted by the subscript $t\text{-}1|t\text{-}1$. The first step in each iteration consists of prediction phase where the first estimate of the state vector and its covariance matrix for the current interval is obtained only using the data state vector and covariance matrix information from previous interval. The resulting estimates are denoted by the subscript $t|t\text{-}1$.

The second step in the iteration involves the update phase where the intermediate estimates from the prediction phase are updated using the measurement data and the final estimates for the particular time interval are calculated. These are denoted by the subscript $t|t$. The matrix K_t is called the *Kalman Gain* Matrix.

The algorithm is initialed with the apriori estimates of the state and its covariance, denoted by X_0 and P_0 respectively.

Initialization:

$$X_{0|0} = X_0 \tag{5.14}$$

$$P_{0|0} = P_0 \tag{5.16}$$

Iterations:

Prediction Phase

$$X_{t|t-1} = F_{t-1} X_{t-1|t-1} \tag{5.17}$$

$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^T + Q_t \tag{5.18}$$

Update Phase

$$K_t = P_{t|t-1} H_t^T (H_t^T P_{t|t-1} H_t^T + R_t)^{-1} \tag{5.19}$$

$$X_{t|t} = X_{t|t-1} + K_t (Y_t - H_t X_{t|t-1}) \tag{5.20}$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \tag{5.21}$$

Further information on the Kalman Filter can be found in several texts, such as Gelb (1974), Sorenson (1985), and Chui and Chen (1999).

5.8 Extensions of Kalman filtering technique

The basic Kalman filtering algorithm makes two critical assumptions about the problem formulation. First, it assumes a linear state-space model wherein both the transition and the measurement equations are linear. Second, it assumes that the error terms are uncorrelated, zero mean and normally distributed. However, many real world systems cannot be modeled under these restrictive assumptions. The linearity assumption especially hinders the generalization of Kalman filtering technique to solving a variety of problems. Hence several extensions of the original filtering technique have been proposed that handle non-linear state-space formulations. Some of the important algorithms belonging to this class are briefly reviewed in the this section. The important feature of state-space formulation of DTA calibration problem is that the measurement

equation is often non-linear. The transition equation can still be assumed to be linear. Therefore we will examine the case where the measurement equation can be represented by any general function $h_t(X_t)$ of the state vector X_t at time interval t. The transition equation will be still represented by a multiplication matrix F_t. We will use the same notations as in the basic Kalman case explained earlier. Only those aspects of these algorithms, which differ from the basic Kalman filter, will be described in details.

5.8.1 Extended Kalman filter

In extended Kalman filter (EKF) the state transition and observation models need not be linear functions of the state but may instead be any differentiable functions. It is the most straight-forward extension in which optimal quantities are approximated by linearization of the nonlinear functions in the measurement and transition equations through first order Taylor series expansion.

The algorithm is initialized in the same way as the basic Kalman filter. In case of a linear transition equation, the first phase of each iteration remains the same. The change occurs before the update phase in each of the iterations. The H_t matrix is not directly available for non-linear case. Therefore, H_t is calculated as the derivative of the function h_t.

$$H_t = \left.\frac{\partial h(x)}{\partial x}\right|_{x=X_{t|t-1}} \tag{5.22}$$

Then the update phase is proceeded as before using this new H_t matrix. Thus the H_t matrix is calculated at each iteration. This can be computationally burdensome. Plus, in case of DTA calibration problem, the measurement function $h_t(x_t)$ is calculated through a simulation run. Hence no analytical closed form expression of the derivative is possible. Therefore the derivative has to be estimated numerically. This itself requires several function evaluations. Using central derivatives, each gradient calculation requires 2n

function evaluations. Hence the extended Kalman filter technique is computationally much more expensive than the basic Kalman filter.

5.8.2 Iterated extended Kalman filter

In the update phase of the extended Kalman filter, the linearization of the measurement equation has to be performed about the present best estimate of the state vector X, i.e., $X_{t|t-1}$. However, once this step is completed, a new and presumably superior estimate $X_{t|t}$ is available which could then be used to linearize the measurement equation and repeat the update step. Several such repetitions can be performed to improve the accuracy of extended Kalman filter and the resulting filter is often called the iterated extended Kalman filter. Note that each iteration of the iterated extended Kalman filter involves the linearization step which requires at expensive function evaluations for the calculation of numerical derivatives of the function $h_t(x_t)$. Hence, the iterated extended Kalman filter algorithm is computationally several times more burdensome than the extended Kalman filter algorithm.

5.8.3 Limiting extended Kalman filter

This is not an independent estimation method by itself but rather is an approximation of extended Kalman filter algorithm. The most computationally expensive step in the extended Kalman filter is the linearization of the measurement equation. Second most expensive step is the matrix inversion involved in the calculation of the Kalman gain matrix K_t. Most of the computation time can be saved if the Kalman gain matrix K and the linearization matrix H is made available based on offline calculations. The limiting extended Kalman filter eliminates these major calculations and instead uses limiting values of K and H matrices in the update phase of each iteration. There are several ways of calculating the limiting values of K and H. One common way is just to take arithmetic average of all the available K and H matrices as the limiting K and H matrices respectively. This can improve the computational efficiency of EKF significantly.

5.8.4 Unscented Kalman filter

The Unscented Kalman Filter (ULF) (Julier et al., 1995) uses a deterministic sampling approach known as Unscented Transformation (UT) to represent a random variable using a number of deterministically selected sample points around the mean. These points are called sigma points. These points capture the mean and covariance of the random variable and, when propagated through the true nonlinear system, capture the posterior mean and covariance. In addition, this technique removes the requirement of analytical calculation of derivatives, which for complex functions can be a difficult task.

The normal distribution is approximated by generating a discrete distribution having the same first and second moments. In case of an n-dimensional Gaussian distribution, 2n+1 discrete sigma points are generated and appropriate weights are assigned to each of these to ensure the same first and second moments. However, because of the symmetry property of both the Gaussian distribution and this approximating discrete distribution, the two have the same third moments, which are equal to zero, as well. Initialization phase involves the generation of these sigma points as well as their weights.

In the prediction phase, each of the Sigma-points is propagated independently using the transition equation and a prior estimate of the state vector is calculated using a weighted average of these 2n+1 points. A prior estimate of the covariance of the state vector is calculated using the actual propagated sigma point values. Each of the sigma points is transformed using the measurement function into respective measurements and the measurement equation is computed as a weighted average of each of these individual measurement equations.

The update phase involves calculations of the covariance matrix of the measurements and the covariance between the state vector and measurements. These two are used subsequently to calculate the Kalman gain matrix which is eventually used to update the state vector and the covariance of state vector.

For details of implementation of unscented Kalman filter, the reader is referred to Antoniou et al. (2006). Since the unscented Kalman filter requires the function evaluation at 2n+1 points, it is computationally intensive. The computational performance extended Kalman filter and the unscented Kalman filter are comparable.

5.8.5 Particle filter

Particle filters (Gordon, 2003; Arulampalam et al., 2002; Pitt and Shephard, 1999) belong to the class of Monte–Carlo methods based on discrete particle representations of continuous probability density functions. These techniques require no simplifying assumptions about the state-space model. Neither linearity nor normal distribution of error terms is required for applying these methods. They can be applied to any state–space model, and generalize the traditional Kalman filtering methods.

Particle filter techniques are useful for solving the dynamic systems formulated as Hidden Markov Models (HMM). A Markov chain is a sequence of random variables π_1, π_2, π_3, ... with the Markov property, which states that given the present state, the future and past states are independent of each other.

$$Pr(\Pi_{n+1} = \pi | \Pi_n = \pi_n, ..., \Pi_1 = \pi_1) = Pr(\Pi_{n+1} = \pi | \Pi_n = \pi_n) \tag{5.23}$$

A hidden Markov model is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the values of the observables. A hidden Markov model can be considered as the simplest dynamic Bayesian network.

In a hidden Markov model, the hidden state vector $\{x_t; t \in N\}$, $x_t \in X$, is modeled as a Markov Process of initial distribution $p(x_0)$ and transition distribution $p(x_t|x_{t-1})$. The observations, $\{y_t; t \in N\}$, $y_t \in Y$, are assumed to be conditionally independent given the

process $\{x_t;\ t \in N\}$ and the marginal distribution $p(y_t|x_t)$. These probabilities are assumed to be known apriori for all $t \geq 1$.

The aim is to estimate the a posteriori distribution of state vector given the measurement vector i.e. $p(x_{0:t} \mid y_{1:t})$. At any time, the a posteriori distribution is given by the Bayes' Rule as follows:

$$p(x_{0:t} \mid y_{1:t}) = \frac{p(y_{1:t} \mid x_{0:t}).p(x_{0:t})}{\int p(y_{1:t} \mid x_{0:t}).p(x_{0:t})dx_{0:t}} \tag{5.24}$$

The left hand side of the above equation describes the a posteriori distribution of state vector $x_{0:t}$. The right hand side numerator has two terms. The first term describes the measurement distribution given the state vector. The second term is the apriori distribution of state vector. The denominator is constant for all state vector values and is essential for normalization purposes. This equation is valid for any general distributions $p(y_{1:t} \mid x_{0:t})$ and $p(x_{0:t})$. However it is, in general, extremely difficult to calculate the right hand side analytically. Especially the integral in the denominator is highly intractable even if the individual terms are known. Therefore, the particle filter uses Monte Carlo simulation techniques to evaluate the right hand side. In theory, if every possible value of state vector is considered, then the entire posteriori distribution can be described by this equation. But for a continuous distribution of state vector, the number of possible values that can be taken by even a single dimensional state vector is infinite. Hence, the particle filter simplifies the calculation by using a finite number of discrete values of state vector.

The particle filter approximates the apriori distribution of $x_{0:t}$ by a finite number of points. Each of these points is called as a '*particle*', hence the name particle filter. These particles are propagated in time and updated using the measurement values at each interval. The resulting distribution approaches the true posterior distribution of state in

the limit as the number of particles tends to infinity. The algorithm is described as follows:

Initialization:

The initial distribution of state at time zero is sampled from the apriori distribution of state at time zero

i.e.

n particles are sampled for t = 0 from $x_0 \sim p(x_0)$.

Iterations: for time $t = 1 \ldots T$

1. Propagate the particles from interval t-1 to interval t. i.e. Sample n points from $x_t \sim p(x_t|x_{t-1})$.

2. Importance weight of each particle is evaluated at time t.

$$w_{i,t} = p(y_t|\, x_{1:t}) \text{ for } i = 1 \ldots n \tag{5.25}$$

3. Importance weights are normalized by dividing each by the sum of all the weights.

$$W_{i,t} = \frac{w_{i,t}}{\sum_i w_{i,t}} \text{ for } i = 1 \ldots n \tag{5.26}$$

4. Resample n particles with replacement according to importance weights

$x_t \sim w_{i,t}$

The fourth step in each iteration involves the resampling with replacement from the existing particles. This technique is called bootstrapping. It ensures that the particle population at the beginning of every iteration will have equal weights. Hence this particular particle filter is also called *'bootstrap filter'*. At the end of N^{th} iteration, the final particle population is representative of the entire posterior distribution of state vector. Thus a simple arithmetic mean of the state vectors of each particle provides the final estimate of state vector. Other required entities can be calculated similarly form this distribution estimate. This discrete estimate of a posteriori distribution of state vector approaches the true distribution with increasing number of particles.

The particle filter requires tremendous calculation effort. However, these calculations may be performed independently for each particle in every iteration. Hence this technique is found to be suitable for parallel computing. Another problem is the repetition of particles. Several particles might be repeated due to repeated resampling with replacement after each iteration. The technique calculates the posterior estimate of state vector by averaging the individual particles. This can present problems when estimating the traffic network state since the relationship between the traffic state variables such as OD flows, speed-density curve parameters etc with the measurements e.g. sensor counts, travel time measurements etc. can be highly non-linear. So even if some of the particle themselves are good estimates for the true state vector, the average of all particles may drive it away from the true values. Besides, there is no clear way of calculating the probability of realization of a measurement vector given a particular state vector $p(y_t| x_{1:t})$. This entity can be approximated by some kind of closeness measure of the simulated measurements using the state vector $x_{1:t}$ and the actually measured value y_t. However, the effectiveness of the technique may depend on this choice.

5.9 Choice of candidate algorithms

The above discussion suggests that path search, pattern search and random search are three important classes of algorithms for calibration of simulation based systems under optimization framework. Since RMSA is not directly applicable for constrained optimization problems, FDSA and SPSA are the two candidate gradient approximation algorithms. Theoretical results indicate that SPSA reaches convergence with much lower computational effort. Besides, SPSA has been found to be highly successful in solving for large scale DTA calibration problems (Balakrishna 2006; Antoniou et al., 2007). Previous research (Balakrishna 2006) has shown that SPSA outperformed SNOBFIT, which is an RSM based approach, when applied to DTA calibration problem. Pattern search methods such as downhill simplex method and Hooke and Jeeves method do not require an approximation of objective function gradient. But the poor convergence to the global optimum and difficulties in handling stochasticity reduces their attractiveness for application to the DTA calibration problem. Simulated annealing and genetic algorithm both have the ability to reach the global optimum efficiently. However, SA may suffer from slow convergence even for small scale problems especially in case of noisy function measurements. GA, on the other hand, has been successfully applied to microscopic model calibration in the context of transportation. Therefore, SPSA and GA have been chosen as the candidate algorithms for evaluation for calibration on the test network.

For solution to state estimation problem, the basic Kalman filter is inappropriate for the non-linear problems. However, extended Kalman filter and the limiting extended Kalman filter are found to be useful for state estimation in the context of DTA calibration. Unscented Kalman filter is computationally intensive method. Previous studies indicate that while the unscented Kalman filter did not perform on par with extended Kalman filter, some of its limitations may be overcome by the simulation based extension called particle filter. Hence particle filter is chosen as the candidate algorithm for evaluation in case for calibration on test network.

5.10 Summary

The DTA calibration problem can be modeled as a large scale stochastic optimization problem. The objective function of this optimization is calculated through Monte-Carlo simulation making it extremely expensive to evaluate at each point. Several optimization algorithms have been previously used in previous studies to solve the calibration problem that can be classified into three main categories viz. path search methods, pattern search methods and random search methods. Based on the discussion in this chapter we choose the simultaneous perturbation stochastic approximation and genetic algorithms as the two most promising approaches for the calibration efforts in this study. SPSA is a path search method that has been proven to be extremely useful tools for solving simulation optimization problems. Genetic algorithm, which belongs to the category of random search methods, is also likely to be effective for the problem at hand. However, one has to be careful in choosing and fine-tuning the algorithm parameters that can be critical to the performance of these methods.

Alternatively, network state estimation problem can be modeled in the state-space framework. Most previous studies to model network as state-space formulation have attempted to solve the problem using variants of the Kalman filter algorithms. However, since the basic Kalman filter algorithm makes several restricting assumptions, various methods have been developed that extend the basic algorithm. Extended Kalman filter, iterated extended Kalman filter, limiting extended Kalman filter and unscented Kalman filter are some of the methods that have been utilized so far. Previous studies suggest that although the unscented Kalman filter algorithm has been found to perform poorly, an extension called particle filters is likely to be much more effective. The particle filter algorithm has been chosen as the solution methodology for this study. It is a sufficiently general algorithm that is not restricted to modeling only the normally distributed relationships and can incorporate any continuous probability distributions.

6. Comparative Assessment of Algorithms on Synthetic Network

So far we have described the general DTA calibration problem using traffic sensor data and apriori values. The problem has been formulated as a stochastic optimization problem as well as in the state-space framework. Previous chapter described some of the advantages and disadvantages of various potential solution techniques. Based on these and experience of other researchers in previous studies, three algorithms have been chosen for evaluation viz. Simultaneous Perturbation Stochastic Approximation (SPSA) and Genetic Algorithm (GA) for solving the optimization problem while Particle Filter (PF) for solving the state estimation problem.

This chapter demonstrates the relative performance of each of these algorithms especially for solving the calibration problem with combination of loop detector data and the AVI sensor data. A small study network has been used for this purpose. The *true* sensor values are generated by means of a microscopic traffic simulator called MITSIMLab. MITSIMLab is assumed to perform as a proxy for real world. Noise is added to the *true* sensor count values to represent reality more closely. A mesoscopic DTA model called DynaMIT has been calibrated using the sensor values generated by MITSIMLab. The results are compared with the base case where only link counts are available for calibration.

6.1 Objectives

The main objectives of this case study are as follows:

- To demonstrate the feasibility and effectiveness of the proposed DTA calibration approach involving utilization of AVI data.

- To evaluate the relative effectiveness of simultaneous demand-supply calibration compared with demand-only calibration.

- To compare the numerical accuracy and computational performance of the SPSA, Genetic Algorithm and Particle Filter algorithm and to decide the most suitable algorithm for the large case study.

- To analyze the sensitivity of calibration results to relative weights for the travel time and link count measurements; and to seek the optimum trade-off between the two.

6.2 Simulation of AVI sensors

DynaMIT is a state-of-the-art DTA system that includes mesoscopic demand and supply simulators. Modeling of the iterations between the demand and supply simulators results in the state estimation and prediction of future state for real time traffic management purposes. The demand simulator is microscopic where demand choices such as departure time, destination, mode and route choice are performed for every vehicle separately through microscopic Monte-Carlo simulation. On the other hand, the supply simulator is mesoscopic and it explicitly captures the traffic dynamics related to the development and dissipation of queues, spillbacks and congestion. The reader is referred to the description of DynaMIT simulator provided in appendix A of this thesis for more details.

DynaMIT can generate sensor counts based on the number of vehicles that cross the point sensor locations in the network in every time interval. The current version of DynaMIT does not support sensor measurements using point-to-point sensors. Therefore DynaMIT had to be modified to record the vehicles crossing each AVI sensor. The following methodology is used for this purpose.

6.2.1 DynaMIT modifications

A random vehicle ID is assigned to each vehicle at the time of its generation in the microscopic demand simulator. Each vehicle is associated with a Monte-Carlo simulated binary Bernoulli variable indicating whether it is AVI equipped or not. The expected

value of this Bernoulli variable equals the AVI penetration rate in the network. AVI penetration rate has been specified to be 30%. It should be noted that the penetration rate does not affect the calibration accuracy too much, because only the travel times are used for calibration. AVI sensor locations across the network are specified as one of the inputs to the supply simulator. Whenever a vehicle crosses an AVI sensor equipped segment in the network, the supply simulator first checks whether the vehicle is equipped with the AVI tag. If it is not, then no action is taken. If it is, then the following three things are recorded: Sensor ID, Vehicle ID and Time Stamp when the vehicle crossed the sensor. At the end of simulation, each of these records is printed in an output file, which is then used for post processing.

6.2.2 Post processing

The AVI sensor output file is used to perform post processing. All the vehicle-IDs which are recorded by only one AVI sensor are discarded. For the remaining vehicles, which are have passed by at least 2 AVI sensors in the network, all the records are arranged by increasing time-stamp of detection for each vehicle separately. Travel time between each successive pair of sensors is calculated for each vehicle. Subsequently, the average travel time between each pair of AVI sensors is computed for each estimation interval. MITSIMLab already has the capability to simulate the point-to-point sensors. Therefore, it did not require any modification. A similar post processing is performed for MITSIMLab to calculate the *true* travel time between the same sensor locations in the network.

6.3 Experimental design

This case study was performed on a small study network. The modified version of DynaMIT was used for calibration. This section explains the experimental details such as the study network structure, calibration parameters and measurement sensors.

6.3.1 Study network

A study network consisting of 10 nodes and 10 links is used for this case study. All the links are directed links and each link contains only one segment, i.e. the cross sectional characteristics remain constant across the entire length of the link. Out of 10 nodes, 4 nodes are intersections and remaining 6 denote the origins and destinations of drivers in this network. There are totally six origin destination pairs. Two out of these six involve a route choice. The simulation period is one hour, from 4:00 am to 5:00 am. It has been divided into four intervals of fifteen minutes each. Aggregated sensor data is available for these 15 minute intervals. Also the origin-destination flows are estimated for these same 15 minute intervals. Figure provides detailed description of the network.

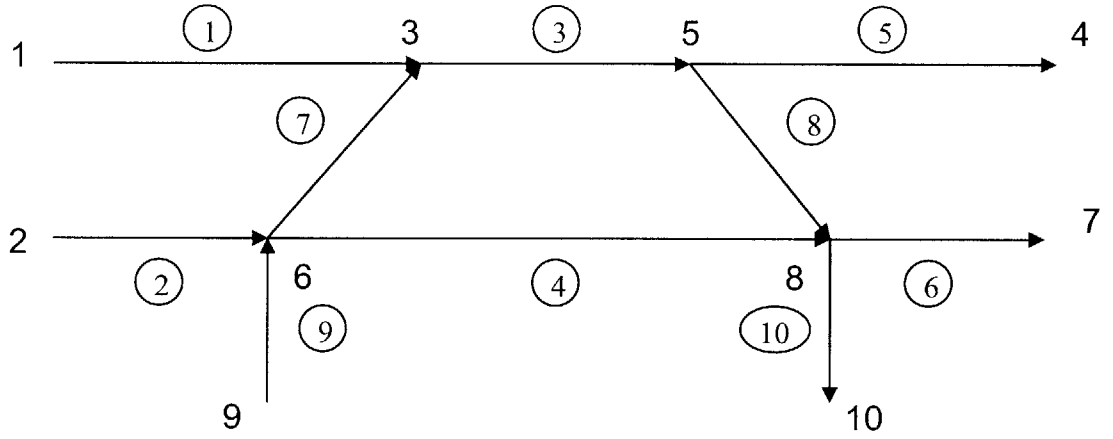

Fig 6.1: Small network topology

6.3.2 Calibration parameters

The six OD pairs are between nodes 1-4, 1-7, 1-10, 2-4, 2-7 and 9-7. The number of estimation intervals is four. Therefore, the number of parameters to be calibrated for demand-only estimation includes the 6 OD pairs for 4 intervals i.e. a total of 24 parameters. On the other hand, the simultaneous calibration of demand and supply parameters includes the additional speed-density curve parameter as well as the segment capacities. In this simple network, there are 10 segments and all of them have the same

set of supply parameters and capacities. Therefore, the number of supply parameters to be calibrated equals 7. They are as follow:

- Free flow speed (V_{max} in miles/hour)
- Jam density (K_{jam} in vehicles/meter)
- Alpha (dimensionless exponent)
- Beta (dimensionless exponent)
- Segment capacity (C in vehicles/second)
- Minimum speed (V_{min} in miles/hour) and
- Minimum density (K_{min} in vehicles/meter)

The vehicles in DynaMIT move according to the speed-density relationship given by,

$$v = \begin{cases} v_{max} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots k \le k_{min} \\ v_{max}\left[1-\left(\dfrac{k-k_{min}}{k_{jam}}\right)^{\beta}\right]^{\alpha} \ldots\ldots k > k_{min} \end{cases} \tag{6.6}$$

6.3.3 Sensors

The network contains 10 links out of which 3 are equipped with loop detectors whose measurements are aggregated into 15 minutes intervals. These are located at the center of the links numbered 2, 3 and 4. Hence they provide 3 sets of link flow counts for each interval. Thus together they provide 12 sensor measurements. Further, there are 3 AVI

sensors located at the same three locations i.e. at the center of links 2, 3 and 4. As can be seen in the fig 6.1, link 2 begins at node 2 and each vehicle that passes link 2 has to travel through either link 3 or link 4. Therefore, these 3 AVI sensors provide 2 sets of travel time measurements per time interval viz. between center of link 2 and center of link 3 and between center of link 2 and center of link 4. Thus the total number of AVI travel time measurements equals 8. Note that several AVI readings at the AVI sensor on link 3 and 4 have to be discarded as those vehicles are not recorded at link 2. They are the vehicles which originate either at node 1 or at node 9.

The data is generated using MITSIMLab Microscopic Traffic Simulator. MITSIMLab uses microscopic driver behavior models to simulate the network supply and hence the models and parameter fundamentally differ from those in DynaMIT. An overview of and MITSIMLab has been provided in the Appendix B of this thesis. MITSIMLab output is stochastic and each run of MITSIMLab produces a slightly different set of sensor measurements even if all the inputs and model parameters are exactly the same. Therefore, 10 runs of MITSIMLab simulator are performed and average sensor values are used for calibration. However, these values are not used directly. The counts data collected from the loop detectors is usually noisy and hence inaccurate. The true (average) sensor counts collected from MITSIMLab are therefore modified by a symmetrical randomly distributed additive noise ranging from -20% to +20% of the true values. These perturbed sensor values are then used for calibration. Since the AVI sensors use technology far more accurate and reliable than the loop detectors, the true counts are directly used for calibration purpose.

6.3.4 Starting parameter values

Because this case involves synthetic data, the true values of origin-destination flows are input to MITSIMLab simulator to generate the sensor measurements. To begin each algorithm, the true OD flows are perturbed randomly between -80% to +100% and used as the starting or seed values for calibration. However, the true supply parameter values are not available because the supply simulator in DynaMIT is mesoscopic and hence

differs from MITSIMLab's microscopic simulator. Hence no true values of supply parameters are available. Therefore, before the beginning of calibration, the supply parameters are calibrated using the speed density curves obtained from MITSIMLab. The optimum set of supply parameters is chosen so as to minimize the sum of squared difference between the observed (MITSIMLab) and the simulated (DynaMIT) values of vehicle speed at the same density value. This procedure is exactly the same as the first step of the three step calibration approach described in Kunde (2002). These supply parameter values, calibrated at segment level are used as the starting values of supply parameters for each of the algorithms.

6.3.5 Measures of goodness-of-fit

Five different statistics have been used to calculate the goodness of fit of the results (Toledo, 2003; Toledo and Koutsopoulos, 2004):

- Normalized root mean square error (RMSN) (Toledo and Koutsopoulos, 2004, Ashok and Ben-Akiva, 2002)
- Root mean square percent error (RMSPE) (Pindyck and Rubinfeld, 1997)
- Root mean square error (RMSE) (Pindyck and Rubinfeld, 1997)
- Normalized mean error (MEN)
- Mean percent error (MPE) (Pindyck and Rubinfeld, 1997)

The purpose of using multiple statistics is that they can capture different aspects of the obtained results. The normalized root mean square error (RMSN) and root mean square percent error (RMSPE) quantify the overall error of the simulator. These measures

penalize large errors at a higher rate than small errors. The formula for calculating the RMSN value is:

$$RMSN = \frac{\sqrt{N\sum_{n=1}^{N}\left(Y_n^s - Y_n^o\right)^2}}{\sum_{n=1}^{N} Y_n^o} \tag{6.1}$$

where N is the number of observations, Y_n^o is an observation and Y_n^s is a simulated value at time *n*.

RMSPE is calculated based on the following formula:

$$RMSPE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[\frac{Y_n^s - Y_n^o}{Y_n^o}\right]^2} \tag{6.2}$$

The Root Mean-Square Error (RMSE) is a measure of the deviation of simulated variable from its actual path. RMSE is calculated based on the following formula:

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left[Y_n^s - Y_n^o\right]^2} \tag{6.3}$$

The Normalized Mean Error (MEN) measures the mean normalized difference between simulated and average values. MEN is calculated based on the following formula:

$$MEN = \frac{\sum_{n=1}^{N}\left(Y_n^s - Y_n^o\right)}{\sum_{n=1}^{N} Y_n^o} \tag{6.4}$$

The mean percent error (MPE) statistic indicates the existence of systematic under- or over-prediction in the simulated measurements and is calculated by:

$$MPE = \frac{1}{N}\sum_{n=1}^{N}\left[\frac{Y_n^s - Y_n^o}{Y_n^o}\right] \tag{6.5}$$

Percent error measures are often preferred to their absolute error counterparts because they provide information on the magnitude of the errors relative to the average measurement.

6.4 Implementation details

The calibration is carried out in two experiments. In the first experiment, only the demand parameters are calibrated while the supply parameters are held constant. Three different algorithms are compared in terms of calibration accuracy and computational requirements. In the second experiment, both demand and supply are calibrated simultaneously. In each case, the performance is compared with the base case where only the link counts data is used for calibration.

SPSA, GA and Particle Filter are the three algorithms that were used for calibration. For each of these algorithms, several parameters need to be decided on a case-by-case basis.

6.4.1 SPSA implementation

In case of the SPSA algorithm, the j^{th} component of gradient of objective function at the i^{th} iteration is calculated as follows:

$$\hat{g}_{ij}(\hat{\theta}_i) = \frac{f(\hat{\theta}_i + c_i.\Delta_i) - f(\hat{\theta}_i - c_i.\Delta_i)}{2.c_j.\Delta_{ij}} \tag{6.7}$$

Then the next estimate of parameter vector is calculated as follows:

$$\theta_{i+1} = \Pi_c [\theta_i - a_i Y(\theta_i)] \tag{6.8}$$

The feasible region for optimization is defined by the lower and upper bounds on each of the parameter values. A simple implementation of the projection operator Π_c is used such that whenever any component of the parameter vector is higher than the upper bound or lower than the lower bound then the projection operators sets it to the upper bound and lower bound values respectively.

Here, the values of a, c, alpha and gamma are critical for good convergence. Also the distribution of perturbation vector needs to be determined.

- Alpha and gamma are critical parameters since they determine the gain sequence. The recommended values for alpha and gamma are around 0.602 and 0.101 as per the theoretical convergence conditions provided in Spall (1998b). These values were also found to be good in practice from convergence point of view.

- Various different values for parameters a and c are tried. Ultimately, c = 1.9 and a = 20 are found to be the best values.

- Spall (1998a) provides convergence conditions for the probability distribution of components of the perturbation vector. He states that they should be independent and symmetrically distributed around zero with finite inverse moments. These conditions are satisfied by a ±1 Bernoulli distribution. ±1 Bernoulli distributed variable take either the value 1 or the value -1 with certain probability. The components of perturbation vector can take values 1 and -1 with equal probability. Thus an appropriately scaled multiple of the basic ±1 Bernoulli distribution is used to describe the components of perturbation vector.

The implementation of SPSA algorithm involved computational effort equivalent to 3300 evaluations of the objective function.

6.4.2 GA implementation

In case of Genetic Algorithm, there are more complex implementation decisions to be made than just fixing the best parameter values. The choice of selection operator, crossover operators and the mutation operator need to be made apart from the decision about best parameter values. The number of particles (chromosomes) to be used is also an important decision. These decisions can affect the computational performance and convergence rate of GA tremendously. Therefore, various operators were explored before deciding upon the most suitable ones for final calibration effort.

- For the selection step, Roulette Wheel selection was found to be the best operator. In Roulette Wheel selection operator, each chromosome has probability of getting selected proportional to its fitness value. In this case the fitness value is decided to be the inverse of the objective function value obtained at that point.

- The single point crossover operator was chosen because of its simplicity and keeping in mind the fact that the problem at hand has a small number of variables which can each take a large number of values. So it was deemed unnecessary to use complicated crossover operators such as two-point crossover etc. Further, some of the operators such as cut-and-splice cannot be used at all, since they change the dimension of the decision vector which does not make sense in this case.

- Various crossover parameter values were explored. In the end 0.7 was found to be the most suitable value.

- Choice of mutation operator is one of the critical decisions since there is no direct equivalent of bit mutation in case of integer or continuous variables. A novel

approach was used for mutation where if a variable is to be mutated its value is drawn from a uniform random distribution with mean equal to the current value and the range from 0 to current value multiplied by 2.

- The value of mutation parameter is chosen to be 0.002.

- Finally, 100 particles (i.e. chromosomes) were used.

Most of the values mentioned above are close to the empirically proven values suggested by Jong (1975). He has recommended usage of 50 to 100 particles, crossover rate of 0.6 and mutation rate of 0.001 per bit.

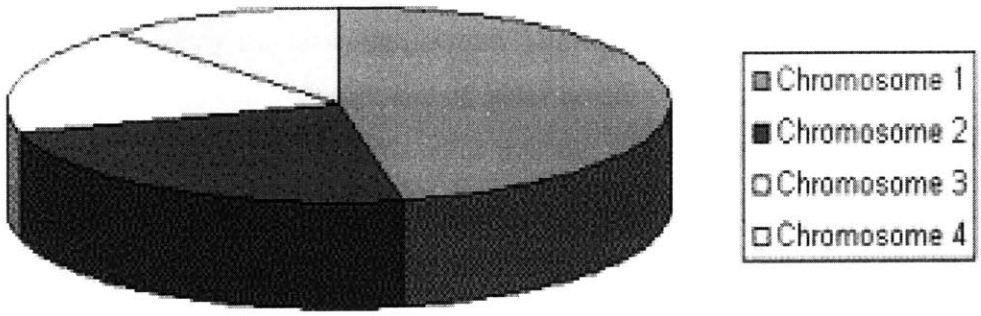

Fig 6.2: Roulette wheel selection operator

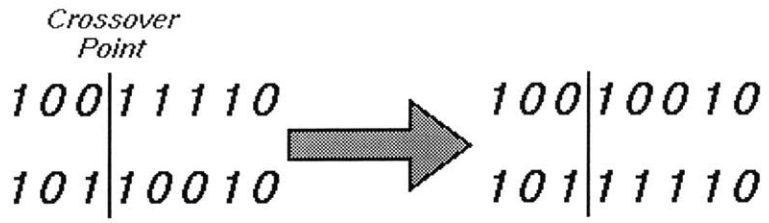

Fig 6.3: Single point crossover operator

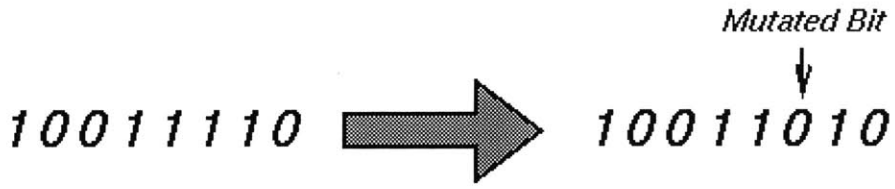

Fig 6.4: Mutation operator

The implementation of GA algorithm involved computational effort equivalent to 4500 evaluations of the objective function.

6.4.3 Particle filter implementation

In case of particle filter, the number of particles is extremely critical parameter. Because it is a simulation based estimation method, the greater the number of particles better it is. After some initial trials, 1000 particles were found to be a suitable tradeoff between the accuracy and computational effort. Initial set of particles are generated as random draws from a uniform distribution with mean centered at the apriori values of parameters. The apriori probability density function for the j^{th} component of i^{th} particle at beginning time interval is given by,

$$f_{X_{ij1}}(x) = \begin{cases} \dfrac{1}{2 * X_{ij1}^{m}} \ldots\ldots 0 \le x \le 2 * X_{ij1}^{m} \\ 0 \ldots\ldots\ldots\ldots\ldots otherwise \end{cases} \tag{6.9}$$

Here, the mean value of the j[th] component of i[th] particle at 1[st] time interval, X_{ij1}^{m} is taken to be equal to the apriori value (X_{j1}^{a}) of the same. i.e. $X_{ij1}^{m} = X_{j1}^{a}$.

In the absence of strong apriori estimates of the conditional distribution $p(x_t|x_{t-1})$, a first order auto-regressive process is assumed with and the parameters of this autoregressive process are estimated using the apriori values as shown in equation 6.12. It may be noted that each component of parameter vector at any time interval k is assumed to be dependent only the value of the same component at the previous interval (k-1) through a linear relationship.

$$X_{jk} = \alpha_{jk} X_{j(k-1)} + \varepsilon_{jk} \tag{6.10}$$

Here, α_{jk} is a coefficient of the autoregressive process while ε_{jk} is the random error term. The particles are propagated in such as way that values at next interval are drawn from a uniform distribution with a mean equal to the product of value of the same component in the previous interval and the corresponding ratio of apriori estimates.

$$f_{X_{ijk}}(x) = \begin{cases} \dfrac{1}{2 * X_{jk}^{m}} \ldots\ldots 0 \le x \le 2 * X_{jk}^{m} \\ 0 \ldots\ldots\ldots\ldots\ldots otherwise \end{cases} \tag{6.11}$$

Where, the mean value of the j[th] component of parameter vector at k[th] time interval X_{jk}^{m} is given by,

$$X_{ijk}^{m} = \frac{X_{jk}^{a}}{X_{j(k-1)}^{a}} * X_{ij(k-1)} \tag{6.12}$$

The fitness or weight of any particle is taken to be the inverse of its objective function value. This weight is used in the bootstrapping procedure at the end of each iteration in order to generate a new set of particles using random draws from the original population with replacement. This objective function value $z(X_{ik})$ is evaluated as the goodness-of-fit through simulation upto current time interval.

$$w(i) = \frac{1}{z(X_{ik})} \tag{6.13}$$

The implementation of particle filter algorithm involved computational effort equivalent to 6000 evaluations of the objective function.

6.5 Habitual link travel times

Habitual link travel times constitute one of the most important inputs to the DTA model. The individual drivers in the DTA demand simulator base their travel demand decisions on the travel times they have historically experienced. Therefore, the habitual link time table needs to be a part of the calibration process. The habitual link travel time table stores the travel times on each link by every time interval. The experienced link times are function of origin-destination flows and the supply parameters. However, they are also dependent on the habitual link times used in the simulation. Over a period of several days the habitual and experienced link times should converge such that barring the special events and minor random daily perturbations the experienced travel time should be close to the habitual travel time. In order to achieve this equilibrium, the implementation of calibration algorithms in this thesis has included a smoothing procedure wherein the habitual travel times are brought to an equilibrium with the network state vector by performing repeated simulations and taking weighted average of the experienced travel

times with the previous habitual travel times to calculate new travel time after each iteration. This process is described mathematically as follows:

$for : i = 1 : n$

$$TT_i = \lambda . TT_{exp} + (1 - \lambda) . TT_{i-1} \tag{6.14}$$

end

Where,

TT_{exp} = Experienced travel time in i^{th} iteration

TT_i = Habitual travel time in i^{th} iteration

λ = Smoothing coefficient

Too high a value of λ may lead to oscillations while too low a value may cause slow convergence. Hence the most suitable value of the smoothing coefficient (λ) has been chosen experimentally to be 0.5. The value of n is selected to be 3. Too high a value of n can cause tremendous computational burden since each iteration involves one simulation run and too low a value may lead to imbalance between habitual and experienced travel times.

In the beginning, free-flow travel times are used in the absence of better travel time estimates. Five iterations of travel time smoothing are performed in the beginning. Note that for each calibration iteration, the state vector changes. Therefore, this travel time smoothing procedure should be ideally repeated after each calibration iteration. However,

the state vector changes are not so significant after each iteration. Hence, in practice the travel time smoothing procedure is performed only once after 5 calibration iterations.

6.6 Calibration results

Two separate experiments are performed. First only the demand is calibrated while holding supply constant. In the second experiment, demand and supply are both calibrated simultaneously. Results of the first experiment are used for the design of the second experiment.

6.6.1 Demand-only calibration

The calibration result statistics are summarized in table 6.1. The RMSN values for calibration using all three algorithms are summarized in the figures 6.6, 6.7 and 6.8. In each of the cases the results were compared with the base case in which the calibration is performed with only the link counts being used as measurements.

It should be noted that the calibration results are going to be stochastic due to inherent stochasticity at two levels. First of all, there are several discrete choice and other models within DynaMIT that model the traffic demand and supply stochastically through Monte-Carlo simulation. So the DynaMIT output is noisy. In addition, all the calibration algorithms SPSA, GA and PF involve some random draws and hence they introduce additional stochasticity into the calibration process. Therefore, each run of the calibration algorithms will result in a different final solution. Hence multiple runs need to be performed in order to obtain stable calibration results. For this reason, each of the algorithm runs are performed 3 times and the results presented here are the averaged values of the three runs. These three runs start with 3 different seeds input to the randomizer in MATLAB to ensure that they result in different solutions. These three randomizer seeds themselves are generated randomly beforehand.

- From the results, it can be seen that the SPSA and GA have calibration accuracy comparable to each other while the particle filter algorithm has been found to perform significantly worse than both SPSA and GA (Fig 6.8). This is found to be the case in the base case calibration as well as the final calibration using AVI data.

- Table 6.1 summarizes various goodness-of-fit measures. TT denotes travel times. Error values are found to be comparable for GA and SPSA while all the error values for PF are in general much higher. RMSE for counts and travel times is directly proportional to the two parts of objective function in case of optimization algorithms. The last two measures, i.e. MEN and MPE denote the bias in the final calibrated values. The bias is also reduced after calibration significantly as compared to the starting values.

- Although GA has slightly better accuracy, it required considerably more function evaluations. This was not a major issue in this case due to small amount of time required for each evaluation. This could be a major issue regarding the scalability of the algorithms, especially when it come to real life traffic networks such as the one used in the next case study in this thesis.

- AVI data is found to increase the calibration accuracy, as expressed in RMSN, for all the cases as seen in figures 6.6, 6.7 and 6.8.

	Starting Values	SPSA Base	SPSA AVI	GA Base	GA AVI	PF Base	PF AVI
RMSN Counts	32.49%	8.98%	10.35%	8.49%	9.15%	16.78%	18.08%
RMSN TT	11.29%	9.52%	4.51%	10.04%	4.60%	6.73%	5.54%
RMSPE Counts	48.68%	15.77%	17.86%	15.51%	16.48%	19.40%	20.64%
RMSPE TT	7.82%	6.59%	3.84%	6.98%	3.89%	5.15%	4.39%
RMSE Counts	144.32	39.88	45.98	37.69	40.62	74.52	80.32
RMSE TT	1.09	0.92	0.43	0.97	0.44	0.65	0.53
MEN Counts	25.97%	3.85%	1.47%	1.63%	0.32%	1.99%	-3.58%
MEN TT	5.49%	2.86%	1.51%	1.33%	1.69%	5.13%	1.80%
MPE Counts	37.30%	7.79%	5.34%	5.51%	1.38%	-3.53%	-7.50%
MPE TT	3.51%	1.96%	1.31%	1.10%	1.39%	3.33%	1.54%

Table 6.1: Accuracy of demand-only calibration

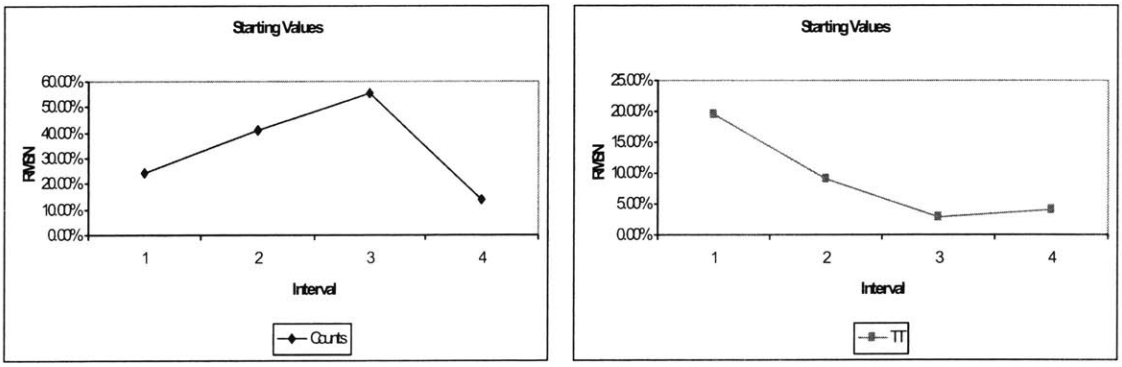

Fig 6.5: RMSN of starting parameter values

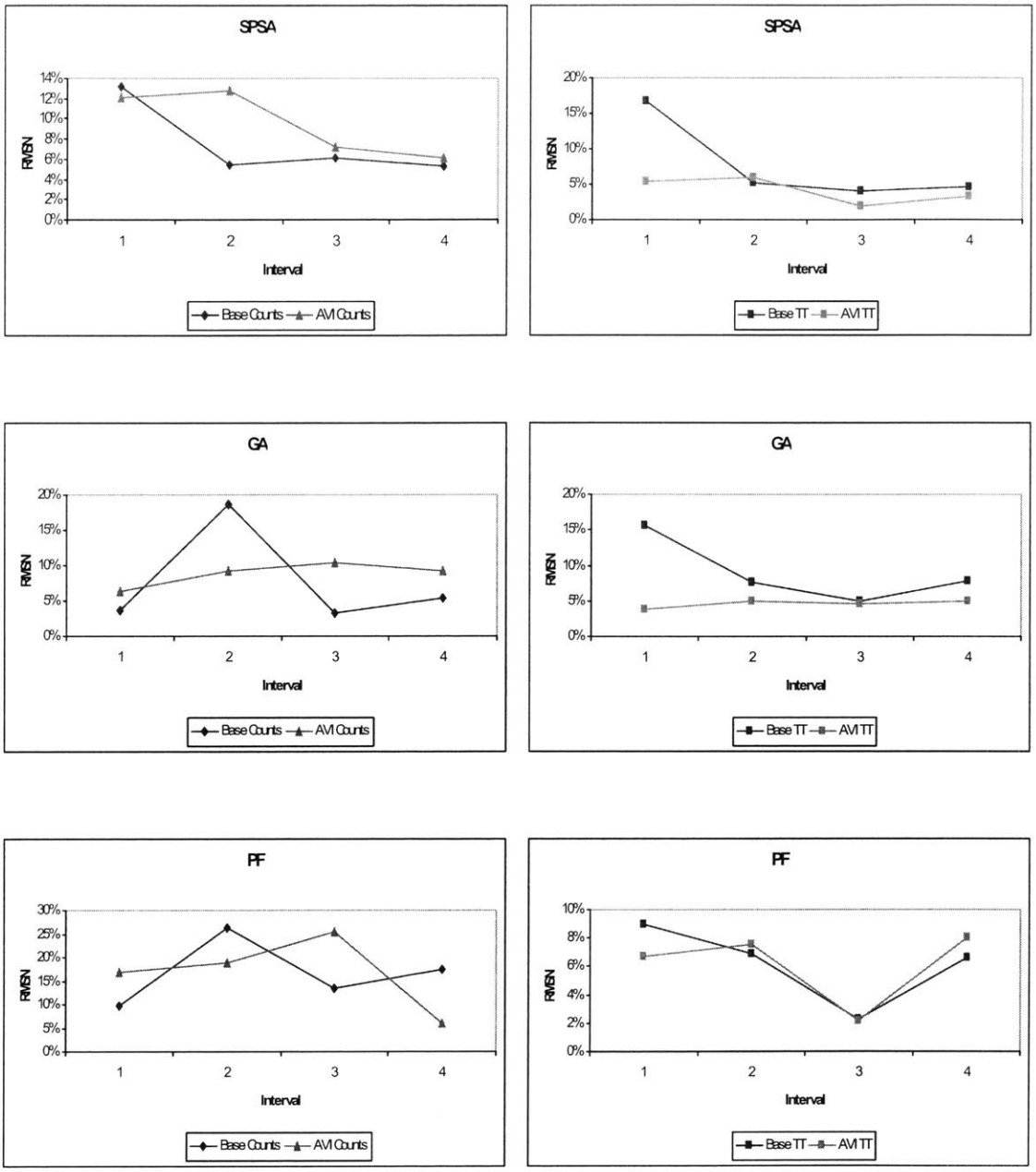

Fig 6.6: Calibration results: Demand-only calibration

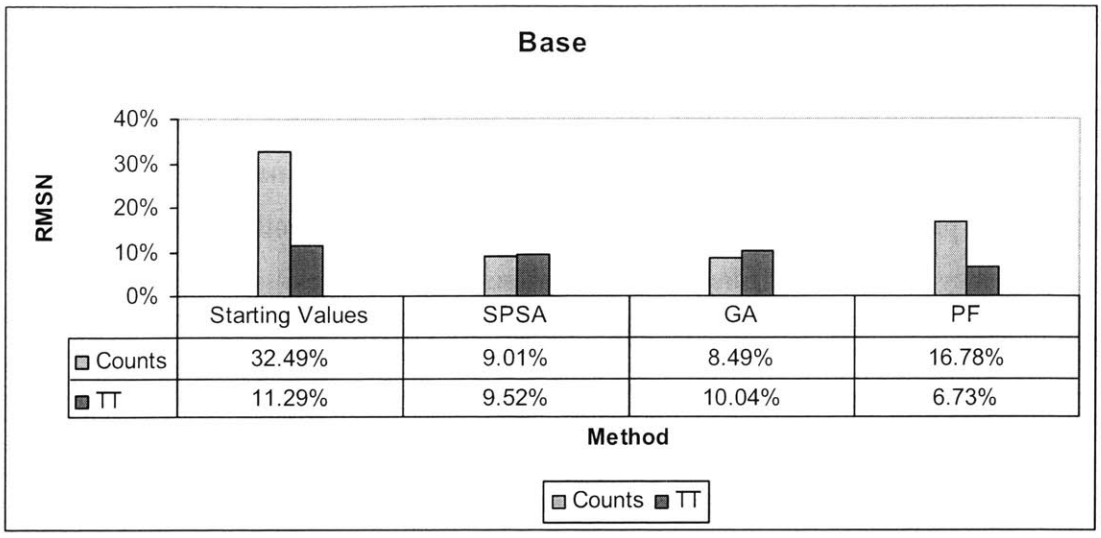

Fig 6.7: Comparison of algorithms: Demand-only calibration without AVI data

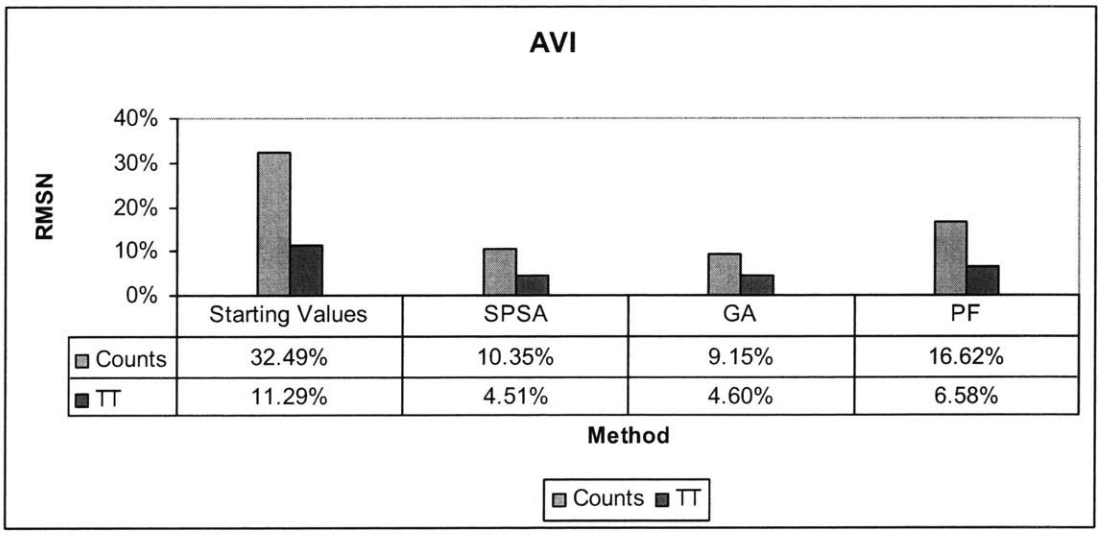

Fig 6.8: Comparison of algorithms: Demand-only calibration with AVI data

6.6.2 Simultaneous demand-supply calibration

For simultaneous calibration of both demand and supply parameters only SPSA and GA are used as candidate algorithms. The previous results from demand-only calibration suggest that particle filters perform considerably worse than the other two algorithms. Therefore they are eliminated from further analysis. Again the results of both algorithms are compared to the base case of calibration using only loop detector data.

- The results indicate that the calibration accuracy improves due to the usage of AVI data as compared to the base case (figures 6.10 and 6.11).

- Various error statistics are compared for the two algorithms with the starting values in table 6.2, for the case of simultaneous demand-supply calibration. GA and SPSA are both found to yield comparable calibration accuracy.

- Also it is noted from the comparison of fig 6.8 and 6.11 that the joint calibration of demand and supply is found to yield greater benefits than the demand-only calibration.

- Comparison between genetic algorithm and SPSA (figures 6.9, 6.10 and 6.11) shows that their performances are found to be comparable to each other both for the base case as well as for the final calibration using AVI data.

- Further, the 45° degree plots in figures 6.12 through 6.17 indicate that the positions of the observed and simulated entities are found to be a lot closer to the 45° line through origin after calibration than before calibration.

	Starting Values	SPSA Base	SPSA AVI	GA Base	GA AVI
RMSN Counts	32.49%	8.30%	6.87%	10.77%	11.05%
RMSN TT	11.29%	9.09%	5.68%	6.97%	4.05%
RMSPE Counts	48.68%	15.20%	14.15%	14.86%	18.84%
RMSPE TT	7.82%	6.18%	4.72%	5.03%	3.44%
RMSE Counts	144.32	36.87	30.50	47.82	49.08
RMSE TT	1.09	0.88	0.55	0.67	0.39
MEN Counts	25.97%	5.23%	2.66%	-0.86%	-4.13%
MEN TT	5.49%	-1.01%	0.89%	0.88%	-0.15%
MPE Counts	37.30%	9.08%	5.44%	1.17%	-6.92%
MPE TT	3.51%	-0.70%	1.14%	0.78%	-0.01%

Table 6.2: Accuracy of simultaneous demand-supply calibration

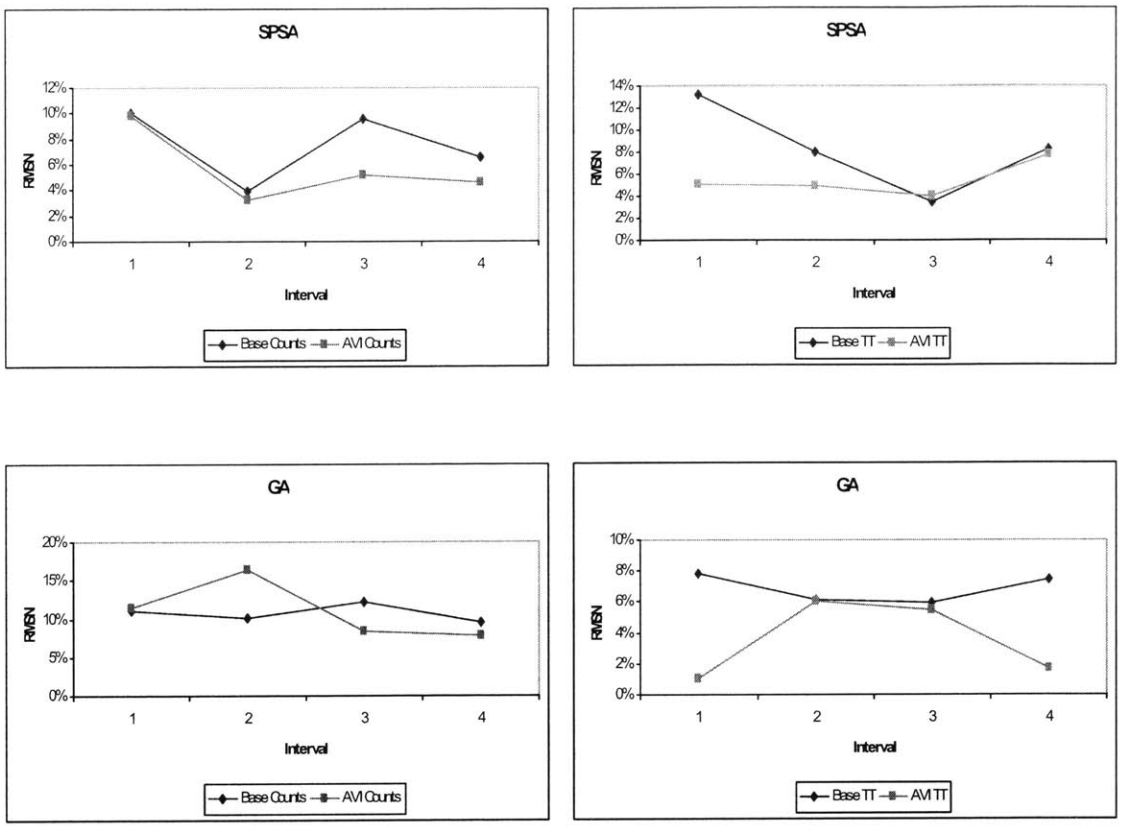

Fig 6.9: Calibration results: Simultaneous demand-supply calibration

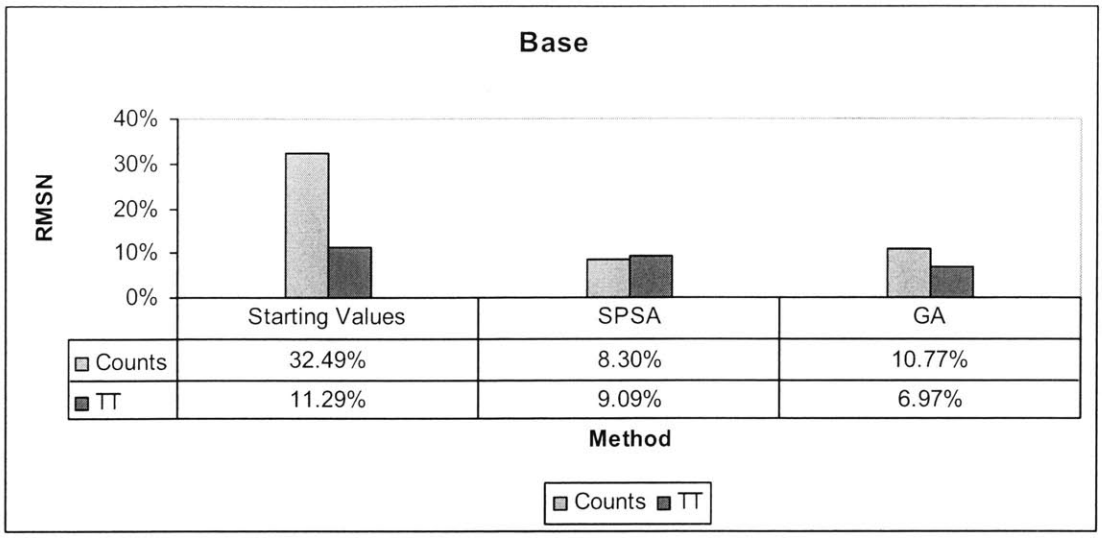

Fig 6.10: Comparison of algorithms: Demand-supply calibration without AVI data

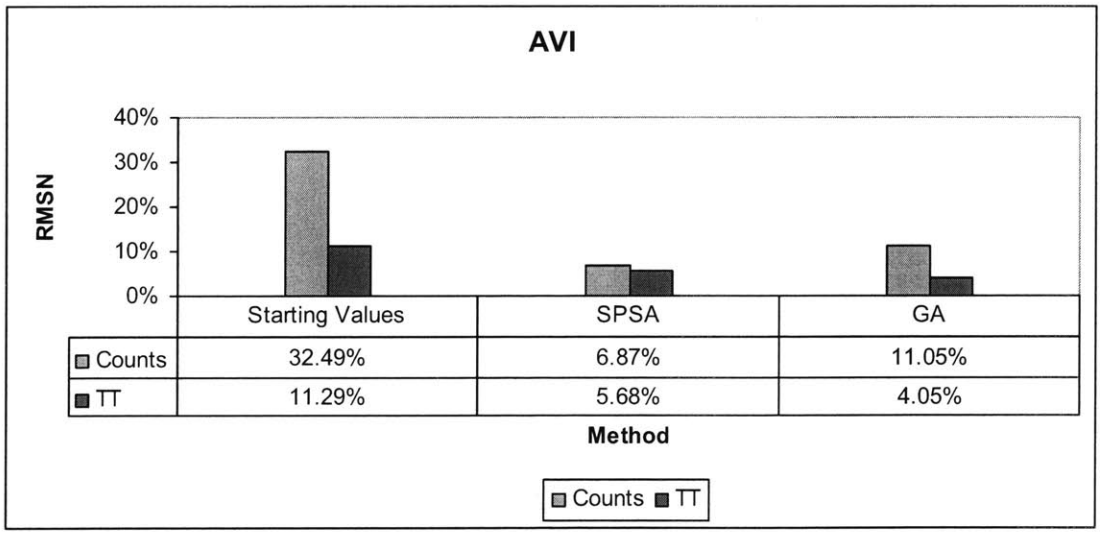

Fig 6.11: Comparison of algorithms: Demand-supply calibration with AVI data

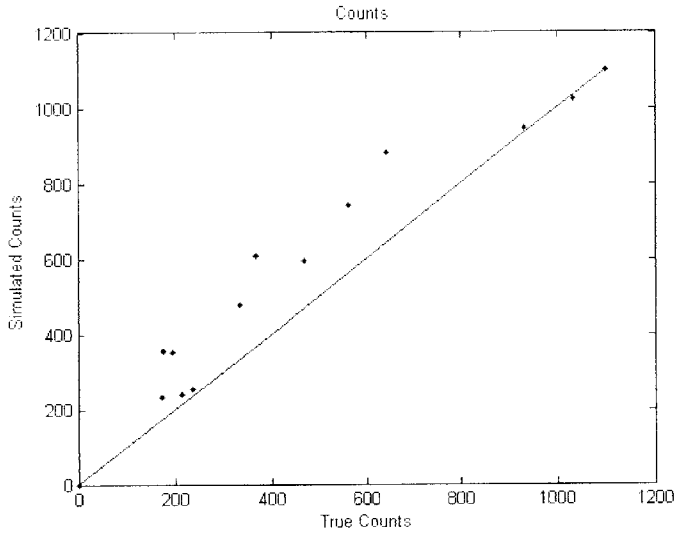

Fig 6.12: Starting values of sensor counts

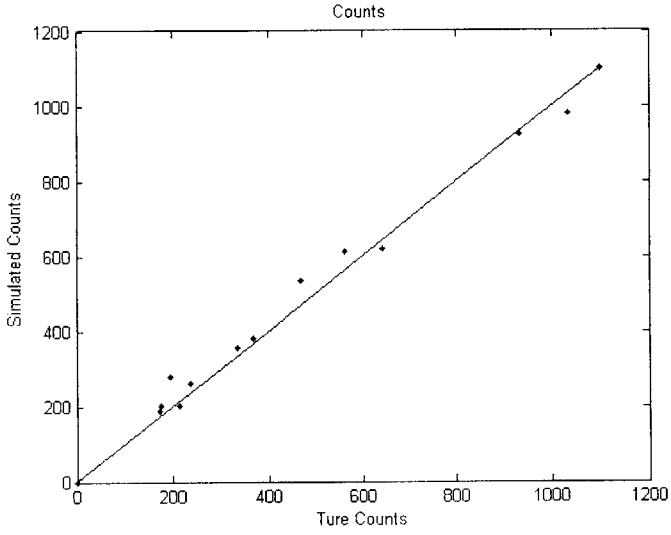

Fig 6.13: Sensor counts after demand-only calibration

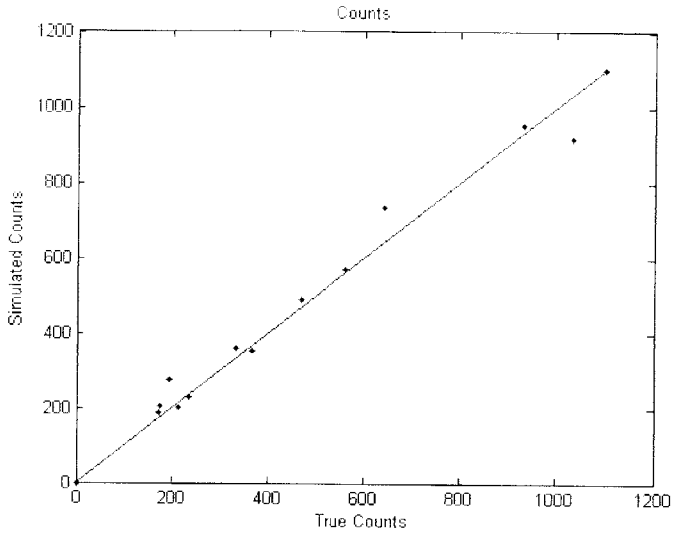

Fig 6.14: Sensor counts after demand-supply calibration

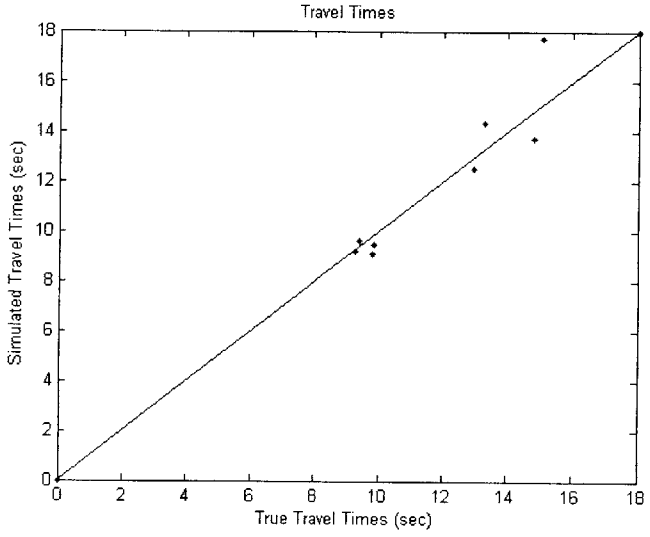

Fig 6.15: Starting values of travel times

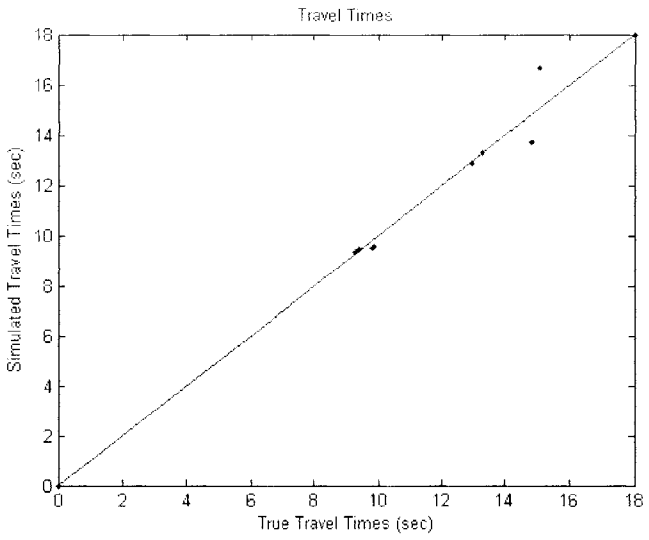

Fig 6.16: Travel times after demand-only calibration

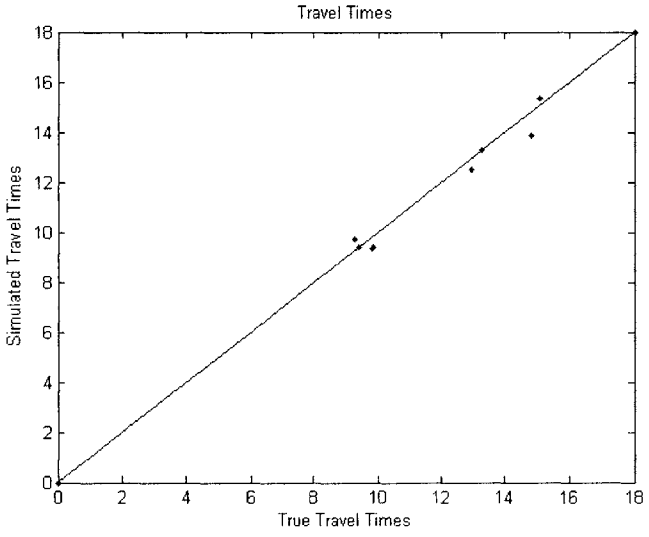

Fig 6.17: Travel times after demand-supply calibration

6.7 Sensitivity analysis

The calibration accuracy involves two measures, the fit between observed and simulated sensor counts, and the fit between the observed and simulated travel times. The final results depend on and are sensitive to how the two aspects of accuracy are relatively weighed in the objective function of the stochastic optimization problem. Therefore, a sensitivity analysis was performed to test the sensitivity of the calibration accuracy to the relative weights given to sensor counts deviations and travel time deviations in the objective function. The general expression for the objective function is as shown below. Note that we have ignored the third part of the objective function that includes the squared deviation of estimated OD flows from the historical values.

$$Minimize \sum_i (C_i^o - C_i^s)^2 + (w * \sum_i (TT_i^o - TT_i^s))^2$$

Where,

w = relative weight of travel time,

TT_i^s = ith simulated travel times measurement (in seconds),

TT_i^o = ith observed travel times measurement (in seconds),

C_i^s = ith simulated link counts measurement (in veh/interval),

C_i^o = ith observed link counts measurement (in veh/interval).

Sensitivity analysis was carried out using SPSA algorithm. Different weight values were considered for w, the relative weight for travel time in the objective function. It was

observed that as w increases, the travel time calibration accuracy also increases but the corresponding sensor counts calibration accuracy decreases slightly. However, the best linear fit line shows that the rate of decrease in sensor count accuracy is lower than the rate of improvement in travel time accuracy. Thus there is a tradeoff between the objective of minimization of sensor count accuracy and the minimization of travel time accuracy. The total RMSN value, i.e. the sum of the two RMSN values of sensor counts and travel times was used as the criteria to select the best tradeoff.

The total Normalized Root Mean Square (RMSN) error is found to be the minimum at w = 50. For weights w=20 and w=25, the total RMSN value was found to be comparable to that at w=50. From a traffic management viewpoint, the travel times are extremely useful and direct measure of link performance. Link counts are important to estimate the network state accurately. But the traffic routing decisions are based on reducing the travel times for drivers and the traffic information systems often provide travel time information. Therefore, accurate estimation of travel times is more critical than accurate estimation of sensor counts. Hence in making the choice between weights of 20, 25 and 50, 50 is chosen as the best tradeoff since out of these three, w=50 gives minimum error in travel time estimation.

Weight (w)	RMSN for Sensor Counts	RMSN for Travel Times	Total RMSN
0	9.01%	9.52%	18.53%
10	9.31%	7.41%	16.72%
20	8.11%	6.49%	14.60%
25	8.07%	6.69%	14.76%
30	12.09%	4.43%	16.52%
50	10.02%	4.51%	14.52%
60	12.47%	4.14%	16.61%

Table 6.3: Sensitivity analysis

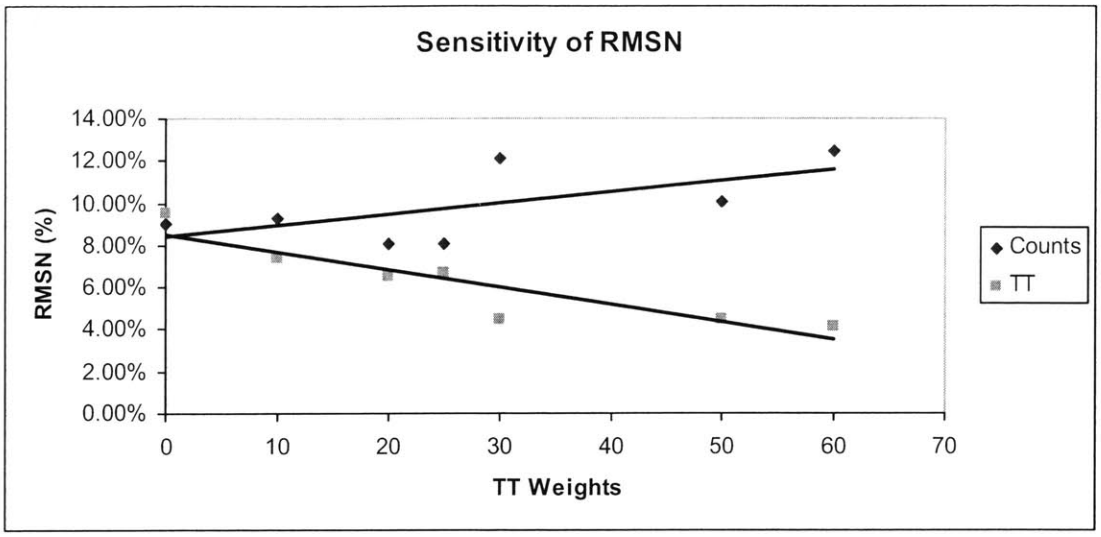

Fig 6.18: Sensitivity analysis

6.8 Validation

After the calibration exercise, the resulting calibrated parameter values are validated against a new set of data. This is essential to avoid over-fitting of the model parameters to a specific data set. For calibration, the sensor measurements were obtained as the average of 10 runs of MITSIMLab and the average values were used for performing the calibration. Now, in order to validate the calibrated DynaMIT, a single additional run of MITSIMLab is performed. The link count sensor data is perturbed between -20% to +20% of the true measurement in order to reflect the low level of accuracy of link counts. This perturbed sensor output from MITSIM is provided as input to the real-time estimation procedure within DynaMIT. The DynaMIT parameters and OD flows that were calibrated using the AVI data are used for this purpose. The resulting simulated sensor measurements are compared and the error statistics are calculated. Table 6.4 lists these error statistics. Validation results show improvements in terms of travel times over the calibration results, confirming the ability of DynaMIT to exploit available surveillance data to tweak the starting parameters so that the simulation output better reflects the prevailing traffic conditions. The validation error statistics values for the link counts are also found to be comparable to the calibration errors. The final RMSN values

for validation indicate that the RMSN error for link counts is 7.31% while that for travel times is 5.22%. These values are comparable and close to the calibration RMSN. Therefore, this validation ensures that the calibration has not led to over-fitting of parameters to a particular data set.

	Starting Values	Calibration	Validation
RMSN Counts	32.49%	6.87%	7.31%
RMSN TT	11.29%	5.68%	5.22%
RMSPE Counts	48.68%	14.15%	15.87%
RMSPE TT	7.82%	4.72%	4.30%
RMSE Counts	144.32	30.50	32.54
RMSE TT	1.09	0.55	0.50
MEN Counts	25.97%	2.66%	2.38%
MEN TT	5.49%	0.89%	0.63%
MPE Counts	37.30%	5.44%	5.85%
MPE TT	3.51%	1.14%	0.91%

Table 6.4: Validation results for synthetic network

6.9 Summary

In this chapter, a small hypothetical study network was used to demonstrate the effectiveness of the calibration methodology and test various algorithms. The individual parameters had to be fine tuned through trial and error to find the set of parameters which are most suitable from the viewpoint of convergence and accuracy. AVI data was in general found to improve the calibration accuracy in case of all the algorithms as compared to the base case. The simultaneous calibration of demand and supply was found to improve the accuracy of calibration as compared to the demand only calibration. Validation results are found to be comparable to calibration results ensuring that there is no data over-fitting. The genetic algorithm and simultaneous perturbation stochastic

approximation algorithm are found to be much more effective than the particle filter algorithm. SPSA resulted in slightly better calibration accuracy at a significantly lower computational effort than the genetic algorithm. Therefore, SPSA is chosen as the algorithm that will be used to perform the next case study with the large scale traffic network.

7. Case Study: Lower Westchester County

The previous chapter demonstrated the usefulness of the proposed calibration approach for calibrating the demand as well as supply parameters of a DTA system. It also established the importance of having AVI data in addition to the link counts data for improved calibration accuracy. SPSA algorithm was found to be the most effective in terms of calibration accuracy and computational effort. This chapter demonstrates the effectiveness of the calibration methodology for the case of a real network with highly complicated traffic flow patterns involving multiple vehicle classes and link use restrictions based on these classes. The number of parameters to be calibrated is much higher than the previous small network. The Lower Westchester County network in New York State USA has been used for calibration purpose. MITSIMLab microscopic traffic simulator is used for simulating the *truth*. This MITSIMLab network has been already calibrated based on real traffic data collected from field. However, the AVI data has been simulated in MITSIMLab since the AVI data from field could not be made available for this study.

7.1 Objectives

The main objectives of this case study are as follows:

- Test the scalability of the SPSA algorithm on a complex network with multiple vehicle classes and different link use restrictions for different classes.

- Illustrate the effectiveness of simultaneous calibration of demand and supply parameters as compared to the demand-only calibration.

- Evaluate the importance of AVI data for the calibration of DTA models for a real network.

7.2 Experimental design

The case study was performed for the real traffic network from the Lower Westchester County (LWC) in New York State in the US. The modified version of DynaMIT was used for calibration. This section explains the experimental details such as the study network structure, vehicle classes and entry restrictions, calibration parameters and measurement sensors.

7.2.1 Study network

The study network is from the Lower Westchester County (LWC) in the New York State in the US. The network is situated just to the north of New York City. It includes the most important highway corridors that connect up-state New York and Connecticut regions to New York City. A majority of commuters, who live in this region and commute daily to New York City, travel southwards in the morning and northwards in the evening along these corridors. The main north-south corridors include the New England Thruway (I-95) corridor on the eastern end and the New York State Thruway (I-87) corridor towards the western part of this network as well as the Hutchinson River Parkway between these two. The Cross Westchester Expressway (I-287) is the main east-west corridor in the region. The detailed map of the highway network in this region is presented in the fig 7.1.

The study network consists of 1767 links and 825 nodes. All the links are directed links and each link is further sub-divided into possibly multiple segments. The division of links into segments is based on the cross sectional characteristics such as geometry as well as traffic control characteristics such as speed limit of the road stretch. A link may contain only one segment if the cross sectional and traffic control characteristics remain constant across the entire length of the link. There are 482 origin-destination pairs and all of them

Fig 7.1: Network description

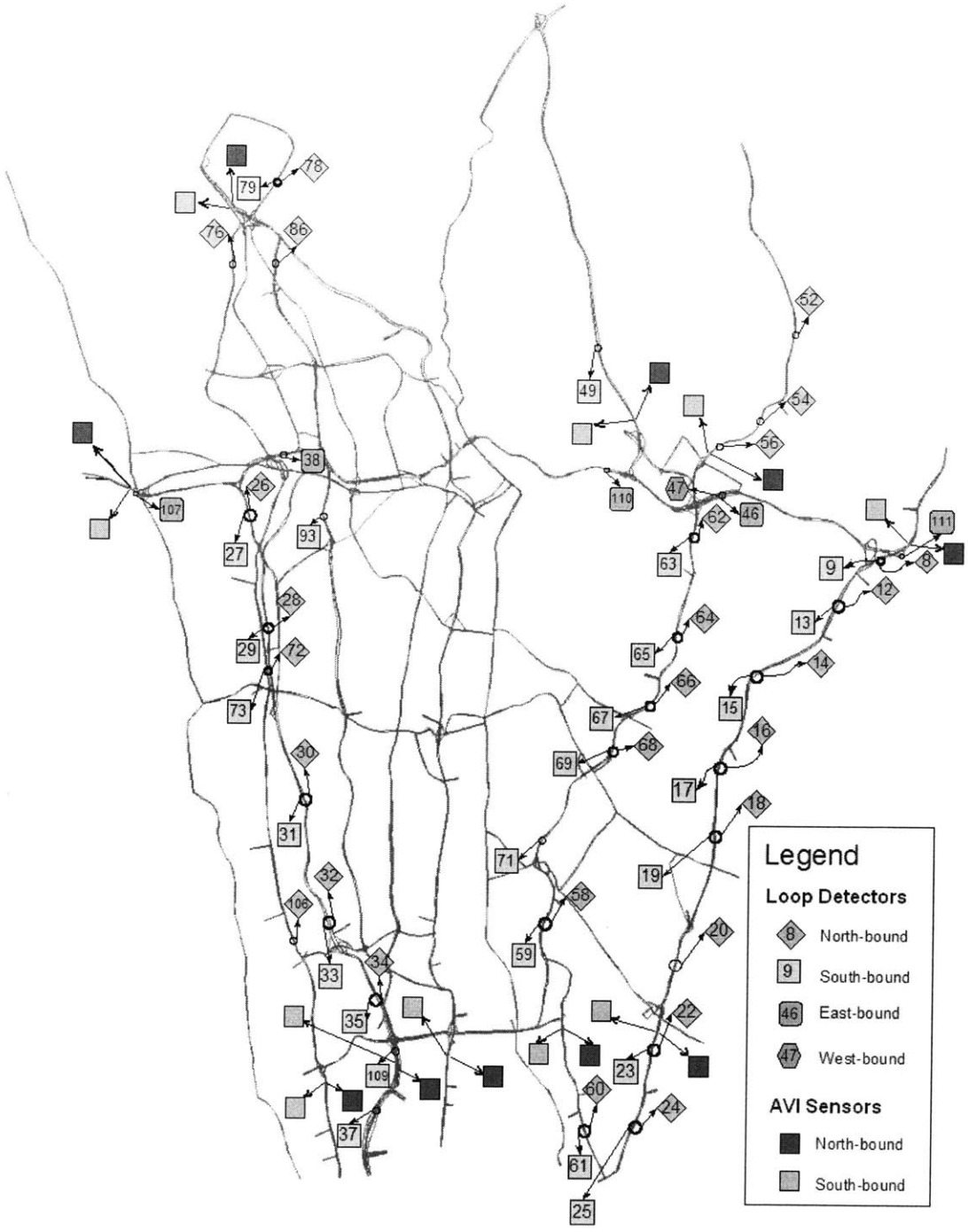

Fig 7.2: Sensor locations

have more than one possible routes. The simulation period is chosen to be the morning peak hour from 6:00 am to 8:00 am. This period has been divided into eight intervals of fifteen minutes each. Aggregated sensor data is available for these 15 minute intervals. Also the origin-destination flows are estimated for these same 15 minute intervals. Figure 7.2 provides detailed description of the network.

7.2.2 Vehicle mix

The study network contains freeways, parkways as well as some arterials. There are entry restrictions on the parkways in this region. No heavy commercial vehicles such as trucks or trailers are allowed to enter the parkways. So the parkways can have only passenger cars traveling on them. In order to model the traffic belonging to multiple vehicle classes the percentage of commercial vehicles within each OD flow must be known. However it is difficult to collect the proportions of commercial vehicles between every OD pair for every time interval. Instead an approximation is used. The network contains three toll plazas. These are the only places in the network where classified vehicle counts are available for every time interval. The proportion of commercial vehicles in all the OD flows for each time interval is approximated based on the average proportion of commercial vehicle observed during that time interval at each of these three toll plazas.

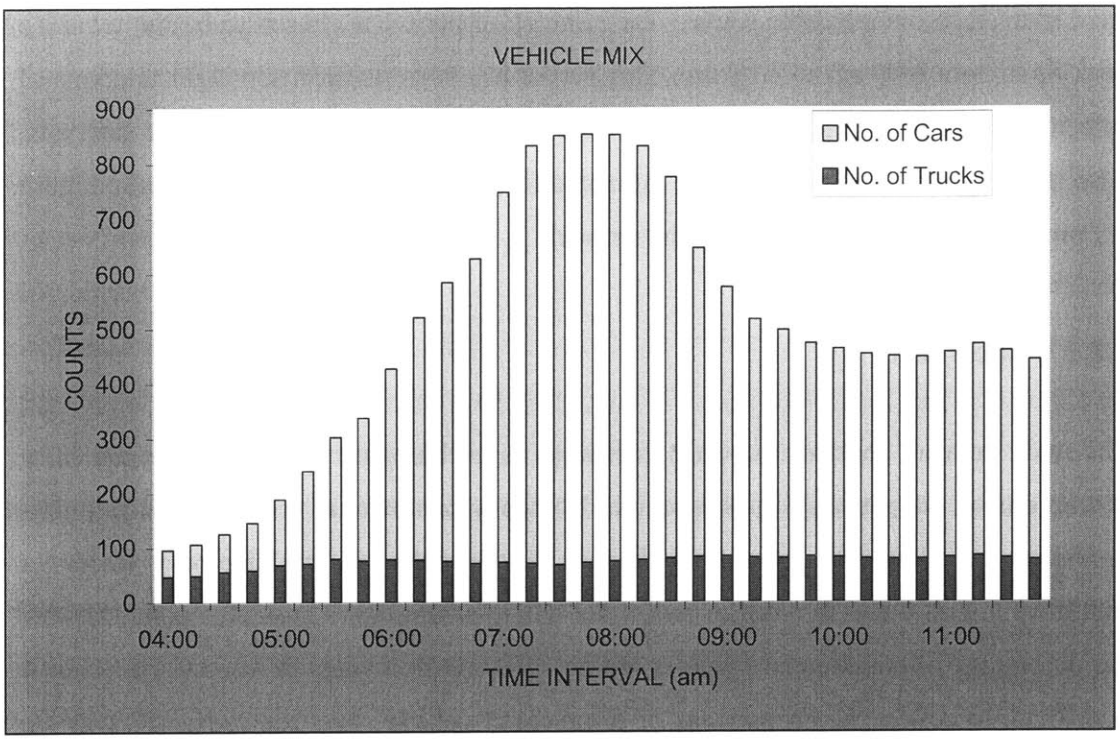

Fig 7.3: Proportion of commercial vehicles during the morning period

Fig 7.3 shows the classified vehicle counts that illustrate the mix of vehicles in the network throughout the morning period. It can be seen that the number of passenger cars varies based on the peak and off-peak hours while the commercial traffic stays more or less constant. Therefore, the proportion of commercial vehicles in the total traffic varies considerably with time.

7.2.3 Calibration parameters

There are 482 OD pairs in the network and the number of estimation intervals is eight. Therefore, the number of parameters to be calibrated for demand-only calibration includes the 482 OD pairs for 8 intervals i.e. a total of 3856 parameters. On the other hand, the simultaneous calibration of demand and supply parameters includes the additional speed-density curve parameter as well as the segment capacities. In this complex network it is highly burdensome to calibrate each of the 7 supply parameters for

each of the 2564 segments. Therefore, the segments are divided into 10 groups each having the same speed density curve parameters. However, the capacity of each segment is likely to be different. Therefore, the number of supply parameters to be calibrated equals 6 for each group in addition to 1 (capacity) for each individual segment. Therefore there are totally 2564*1 + 6*10 = 2624 parameters. Therefore, the total number of parameters for the simultaneous demand-supply calibration equals 3856+2624 = 6480.

7.2.4 Sensors

The study network is highly complex. However, economic constraints restrict detailed installation of surveillance devices. The network contains 1767 links out of which only 58 are equipped with loop detectors whose measurements are aggregated into 15 minutes intervals. Hence they provide 58 sets of link flow counts for each interval. Over all the time intervals, they provide 464 sensor measurements. To explore the effectiveness of AVI information on calibration accuracy, there are AVI sensors are installed in the study network at 10 locations on two-way roads. Thus they cover the links in either direction. Thus effectively, the network contains 20 directed links that are under AVI surveillance. All these sensors are indicated in the map shown in the fig 7.2. The sensor locations are chosen so that 5 of them are near the north end of the network while the other 5 are near the south end of the network. Zhou and Mahmassani (2005) have shown that the installation of AVI sensor so as to cover all the major ODs can be highly beneficial to improve the calibration accuracy. Therefore, the AVI sensor locations are chosen so as to maximize the coverage of major ODs in the network. Most of the traffic during the morning peak hours flows in the north-south directions. Therefore, these 20 AVI sensors provide 5*5 = 25 sets of travel time measurements for northbound traffic and another 25 measurements for the southbound traffic per time interval. Thus the total number of AVI travel time measurements equals 50*8 = 400. However, on certain occasions some additional measurements may be available due to vehicles that traverse two southern sensors or two northern sensors etc. Note that the AVI readings for all those vehicles, which are recorded at only one AVI sensor, are discarded, since they are not useful for travel time measurements.

MITSIMLab, which has been calibrated against field data, is used as a proxy for reality to generate the sensor counts and AVI readings for calibration of DYnaMIT. In order to minimize the random variations on sensor measurements due to stochastic nature of MITSIMLab simulator, 10 runs of MITSIMLab are performed and the average values of sensor measurements are used. Again, because the counts data collected from the loop detectors is usually noisy and inaccurate, the true sensor counts collected from MITSIMLab are modified by a symmetrical randomly distributed additive noise ranging from -20% to +20% of the true values. These perturbed sensor values are then used for calibration. Because the AVI sensors use technology far more accurate and reliable than the loop detectors, the true measurements are directly used for calibration purpose.

7.2.5 Starting parameter values

Some of the earlier studies to calibrate this network involved the estimation of demand and supply parameters for this network. These studies did not involve any usage of AVI data. The best available estimates of the parameters from the previous studies are perturbed randomly between -40% to +40% of the previous value and then used as the starting values for the demand and supply parameters for each of the algorithm run.

7.3 Implementation details

Similar to the previous case study, the calibration is carried out in two experiments. In the first experiment, only the demand parameters are calibrated while the supply parameters are held constant. The performance is compared with the base case where only the loop detector data is used for calibration. In the demand-only calibration, the supply parameter values obtained as a result of previous calibration studies were used. In the second experiment, both demand and supply are calibrated simultaneously.

SPSA algorithm, which performed most effectively in case of the small network case study, is used to calibrate this large network. It must be noted that the proportion of

commercial vehicles was not included as part of the decision variables. The commercial vehicle percentages obtained from the classification counts at the toll plazas for each time interval were taken as the estimates of true values.

Again substantial effort was necessary to fine tune the individual algorithm parameters for the large study network calibration. However, due to the experience from previous synthetic case study, reasonable starting values for most algorithm parameters were available. Those were further adjusted based on the empirical results to improve the algorithm convergence. For the SPSA algorithm, alpha = 0.602, gamma = 0.101, c = 1.9 and a = 15 were used for final calibration. Also a scaled ±1 Bernoulli distribution was used for the components of the perturbation vector. The reader is referred to section 6.5.1 for implementation details of the SPSA algorithm.

7.4 Habitual link travel times

The same procedure was used for updating the habitual travel time table using the simulated travel times.(ref: equation 6.7) However, it was found that the habitual travel times play a far more critical role in determination of the objective function value in the case of large network than the small network. Therefore, larger number of travel time smoothing iterations had to be performed to ensure that the objective function values are close to the equilibrium value. The most suitable value of the smoothing parameter λ was found to be 0.3 in this case. Five iterations of the travel time smoothing procedure are performed in the beginning while, after every 5 iterations of calibration algorithm 5 additional iterations of the smoothing process are performed.

7.5 Calibration results

Calibration involved two separate experiments. In first experiment, demand is calibrated using the fixed supply parameters while in the second experiment demand and supply are calibrated simultaneously using only the SPSA algorithm. Various error statistics

described by the equations 6.1 through 6.5 in the previous chapter are used as a measure of accuracy of calibration.

7.5.1 Demand-only calibration

The results of the demand-only calibration are summarized in the figures 7.5 and 7.6. In each of the cases the results were compared with the base case in which the calibration is performed with only the sensor counts being used as measurements. Root Mean Square Normalized (RMSN) is used as the measure of accuracy. Other error statistics for the starting parameter values, demand-only calibration and simultaneous demand-supply calibration with and without AVI information are listed in table 7.1.

Figures 7.4, 7.5 and 7.6 indicate that AVI data increases the calibration accuracy, as expressed in RMSN, for all the cases. The results show that the inclusion of AVI data into calibration improves accuracy not only in terms of lower travel time error but also lowers the sensor count error. In the previous case study, it was observed that the travel time accuracy improved due to usage of AVI data. But the corresponding sensor counts accuracy diminished slightly. However, in this case study the results indicate that the addition of AVI data helps the calibration algorithms to move in the right direction on the optimization surface more efficiently.

	Starting Values	Demand-Only Calibration Base Case	Demand-Only Calibration AVI Case	Demand-Supply Calibration Base Case	Demand-Supply Calibration AVI Case
RMSN Counts	25.19%	20.09%	18.24%	20.69%	17.94%
RMSN TT	29.04%	22.23%	21.24%	19.46%	18.92%
RMSPE Counts	28.54%	26.84%	30.32%	26.78%	28.92%
RMSPE TT	30.46%	22.43%	22.29%	23.89%	20.98%
RMSE Counts	151.68	120.94	109.83	124.56	108.04
RMSE TT	219.17	167.79	160.30	151.07	147.10
MEN Counts	-12.15%	-9.05%	-6.61%	-8.50%	-3.87%
MEN TT	3.28%	-7.59%	-5.54%	-9.00%	-6.06%
MPE Counts	-7.43%	-3.06%	-0.37%	-3.90%	0.16%
MPE TT	5.82%	-2.74%	-1.23%	-3.49%	-4.84%

Table 7.1: Accuracy of calibration for LWC network

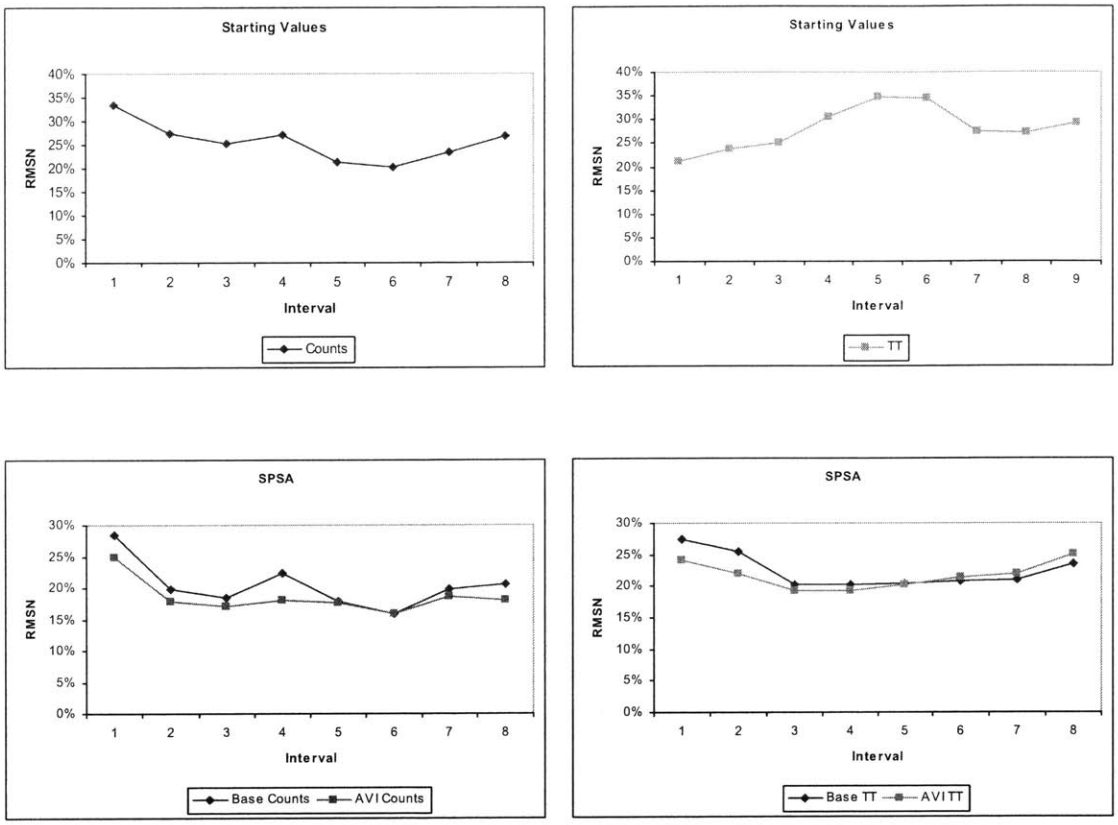

Fig 7.4: Calibration results: Demand-only calibration of LWC network

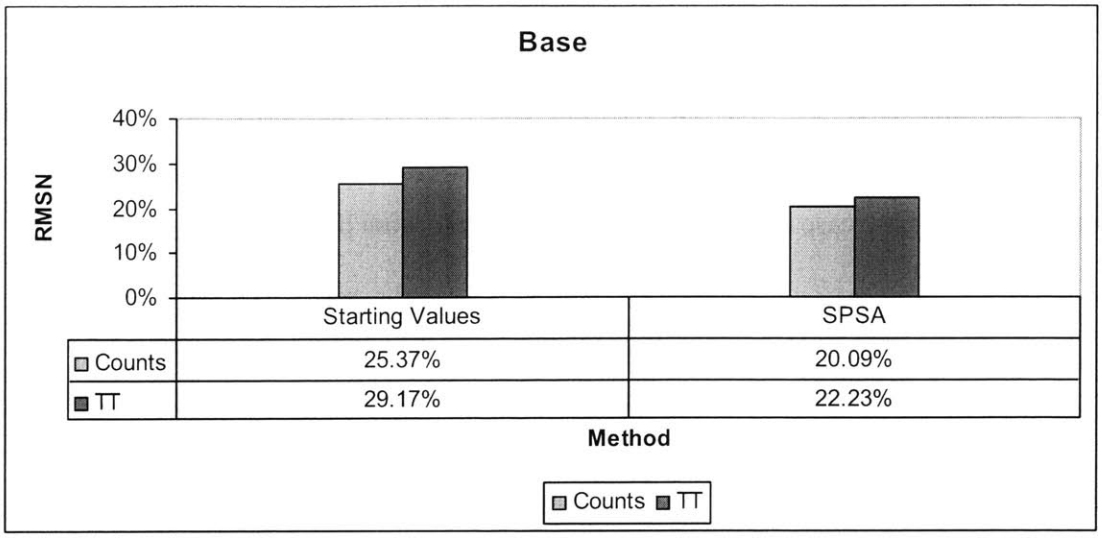

Fig 7.5: Demand-only calibration without AVI data

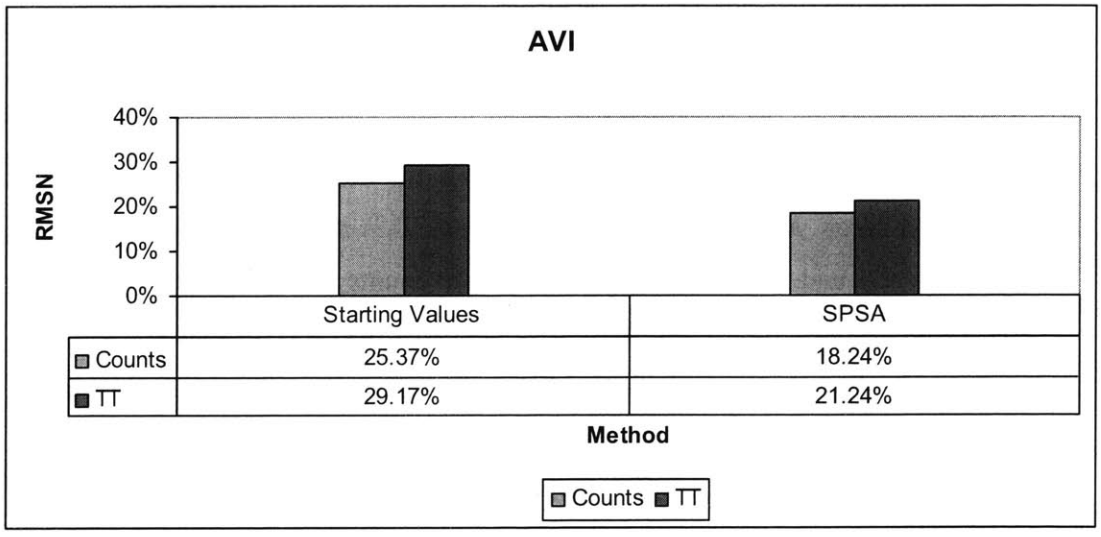

Fig 7.6: Demand-only calibration with AVI data

7.5.2 Simultaneous demand-supply calibration

SPSA algorithm was also selected for the simultaneous demand-supply calibration because of its superior performance in the previous case study. Again the results are compared to the base case of calibration using only link count data.

- From the final RMSN values it is observed that the calibration accuracy improved due to the addition of AVI data in the calibration process as it did in the previous experiment.

- Also it is found (figures 7.6 and 7.9) that the simultaneous calibration of demand and supply parameters has considerably better accuracy as compared to the demand-only calibration. Figures 7.10 to 7.15 show the 45^{o} plots which compare the simulated and the observed values of sensor counts and travel times in all the cases, with the line through origin having slope equal to 1. The successive calibration improvements can be observed especially significantly for the travel times.

- Table 7.1 describes various error statistics for the starting values and the four calibration cases. Apart from the RMSN values, it is interesting to note that the starting MEN and MPE values for the sensor counts indicate that there was a significant negative bias in the starting values. This bias was decreased due to the process of calibration and the final calibrated parameters under the simultaneous demand-supply calibration case using AVI data show that the MEN values reduced from -12.15% to -3.87% and the MPE values reduced from -7.43% to +0.16%.

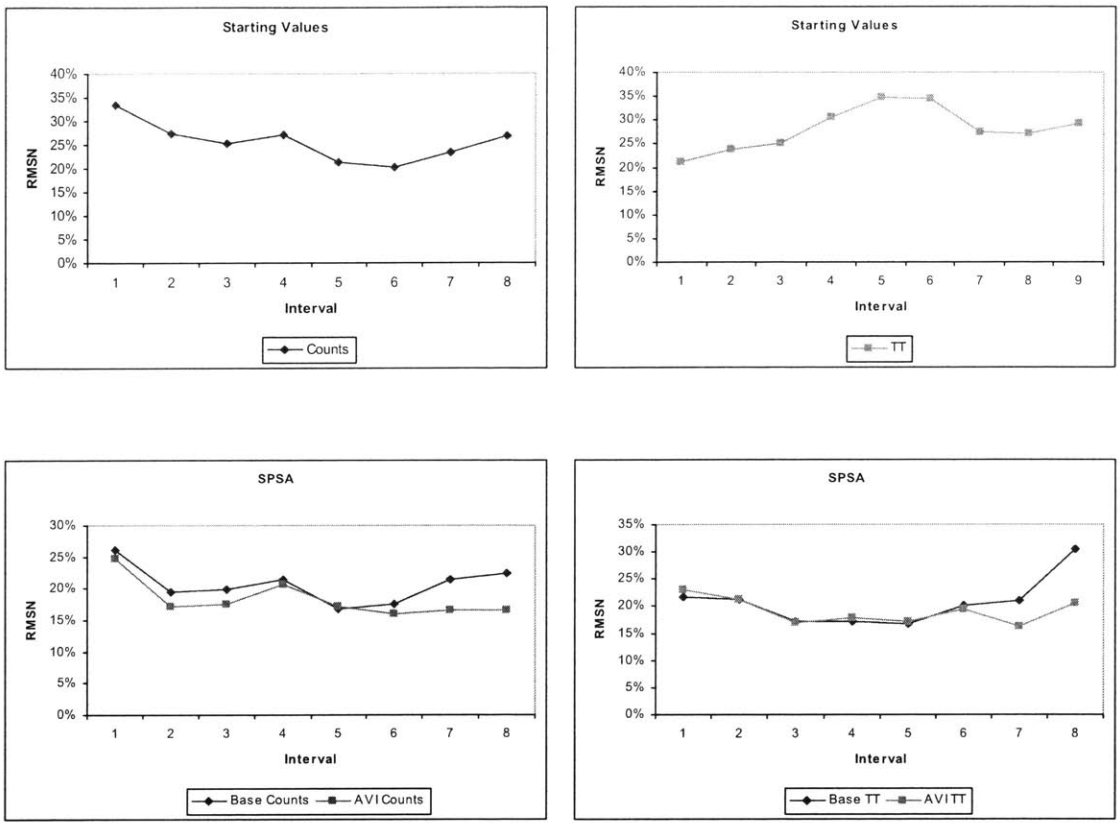

Fig 7.7: Calibration results: Simultaneous demand-supply calibration of LWC network

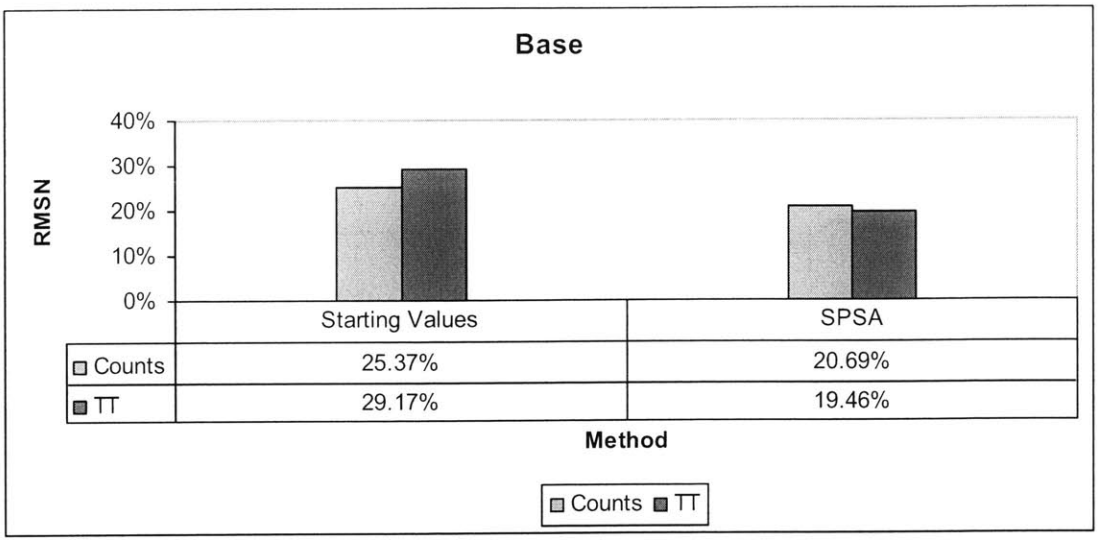

Fig 7.8: Final demand-supply calibration without AVI data

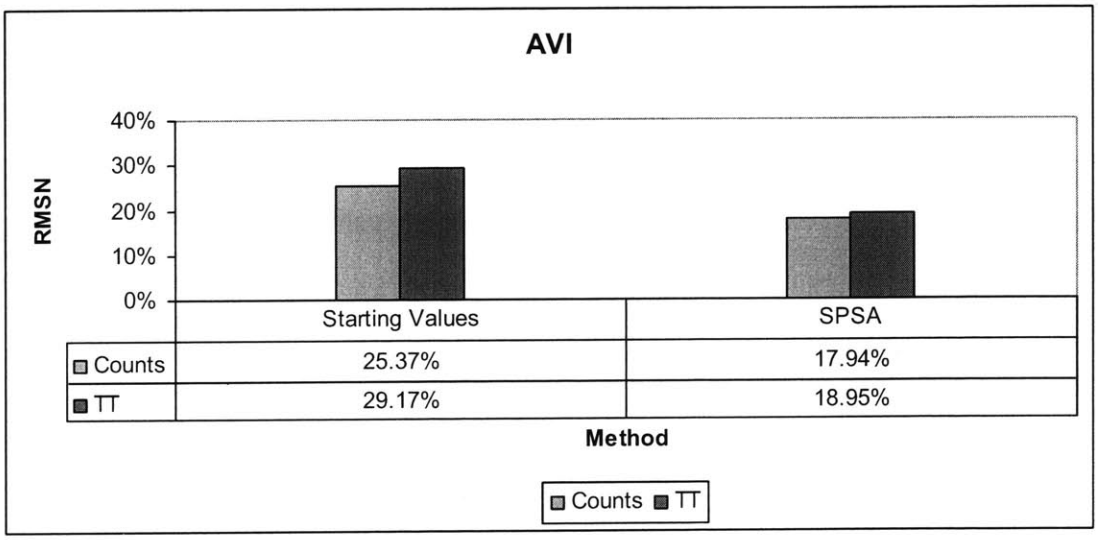

Fig 7.9: Final demand-supply calibration with AVI data

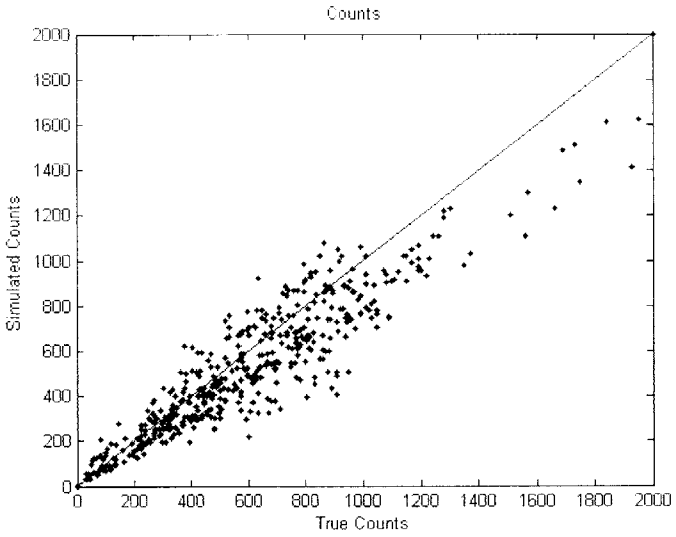

Fig 7.10: Starting value of sensor counts

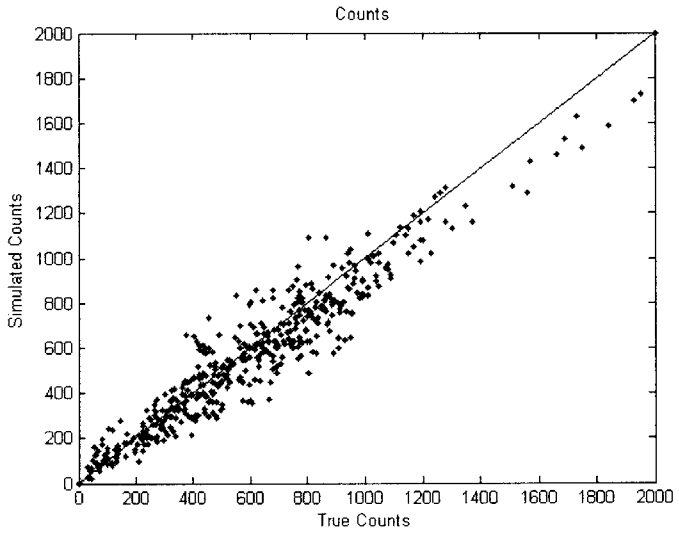

Fig 7.11: Sensor counts after demand-only calibration

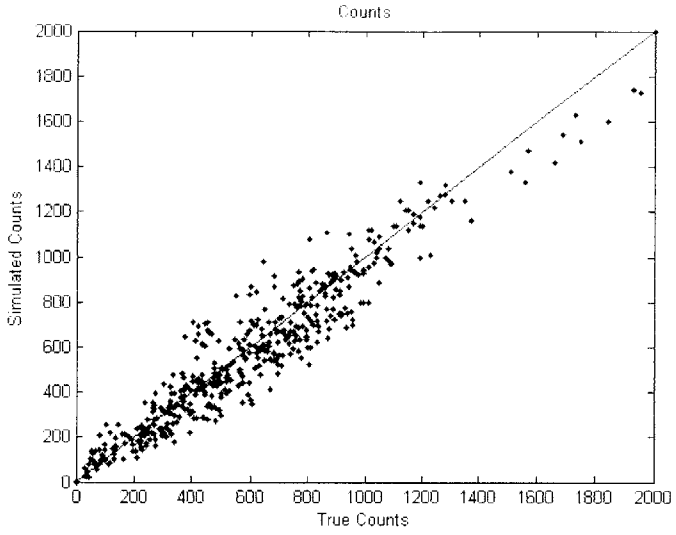

Fig 7.12: Sensor counts after demand-supply calibration

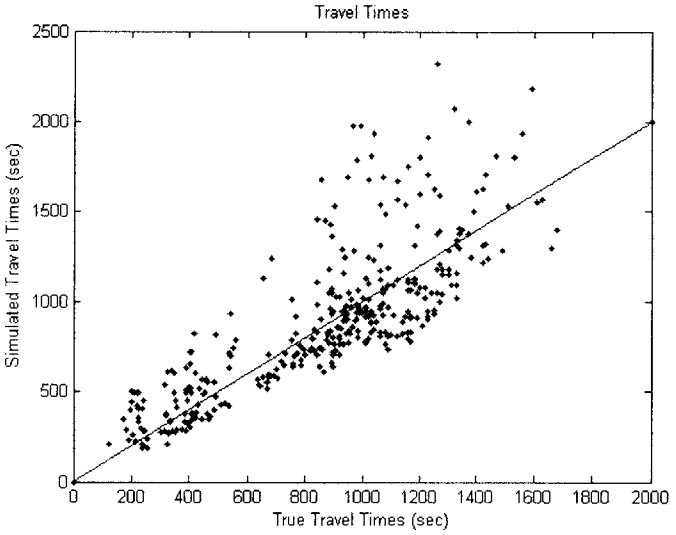

Fig 7.13: Starting values of travel times

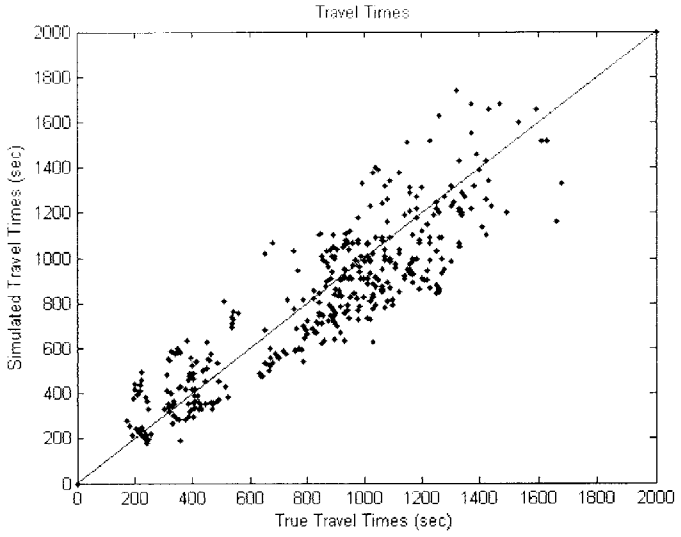

Fig 7.14: Travel times after demand-only calibration

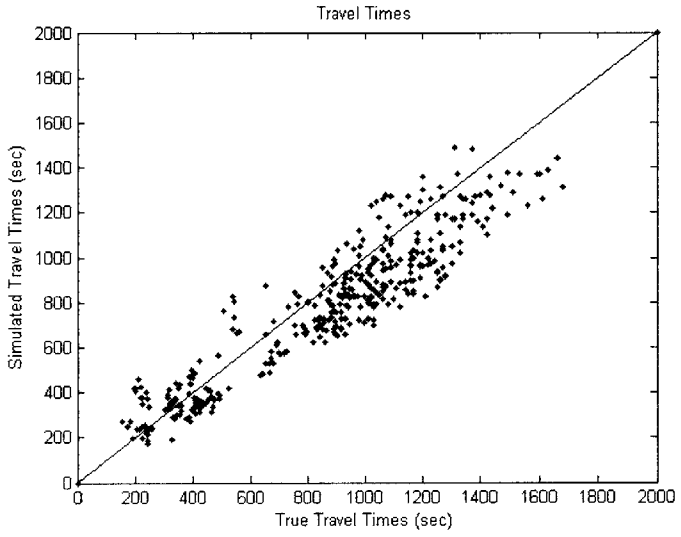

Fig 7.15: Travel times after demand-supply calibration

7.6 Validation

Similar to the previous case study, another set of MITSIMLab output was generated in order to validate the calibrated parameters. A single run of MITSIMLab was performed to generate a new day's sensor data. The sensor data was perturbed between -20% to +20% of the true values. The real time estimation procedure within DynaMIT was used and sensor output obtained from DynaMIT was compared to the perturbed sensor measurements given by MITSIMLab. Table 7.2 lists various error statistics for the final calibration and validation results. The statistics for starting values are provided for the purpose of comparison.

	Starting Values	Calibration	Validation
RMSN Counts	25.19%	17.94%	17.63%
RMSN TT	29.04%	18.92%	18.55%
RMSPE Counts	28.54%	28.92%	27.60%
RMSPE TT	30.46%	20.98%	21.06%
RMSE Counts	151.68	108.04	106.27
RMSE TT	219.17	147.10	148.67
MEN Counts	-12.15%	-3.87%	-3.98%
MEN TT	3.28%	-10.06%	-9.88%
MPE Counts	-7.43%	0.16%	-0.39%
MPE TT	5.82%	-4.84%	-5.15%

Table 7.2: Validation results for LWC network

The results indicate that the validation RMSN for sensor counts is 17.63% and for travel times it is 18.55%, both of which are both slightly lower than the corresponding values for calibration. This gives indications that over-fitting issue has been avoided in this calibration effort.

7.7 Summary

In this chapter, a real traffic network was used as a case study to demonstrate the feasibility of the proposed approach for real networks and to illustrate its scalability for extremely large set of calibration parameters. The individual parameters were fine-tuned through trial and error to find the most suitable combination of algorithm parameters. The simultaneous perturbation stochastic approximation algorithm is found to perform effectively even for the large size problem. AVI data was found to improve the calibration accuracy as compared to the base case in both demand-only calibration as well as simultaneous demand-supply calibration. The simultaneous calibration of demand and supply improved the accuracy of calibration as compared to the demand-only calibration. Validation of final calibrated parameter suggests that over-fitting is unlikely to be a major concern. Overall, the proposed approach was found to be scalable and could be extended to calibrate large scale real traffic networks with multiple vehicle classes.

8. Conclusion

This thesis focused on the calibration of Dynamic Traffic Assignments models using automatic vehicle identification data. A mesoscopic DTA system called DynaMIT was used to demonstrate the usefulness of the proposed methodology. Both the demand and supply parameters of the DTA system were calibrated with and without the additional travel time information available from AVI data. Three different calibration algorithms were evaluated in terms of comparative performance. The first section summarizes the major findings and research contribution of the thesis. The final section focuses on directions for future research.

8.1 Major findings and research contribution

Review of previous studies emphasized the following findings:

- Majority of the earlier calibration efforts have relied on the separate calibration of individual models within the DTA systems.

- Most previous efforts have focused on the loop detector data available at aggregate level.

- All the previous studies that have tried to incorporate disaggregate AVI data into rigorous calibration studies have only been restricted to OD estimation. Although some researchers have provided a generalized framework that can incorporate various traffic data types, there are no calibration results for real traffic networks using AVI data, to the best of author's knowledge.

- A variety of information about the state of traffic network can be extracted from AVI data. Although travel time information collected from AVI data can be extremely useful for calibration of variety of DTA components models, previous

studies have ignored travel time information in favor of other information types that are only appropriate for calibration of specific models within DTA.

In this study, DTA calibration problem was formulated as a state estimation problem and as a simulation based optimization problem. Calibration of DTA models was performed using three different algorithms, viz. Simultaneous Perturbation Stochastic Approximation (SPSA), Genetic Algorithm (GA) and Particle Filter (PF). In each of the cases, the calibration performance was compared with the case where only loop detector data is available. The feasibility and effectiveness of the proposed approaches was tested using a small synthetic network. Afterwards, a real traffic network with multiple vehicle classes was used to illustrate the scalability of selected approach. For each network, two experiments were performed. In the first experiment, only the demand parameters were calibrated while holding supply parameters constant at the apriori value. In the second experiment, the demand and supply parameters were simultaneously calibrated. The following were the main findings and conclusions:

- Demand calibration was found to improve the calibration accuracy considerably as compared to the accuracy of the apriori estimates.

- Simultaneous demand-supply calibration was found to be superior compared to the demand-only calibration and it increased the calibration accuracy substantially.

- Comparison between calibration results using combined loop detector and AVI data with the calibration results using only loop detector data indicated that the AVI data is useful in improving the calibration accuracy in all the experiments.

- For the small scale network, the sensitivity analysis suggested that the relative weights given to AVI measurements is critical in determination of trade-off between the sensor counts accuracy and travel time accuracy. While AVI data

helps improve the travel time accuracy significantly, it tends to decrease the sensor count accuracy slightly.

- However, in case of the large network, the AVI data was found to improve the calibration accuracy both in terms of sensor count error as well as travel time error. A possible reason for this phenomenon is that the number of loop detectors is small in real large scale networks. Also the measurement accuracy and reliability of loop detectors is low. Hence the loop detector data by itself cannot provide all the necessary information to calibrate the enormous set of parameters. However, the addition of accurate AVI data aids the calibration process to move towards the true network state more efficiently, hence improving the overall calibration performance.

- For the small network, SPSA was found to be the most effective algorithm followed closely by GA. However, the GA's performance required additional computational effort due to greater number of function evaluations. Particle filter algorithm did not perform as well as the other two algorithms in this particular study. A likely reason could be considerably poor starting values of the calibration parameters.

- The empirical results from the validation tests are found to be consistent with the calibration results and thus validate the feasibility and accuracy of the methods used for calibration in this research.

- The SPSA algorithm was also found to be scalable. A large set of parameters (6480 parameters) could be calibrated using SPSA.

8.2 Directions for future research

This thesis focused on a generalized framework for calibration and explored the usefulness of incorporating AVI data with a special focus on travel times. Two different frameworks were used for formulating the problem and three algorithms were considered for calibration. Some of the directions for future research are as follows:

- Travel times were considered in calibration efforts and were shown to improve the accuracy. It would be of interest to explore the effectiveness of a combination of travel times and other types of AVI information such as sub-path flows and split-fractions in addition to link flow counts in terms of further improvements in calibration performance.

- A fixed set of locations were used for the AVI sensors. In large scale networks, there exist multiple suitable locations for placement of these sensors. The relative usefulness of different sensor locations in terms of calibration accuracy could be evaluated. Multiple sensor placements may be compared to evaluate the optimum placement that provides the most useful calibration data.

- This study showed that the particle filter algorithm did not perform on par with the other two algorithms. However, some recent studies (such as Doucet et al., 2001) have proposed extensions of the basic particle filter algorithm that could potentially improve the accuracy of calibration using Monte-Carlo simulation based methods. This could be another natural extension of this work.

- While preliminary validation results indicate low error values, the calibrated parameters need to be validated through further real time applications and actual field data.

Appendix A: Overview of DynaMIT

DynaMIT (Ben-Akiva et al., 1997) is a state-of-the-art DTA system with both real-time and planning applications. This appendix provides an overview of DynaMIT-R, a DTA-based real-time mesoscopic traffic simulation model. The features and functionalities of the DynaMIT-R system are presented, along with an overview of its model components. The system's unknown quantities (both model inputs and parameters) are enumerated, both in order to illustrate the dimensionality of the problem, and provide an introduction to the case studies presented in this thesis. Most of the material in this appendix is derived from Balakrishna (2006).

Apart from its real-time applications, DTA has the potential to significantly improve the transportation planning process for networks with congested facilities. DynaMIT-P is a DTA-based planning tool developed at MIT that is designed to assist planners in making decisions regarding proposed investments and operational changes in local and regional transportation networks. DynaMIT-P adapts the modules contained in the real-time DynaMIT system for off-line planning applications.

The models in a DTA system can be broadly categorized into two classes. The demand simulator captures aggregate flows of vehicles between points on the network, and models individual drivers' route choice decisions at various stages of their trips. In addition, the demand models play a key role in the prediction of future network flows. The supply simulator models vehicle movements on the links of the network. The outputs of the supply simulator include link, path and sub-path travel times, link flows, speeds and densities, and queue lengths upstream of bottlenecks.

A.1 Introduction

DynaMIT (Dynamic Network Assignment for the Management of Information to Travelers) is a state-of-the-art traffic simulation system based on the principle of dynamic traffic assignment. Its real-time version (DynaMIT-R) is designed for traffic estimation

and prediction, and the generation of traveler information and consistent anticipatory route guidance. DynaMIT-R supports the operation of Advanced Traveler Information Systems (ATIS) and Advanced Traffic Management Systems (ATMS) at Traffic Management Centers (TMC). A planning version of DynaMIT, codenamed DynaMIT-P, employs DTA for short-term planning scenarios such as work zones, optimal VMS locations and OD estimation. Sponsored by the Federal Highway Administration (FHWA), DynaMIT was designed and developed at the Intelligent Transportation Systems Program at the Massachusetts Institute of Technology.

A.2 Features and functionality

The key to DynaMIT's functionality is its detailed network representation, coupled with models of traveler behavior. Through an effective integration of historical databases with real-time inputs from field installations (surveillance data and control logic of traffic signals, ramp meters and toll booths), DynaMIT is designed to efficiently achieve:

- Real time estimation of network conditions.

- Rolling horizon predictions of network conditions in response to alternative traffic control measures and information dissemination strategies.

- Generation of traffic information and route guidance to steer drivers towards optimal decisions.

To sustain users' acceptance and achieve reliable predictions and credible guidance, DynaMIT incorporates unbiasedness and consistency into its core operations. Unbiasedness guarantees that the information provided to travelers is based on the best available knowledge of current and anticipated network conditions. Consistency ensures that DynaMIT's predictions of expected network conditions match what drivers would experience on the network. DynaMIT has the ability to trade-off level of detail (or

resolution) and computational practicability, without compromising the integrity of its output.

A.3 Overall framework

DynaMIT is composed of several detailed models and algorithms to achieve two main functionalities:

- Estimation of current network state using both historical and real-time information.

- Generation of prediction-based information for a given time horizon.

The estimation and prediction phases operate over a rolling horizon. This concept is illustrated with a simple example in Fig A.1.

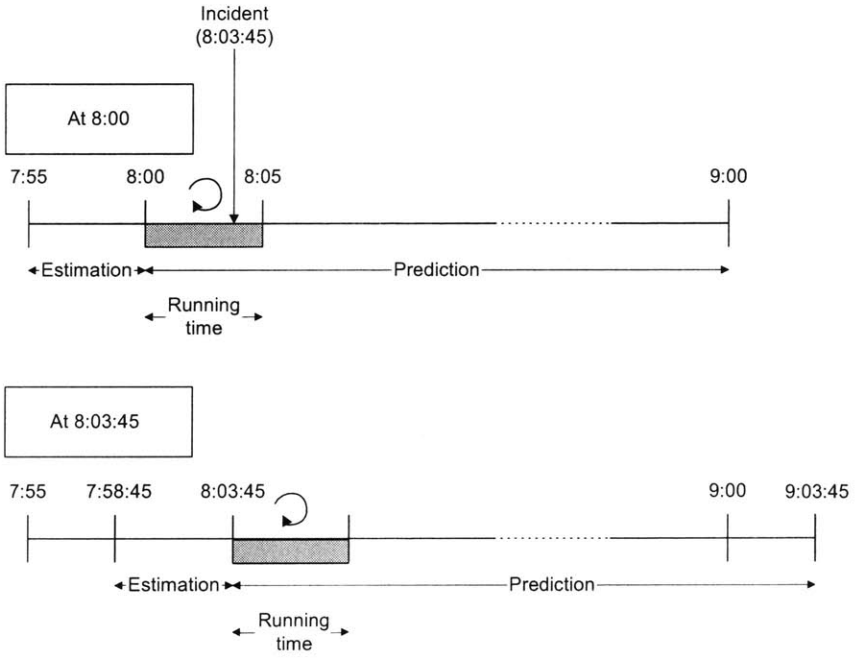

Fig A.1: The rolling horizon

It is now 8:00am. DynaMIT starts an execution cycle, and performs a state estimation using data collected during the last 5 minutes. When the state of the network at 8:00 is available, DynaMIT starts predicting for a given horizon, say one hour, and computes a guidance strategy which is consistent with that prediction. At 8:07, DynaMIT has finished the computation, and is ready to implement the guidance strategy on the real network. This strategy will be in effect until a new strategy is generated. Immediately following that, DynaMIT starts a new execution cycle. Now, the state estimation is performed for the last 7 minutes. Indeed, while DynaMIT was busy computing and implementing the new guidance strategy, the surveillance system continued to collect real-time information, and DynaMIT will update its knowledge of the current network conditions using that information. The new network estimate is used as a basis for a new prediction and guidance strategy. The process continues rolling in a similar fashion during the whole day.

The overall structure with interactions among the various elements of DynaMIT is illustrated in Fig A.2. DynaMIT utilizes both off-line and real-time information.

The most important off-line information, in addition to the detailed description of the network, is a database containing historical network conditions. This database might combine directly observed data and the results of o®-line models. The historical database contains time-dependent data, including origin-destination matrices, link travel times and other model parameters. Clearly, the richer the historical database, the better the results. Such a rich historical database requires substantial data collection and careful calibration.

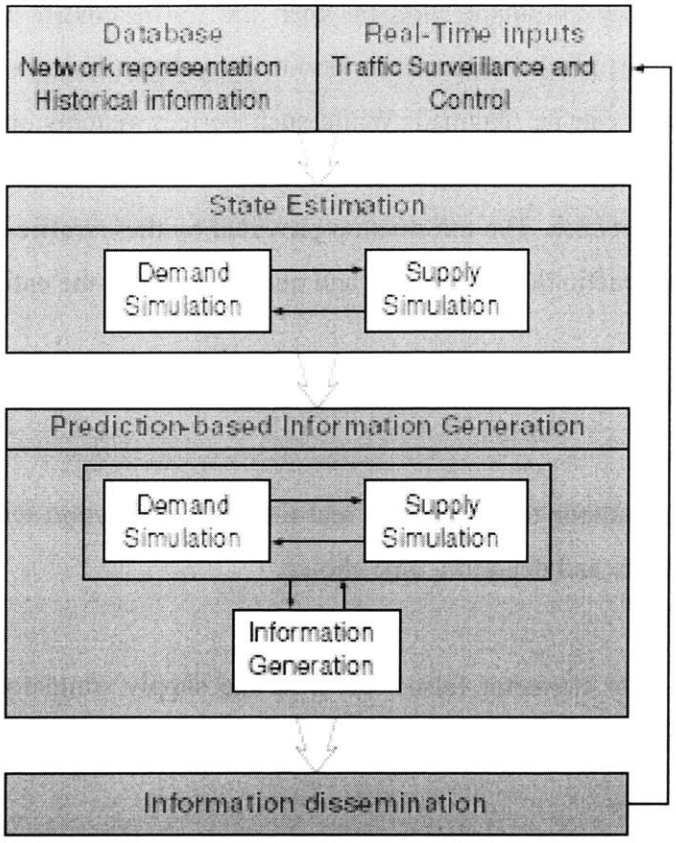

Fig A.2: The DynaMIT framework

Real-time information is provided by the surveillance system and the control system. DynaMIT is designed to operate with a wide range of surveillance and control systems. The minimum real-time information required by DynaMIT is time dependent link flows, incident characteristics (location, starting time, duration and severity), and traffic control strategies.

State Estimation

The state estimation module provides estimates of the current state of the network in terms of OD flows, link flows, queues, speeds and densities. This step represents an important function of DTA systems, since information obtained from the traffic sensors can vary depending on the type of surveillance system employed. In an ideal system

where there is two-way communication between the traffic control center and every vehicle in the network, perfect information about the vehicle location and possibly its origin and destination can be obtained. While such perfect systems are possible in the future, most existing surveillance systems are limited to vehicle detectors located at critical points in the network. The information provided by these traffic sensors therefore must be used to infer traffic flows, densities and queue lengths in the entire network.

The main models used by the State Estimation module are:

- A demand simulator that combines real-time OD estimation with user behavior models for route and departure time choice.

- A network state estimator (also known as the supply simulator) that simulates driver decisions and collects information about the resulting traffic conditions.

The demand and supply simulators interact with each other in order to provide the demand and the network state estimates that are congruent and utilize the most recent information available from the surveillance system (Fig A.3).

Demand Simulation

Demand estimation in DynaMIT is sensitive to the guidance generated and information provided to the users, and is accomplished through an explicit simulation of pre-trip departure time, mode and route choice decisions that ultimately produce the OD flows used by the OD estimation model. The pre-trip demand simulator updates

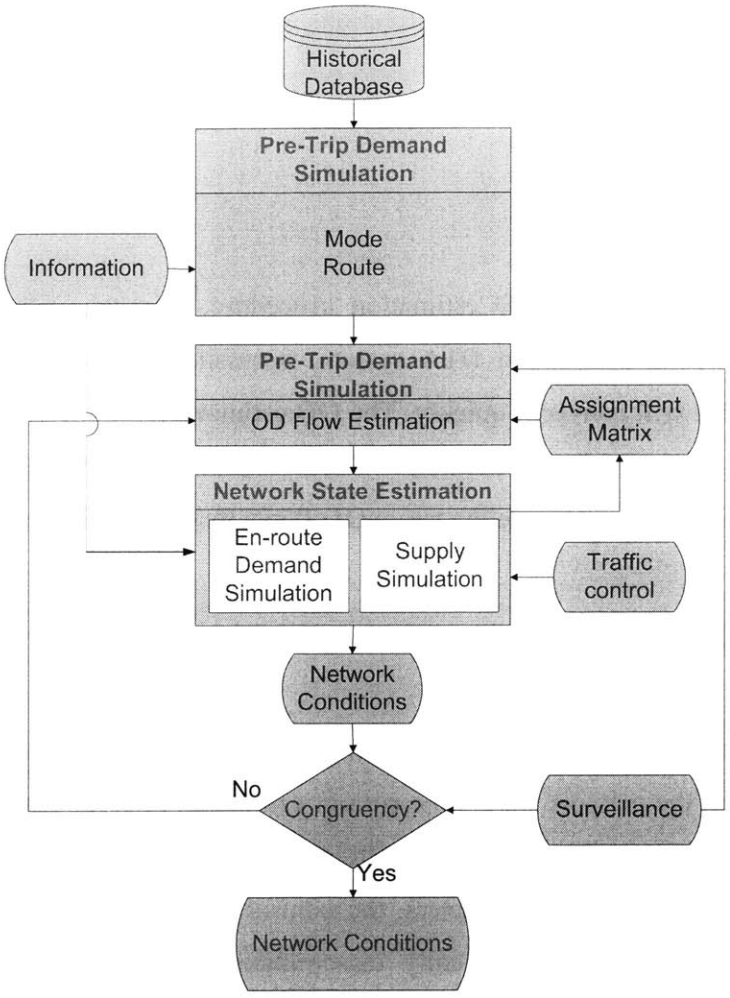

Fig A.3: State estimation in DynaMIT

the historical OD matrices by modeling the reaction of each individual to guidance information. The consequent changes are then aggregated to obtain updated historical OD matrices. However, these updated historical OD flows require further adjustments to reflect the actual travel demand in the network. Reasons for the divergence of actual OD flows from historical estimates include capacity changes on the network (such as the closure of roads or lanes), special events that temporarily attract a large number of trips to a destination, and other day-to-day fluctuations. Consequently, one of the requirements for dynamic traffic modeling is the capability to estimate (and predict) OD flows in real time. The OD model uses updated historical OD flows, real-time measurements of actual link flows on the network, and estimates of assignment fractions (the mapping from OD

flows to link flows based on route choice fractions and travel times) to estimate the OD flows for the current estimation interval.

OD Smoothing

The fixed point nature of the OD estimation procedure, coupled with the real-time requirements of a prediction-based DTA system, necessitates the use of an efficient solution scheme that will converge quickly. The OD estimation module within DynaMIT utilizes an algorithm similar to the Method of Successive Averages with Decreasing Re-initializations (MSADR) to compute the target OD flows for successive iterations.

Supply Simulation

The network state estimator utilizes a traffic simulation model that simulates the actual traffic conditions in the network during the current estimation interval. The inputs to this model include the travel demand (as estimated by the demand simulator), updated capacities and traffic dynamics parameters, the control strategies implemented and the traffic information and guidance actually disseminated. The driver behavior model captures the responses to ATIS in the form of en route choices.

Demand-Supply Interactions

One of the inputs to the OD estimation model is a set of assignment matrices. These matrices map the OD flows from current and past intervals to link flows in the current interval. The assignment fractions therefore depend on the time interval, and also on the route choice decisions made by individual drivers. The flows measured on the network are a result of the interaction between the demand and supply components.

It may be necessary to iterate between the network state estimation and the OD estimation models until convergence is achieved. The output of this process is an estimate

of the actual traffic conditions on the network, and information about origin destination flows, link flows, queues, speeds and densities.

A.4 Prediction and guidance generation

The prediction-based guidance module described in Fig A.4 consists of several interacting steps:

- Pre-trip demand simulation
- OD flow prediction
- Network state prediction
- Guidance generation

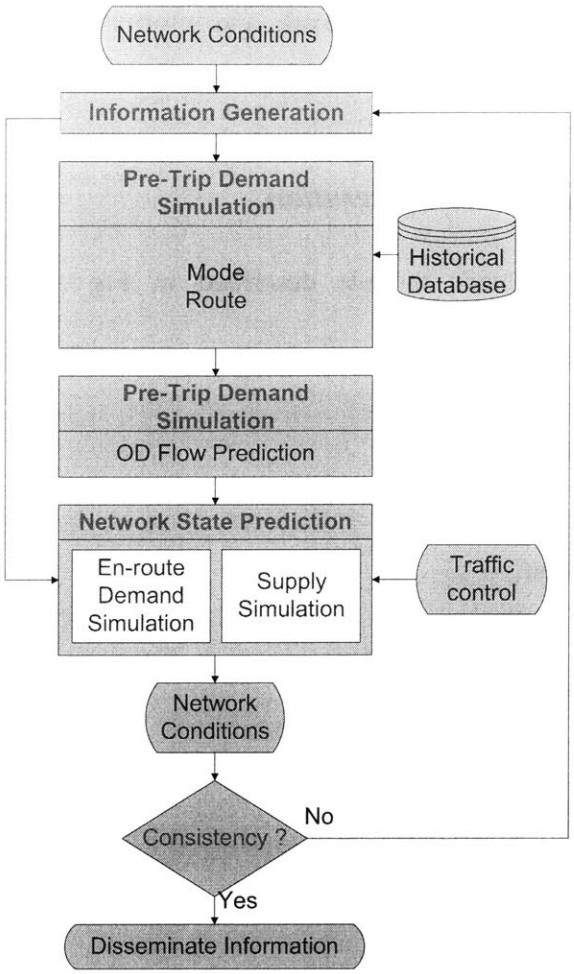

Fig A.4: Prediction-based information generation

The OD prediction model uses the aggregate historical demand adjusted by the pre-trip demand simulator as input to account for departure time, mode and route choices in response to guidance, and provides the required estimates of future OD flows. The network state prediction function undertakes the important task of traffic prediction for a given control and guidance strategy and predicted set of OD flows, using the current network conditions estimated by the state estimation module as a starting point. The performance of the network over the prediction horizon is evaluated using a traffic simulation model and en-route behavioral models. The traffic information and guidance generation function uses the predicted traffic conditions to generate information and guidance according to the various ATIS in place. Traffic control is loosely coupled with

DynaMIT in the current version of the system. Control strategies are assumed to be generated outside the DTA system, using the predictions as an input.

The generated traffic information and guidance must be consistent and unbiased. Under such conditions, there would be no better path that a driver could have taken based on the provided information. An iterative process is employed in order to obtain guidance that satisfies these requirements. Each iteration consists of a trial strategy, the state prediction (comprising both demand prediction and network state prediction) under the trial strategy, and the evaluation of the predicted state for consistency.

Since, in general, the updated historical OD flows depend on future guidance and information, the update of the historical OD flows (using the departure time and mode choice models) and the OD prediction models are included in the iteration.

This general case represents the situation where pre-trip guidance is available to the drivers. In the special case where only en-route guidance is available, the pre-trip demand simulator is bypassed in the iterations. The initial strategy could then be generated from the prediction and guidance generation of the previous period.

Appendix B: Overview of MITSIMLab

MITSIMLab is a simulation-based laboratory developed for evaluating the impacts of alternative traffic management system designs at the operational level and assisting in subsequent refinement. Examples of systems that can be evaluated with MITSIMLab include advanced traffic management systems (ATMS) and route guidance systems. MITSIMLab is a synthesis of a number of different models and represents a wide range of traffic management system designs. It has the ability to model the response of drivers to real-time traffic information and controls and can incorporate the dynamic interaction between the traffic management system and the drivers on the network.

The various components of MITSIMLab are organized in three modules:

1. Microscopic Traffic Simulator (MITSIMLab)

2. Traffic Management Simulator (TMS)
3. Graphical User Interface (GUI)

A microscopic simulation approach, in which movements of individual vehicles are represented, is adopted for modeling traffic flow in the traffic flow simulator MITSIMLab. The traffic and network elements are represented in detail in order to capture the sensitivity of traffic flows to the control and routing strategies. The road network is represented by nodes, links, segments (links are divided into segments with uniform geometric characteristics) and lanes. Traffic controls and surveillance devices are represented at the microscopic level.

The traffic simulator accepts time-dependent origin to destination trip tables as inputs. The OD tables represent either expected conditions or are defined as part of a scenario for evaluation. A probabilistic route choice model is used to capture drivers' route choice decisions. The origin/destination flows are translated into individual vehicles wishing to

enter the network at a specific time. Behavior parameters (e.g., desired speed, aggressiveness) and vehicle characteristics are assigned to each vehicle/driver combination. MITSIMLab moves vehicles according to car-following and lane-changing models. The car-following model captures the response of a driver to conditions ahead as a function of relative speed, headway and other traffic measures. The lane changing model distinguishes between mandatory and discretionary lane changes and simulates driver action as an output of a complex decision-making framework. Merging, drivers' responses to traffic signals, speed limits, incidents, and toll booths are also captured.

The traffic management simulator (TMS) mimics the traffic control system under evaluation. A wide range of traffic control and route guidance systems can be evaluated. These include regular traffic signals, ramp control, freeway mainline control, lane control signs, variable speed limit signs, portal signals at tunnel entrances, intersection control, variable Message Signs and in-vehicle route guidance. TMS has a generic structure that can represent different designs of such systems with logic at varying levels of sophistication (pre-timed, actuated or adaptive). An extensive graphical user interface is used for both debugging purposes and demonstration of traffic impacts through vehicle animation. A detailed description of MITSIMLab appears in Yang and Koutsopoulos (1996) and Yang et al (2000).

Appendix C: MATLAB Code for SPSA Algorithm

```
clear all
global no_ODs          % # OD pairs
global no_intervals    % # Time intervals
global no_segments     % # Segments (spddsy)
global no_groups_cap   % # Segments (capacities)
global no_groups       % # Segment groups
global no_other_params % # other parameters
global demand_factor   % scaling factor for demand
global paramsPerDay    % # parameters to be calibrated per day of data
global start_time      % starting time of estimation interval
global main_dir
global ns              % Sensors
global od_init         % Initial ODs

% Filenames
initial_fn_value_file = 'spsa_output/fn_theta_0.dat';
delta_file = 'spsa_output/delta.dat';
scaled_delta_file = 'spsa_output/delta_scaled.dat';
gradient_log_file = 'spsa_output/grad_log.dat';
final_obj_fn_file = 'spsa_output/final_fn.dat';
final_theta_file = 'spsa_output/x_final.dat';
iter_theta_file = 'spsa_output/x_values.dat';
iter_obj_fn_file = 'spsa_output/fn_values.dat';
fn_path_file = 'spsa_output/fn_path.dat';
rmsn_file = 'spsa_output/best/rmsn_best.dat';
best_theta_file = 'spsa_output/best/best_od.dat'
ResetLinktimes_EXEC = './ccfiles/resetLinktimes'
main_dir = '/home/vikrantv/thesis/costas_network/AVI/';
no_ODs = 6;
start_time = input('enter the start time in hours: ');
start_time = start_time * 3600;
no_intervals = input('enter number of intervals: ');
demand_factor = 4.0;
no_other_params = 1;
ns = 3
cd(main_dir)
paramsPerDay = no_ODs*no_intervals;
mitsim_sensor = load('mitsim/Output/sensor_perturbed.out');
time=start_time+900;%at end of the first interval
cd spsa_output/;
for i=1:no_intervals
  nam=['counts',int2str(i),'.dat'];
  system(['rm ', nam]);
  f=fopen(nam,'at');
```

```
  countmatrix=mitsim_sensor(mitsim_sensor(:,1)==time,4);
  for j=1:ns
   fprintf(f,'%d\n',countmatrix(j));
  end
  fclose(f);
  time=time+900;
end
cd ..;
for i=1:no_intervals
   eval(['load mitsim/seed/od_seed', int2str(i), '.dat;']);
end % for loop
od_seed = [];
for i=1:no_intervals
   eval (['od_seed = [od_seed; od_seed', int2str(i), '(1:no_ODs)];']);
end % for loop
od_init = od_seed;
theta_0 = od_seed(1:no_intervals*no_ODs);
[p q] = size(theta_0); % Set p, the dimension of the parameter vector

n=1000                %total no. of loss measurements
cases=1;
grad_reps = 1;        % no. of reps for averaging the gradient approximation
alpha =.602;
gamma =.101;
a = 15.0;
c = 1.9               % used to be 1.9 before %chosen by standard guidelines
A = 100;
mag_scale = 0.15 % we scale delta by this percent of parameter magnitude
lossfinalsq=0;        %variable for cum.(over 'cases')squared loss values
lossfinal=0;          %variable for cum. loss values
theta_lo = 0.5*theta_0;
theta_hi = 2.0*theta_0;
rand('seed', 47);
rmsn = [];
cd mfiles;
initial_fn_value = snob_func_tt(theta_0);
cd ..;
best_snob = initial_fn_value;
best_theta = theta_0;
cd mfiles/;
rmsn_temp = rmsn_function();
cd ..;
rmsn = [rmsn; rmsn_temp];
eval(['save ', rmsn_file,' rmsn -ascii']);
system(['cp -p DynaMIT/demand.dat DynaMIT/demand_iter_0.dat']);
tmp_x = theta_0';
```

```
tmp_fn = initial_fn_value;
eval(['save ', initial_fn_value_file, ' tmp_fn -ascii;']);
x_values = [];
fn_values = [];
fn_path = initial_fn_value;   % fn_path = [fn_path; initial_fn_value];
grad_log = [];
theta=theta_0;  % Start at seed parameter values

for k=0:n-1
 if (k/10.0 == round(k/10.0))
  cd DynaMIT/;
  system('rm __equilibriumUnfinished.dat');
  system('rm __estimatedUnfinished.dat');
  system(['./DynaMIT_P ./dtaparam_P.dat']);
  cd ..;
  system([ResetLinktimes_EXEC,
' ./DynaMIT/linktime.dat ./DynaMIT/equilibrium_linktime.out ./DynaMIT/linktime1.dat']
);
  system(['mv ./DynaMIT/linktime1.dat ./DynaMIT/linktime.dat']);
 end % if
 a1 = theta.^2;
 ak = a/(k+1+A)^alpha;
 ak = ak*a1;
 ck = c/(k+1)^gamma;
 ghat = 0; % store sum of gradient reps for averaging
 tmp_fn_grad_reps = []; % Store the fn values for averaging
 for xx = 1:grad_reps,
  delta = 2*round(rand(p,1))-1;
  eval(['save ', delta_file, ' delta -ascii;']);
  delta = mag_scale*delta.*theta; % scale delta by percentage of parameter magnitude
  eval(['save ', scaled_delta_file, ' delta -ascii;']);
  thetaplus = theta + ck*delta;
  thetaminus = theta - ck*delta;
  % These four lines below invoke component-wise constraints
  thetaplus=min(thetaplus,theta_hi);
  thetaplus=max(thetaplus,theta_lo);
  thetaminus=min(thetaminus,theta_hi);
  thetaminus=max(thetaminus,theta_lo);
  x_values = [x_values; thetaminus' tmp_x thetaplus'];
  tmp_x = theta'; % reset to theta for next grad rep
  cd mfiles/;
  yplus=snob_func_tt(thetaplus);
  yminus=snob_func_tt(thetaminus);
  cd ..;
  tmp_fn_grad_reps = [tmp_fn_grad_reps; yminus yplus];
  ghat = ghat + ((yplus - yminus)./(2*ck*delta));
```

```
  end % end of reps for grad averaging

  aa = mean(tmp_fn_grad_reps(:,1));   % aa = mean(tmp_fn_grad_reps(:,1:1));
  bb = mean(tmp_fn_grad_reps(:,2));   % bb = mean(tmp_fn_grad_reps(:,2:2));
  fn_values = [fn_values; aa bb];
  ghat = ghat/grad_reps;
  grad_log = [grad_log; ghat'];
  eval(['save ', gradient_log_file, ' grad_log -ascii;']);
  theta=theta-ak.*ghat;
  theta=min(theta,theta_hi);
  theta=max(theta,theta_lo);
  cd mfiles/;
  path_y = snob_func_tt(theta);
  cd ..;
  if path_y < best_snob
    best_theta = theta;
    best_snob = path_y;
  end % if

  cd mfiles/;
  rmsn_temp = rmsn_function();
  cd ..;
  rmsn = [rmsn; rmsn_temp];
  eval(['save ', rmsn_file,' rmsn -ascii']);
  system(['cp -p DynaMIT/demand.dat DynaMIT/demand_iter_', int2str(k+1), '.dat']);
  fn_path = [fn_path; path_y];
  eval(['save ', fn_path_file, ' fn_path -ascii;']);
  tmp_x = theta';
  eval(['save ', iter_theta_file, ' x_values -ascii;']);
  eval(['save ', iter_obj_fn_file, ' fn_values -ascii;']);
end % iterations (k = 0:n-1)

eval(['save ', final_theta_file, ' theta -ascii;']);
lossvalue = path_y; %%  lossvalue=snob_func_tt(theta);
lossfinalsq=lossfinalsq+lossvalue^2;
lossfinal=lossfinal+lossvalue;
disp(['mean loss value over ', int2str(cases), ' runs']);
final_fn = lossfinal/cases;
eval(['save ', final_obj_fn_file, ' final_fn -ascii;']);
eval(['save ', best_theta_file, ' best_theta -ascii;']);
cd mfiles/;
```

Appendix D: MATLAB Code for Genetic Algorithm

```
global no_ODs
global no_intervals
global start_time
global od_init
global main_dir
start_time = input('enter the start time: ');
no_intervals = input('enter the number of intervals: ');
ResetLinktimes_EXEC = './ccfiles/resetLinktimes'
start_time = start_time * 3600;
main_dir = '/home/vikrantv/thesis/costas_network/ga/AVI/'
no_ODs = 6;
no_interval = 4;
no_particles = 100;
no_generations  = 30;
ns = 3;
crossover_para = 0.7;
mutation_para = 0.002;
rand('seed', 66);

cd(main_dir);
mitsim_sensor = load('mitsim/Output/sensor_perturbed.out');
time=start_time+900;%at end of the first interval
cd output/;
for i=1:no_intervals
 nam=['counts',int2str(i),'.dat'];
 system(['rm ', nam]);
 f=fopen(nam,'at');
 countmatrix=mitsim_sensor(mitsim_sensor(:,1)==time,4);
 for j=1:ns
  fprintf(f,'%d\n',countmatrix(j));
 end
 fclose(f);
 time=time+900;
end
cd ..;

for i=1:no_intervals
 eval(['load ./mitsim/seed/od_seed', int2str(i), '.dat;']);
end % for loop
od_seed = [];
for i=1:no_intervals
 eval (['od_seed = [od_seed; od_seed', int2str(i), '(1:no_ODs)];']);
end % for loop
```

```
od_init = od_seed;
particles = [];
fn_path_best = [];
cd DynaMIT/;
system('rm __equilibriumUnfinished.dat');
system('rm __estimatedUnfinished.dat');
system(['cp ./linktime_ff.dat ./linktime.dat']);
cd ../mfiles/;
y = snob_func_tt(od_seed);
fn_path_best = [fn_path_best; y];
cd ../;

for i=1:no_particles
  random = rand(1,no_ODs*no_intervals)*2;
  part = od_seed(1:no_ODs*no_intervals)'.*random;
  particles = [particles; part;];
  cd mfiles/;
  CHANGEPARAMETERS(part);
  cd ../;
  system(['cp ./DynaMIT/linktime_ff.dat ./DynaMIT/linktime.dat']);
  for j=1:3
    cd DynaMIT/;
    system('rm __equilibriumUnfinished.dat');
    system('rm __estimatedUnfinished.dat');
    system(['./DynaMIT_P ./dtaparam_P.dat']);
    cd ..;
    system([ResetLinktimes_EXEC,
' ./DynaMIT/linktime.dat ./DynaMIT/equilibrium_linktime.out ./DynaMIT/linktime_tem
p.dat']);
    system(['mv ./DynaMIT/linktime_temp.dat ./DynaMIT/linktime.dat']);
  end % for loop : 3
  system(['mv ./DynaMIT/linktime.dat ./DynaMIT/linktime', int2str(i), '.dat']);
end % for loop : no_particles

fn_path_full = [];
fn_path_best = [];
overall_best_fn = 10000;
overall_best_od = [];

for i = 1:no_generations
%Evaluate each particle
  fn_best = 10000;
  fn_path = [];
  wts = [];
  for j=1:no_particles
    part = (particles(j,:))';
```

```
  cd mfiles/;
  system(['cp ../DynaMIT/linktime', int2str(j), '.dat ../DynaMIT/linktime.dat']);
  fit = snob_func_tt(part);
  if (fit < fn_best)
    fn_best = fit;
    if (fit < overall_best_fn)
      overall_best_od = part;
      overall_best_fn = fit;
      system(['cp ../DynaMIT/linktime.dat ../DynaMIT/linktime_best.dat']);
    end % if
  end % if
  cd ..;
  fn_path = [fn_path fit];
  wts = [wts; 1.0/fit];
end % for loop : particles
wts = wts/sum(wts);
fn_path_best = [fn_path_best; fn_best;];

%Generate new population 2 at a time
new_particles = [];
new_indices = [];
%Selection
for l=1:no_particles/2
  pair = [];
  pair_indices = [];
  for j=1:2
    random = rand(1,1);
    total = 0;
    for k=1:no_particles
      total = total + wts(k);
      if ((random <= total) | k == no_particles)
        pair = [pair; particles(k,:)];
        new_indices = [new_indices; k];
        pair_indices = [pair_indices; k];
        break;
      end % if
    end % for loop : paricles
  end % for loop : 2
  %Crossover: 1 Point Crossover with 0.6 probability per pair
  rand1 = random(1,1);
  if (rand1<=crossover_para)
    rand2 = floor(random(1,1)*no_ODs*no_intervals)+1;
    new_pair = [pair(1,1:rand2) pair(2,rand2+1:no_ODs*no_intervals);
    pair(2,1:rand2) pair(1,rand2+1:no_ODs*no_intervals)];
  else
    new_pair = pair;
```

```
    end % if
    %Mutation with 0.001 probability
    for m=1:2
     for n=1:no_ODs
      rand3 = rand(1,1);
      if (rand3<=mutation_para)
       rand4= rand(1,1)*2;
       new_pair(m,n) = new_pair(m,n)*rand4;
      end % if
     end % for loop no_ODs
    end % for loop 2
    new_particles = [new_particles; new_pair];
  end % for loop : no_particles/2

  fn_path_full = [fn_path_full; fn_path];
  particles = new_particles;
  clear new_particles;
  for l=1:no_particles
   system(['mv ./DynaMIT/linktime', int2str(l), '.dat ./DynaMIT/linktime_', int2str(l),
'.dat']);
  end % for loop : no_particles
  for l=1:no_particles
   system(['cp ./DynaMIT/linktime_', int2str(new_indices(l)), '.dat ./DynaMIT/linktime',
int2str(l), '.dat']);
  end % for loop : no_particles
  new_indices
  clear new_indices;
  save 'fn_path_full.dat' fn_path_full -ascii;
  save 'overall_best_fn.dat' overall_best_fn -ascii;
  save 'overall_best_od.dat' overall_best_od -ascii;
  save 'fn_path_best.dat' fn_path_best -ascii;

  for m=1:no_particles
   random = rand(1,no_ODs*no_intervals)*2;
   part = od_seed(1:no_ODs*no_intervals)'.*random;
   particles = [particles; part;];
   cd mfiles/;
   CHANGEPARAMETERS(part);
   cd ../;
   system(['cp ./DynaMIT/linktime', int2str(m), '.dat ./DynaMIT/linktime.dat']);
   for n=1:1
    cd DynaMIT/;
    system('rm __equilibriumUnfinished.dat');
    system('rm __estimatedUnfinished.dat');
    system(['./DynaMIT_P ./dtaparam_P.dat']);
    cd ..;
```

```
    system([ResetLinktimes_EXEC,
' ./DynaMIT/linktime.dat ./DynaMIT/equilibrium_linktime.out ./DynaMIT/linktime_tem
p.dat']);
    system(['mv ./DynaMIT/linktime_temp.dat ./DynaMIT/linktime.dat']);
   end % for loop : 3
   system(['mv ./DynaMIT/linktime.dat ./DynaMIT/linktime', int2str(m), '.dat']);
 end % for loop : no_particles
end % for loop : generations
```

Appendix E: MATLAB Code for Particle Filter Method

```
global no_ODs
global start_time
global od_init
global main_dir
start_time = input('enter the start time: ');
no_intervals = input('enter the number of intervals: ');
ResetLinktimes_EXEC = './ccfiles/resetLinktimes'
start_time = start_time * 3600;
main_dir = '/home/vikrantv/thesis/costas_network/bootstrap/AVI/'
no_ODs = 6;
no_interval = 4;
no_particles = 1000;
ns = 3;
rand('seed', 91);

cd(main_dir);
mitsim_sensor = load('mitsim/Output/sensor_perturbed.out');
time=start_time+900;
cd output/;
for i=1:no_intervals
 nam=['counts',int2str(i),'.dat'];
 system(['rm ', nam]);
 f=fopen(nam,'at');
 countmatrix=mitsim_sensor(mitsim_sensor(:,1)==time,4);
 for j=1:ns
  fprintf(f,'%d\n',countmatrix(j));
 end
 fclose(f);
 time=time+900;
end
cd ..;

for i=1:no_intervals
 eval(['load ./mitsim/seed/od_seed', int2str(i), '.dat;']);
end % for loop
od_seed = [];
for i=1:no_intervals
 eval (['od_seed = [od_seed; od_seed', int2str(i), '(1:no_ODs)];']);
end % for loop
od_init = od_seed;
ar = [];
for i=2:no_intervals
 ar = [ar; od_seed((i-1)*no_ODs+1:i*no_ODs)./od_seed((i-2)*no_ODs+1:(i-
1)*no_ODs)]
```

```
end % for loop

particles = [];
fn_path_best = [];
cd DynaMIT/;
system('rm __equilibriumUnfinished.dat');
system('rm __estimatedUnfinished.dat');
system(['cp ./linktime_ff.dat ./linktime.dat']);
cd ../mfiles/;
y = snob_func_tt(od_seed,no_intervals);
cd ../;

for n=1:3
  cd DynaMIT/;
  system('rm __equilibriumUnfinished.dat');
  system('rm __estimatedUnfinished.dat');
  system(['./DynaMIT_P ./dtaparam_P', int2str(no_intervals), '.dat']);
  cd ..;
  system([ResetLinktimes_EXEC,
' ./DynaMIT/linktime.dat ./DynaMIT/equilibrium_linktime.out ./DynaMIT/linktime_tem
p.dat']);
  system(['mv ./DynaMIT/linktime_temp.dat ./DynaMIT/linktime.dat']);
end % for loop : 3

cd DynaMIT/;
for i=1:no_particles
  system(['cp linktime.dat linktime', int2str(i), '.dat']);
end % for loop
cd ../;
fn_path_best = [fn_path_best; y];
fn_path_full = [];
overall_best_fn = 10000;
overall_best_od = [];

for t=1:no_intervals
  old_particles = particles;
  particles = [];
  if (t==1)
    for i=1:no_particles
      random = rand(1,no_ODs)*2;
      particles = [particles; od_seed(1:no_ODs)'.*random;];
    end % for loop
  else % if t>1
    for i=1:no_particles
      random = rand(1,no_ODs)*2;
```

```
    temp = (old_particles(i,(t-2)*no_ODs+1:(t-1)*no_ODs).*ar((t-2)*no_ODs+1:(t-
1)*no_ODs)');
    temp = temp.*random;
    particles = [particles; old_particles(i,:) temp;];
  end % for loop
 end % if

 fn_best = 10000;
 fn_path = [];
 wts = [];
 for j=1:no_particles
  part = (particles(j,:))';
  cd mfiles/;
  system(['cp ../DynaMIT/linktime', int2str(j), '.dat ../DynaMIT/linktime.dat']);
  fit = snob_func_tt(part,t);
  if (fit < fn_best)
   fn_best = fit;
   if (fit < overall_best_fn)
    overall_best_od = part;
    overall_best_fn = fit;
    system(['cp ../DynaMIT/linktime.dat ../DynaMIT/linktime_best.dat']);
   end % if
  end % if
  cd ..;
  fn_path = [fn_path fit];
  wts = [wts; 1.0/fit];
 end % for loop : particles
 wts = wts/sum(wts);
 fn_path_full = [fn_path_full; fn_path];
 fn_path_best = [fn_path_best; fn_best;];
 new_particles = [];
 new_indices = [];

 for l=1:no_particles
  part = [];
  index = [];
  random = rand(1,1);
  total = 0;
  for k=1:no_particles
   total = total + wts(k);
   if ((random <= total) | k == no_particles)
    part = particles(k,:);
    new_indices = [new_indices; k];
    index = k;
    break;
   end % if
```

```
    end % for loop : particles
   new_particles = [new_particles; part];
   end % for loop : particles

   particles = new_particles;
   clear new_particles;
   for l=1:no_particles
    system(['mv ./DynaMIT/linktime', int2str(l), '.dat ./DynaMIT/linktime_', int2str(l),
'.dat']);
   end % for loop : no_particles
   for l=1:no_particles
    system(['cp ./DynaMIT/linktime_', int2str(new_indices(l)), '.dat ./DynaMIT/linktime',
int2str(l), '.dat']);
   end % for loop : no_particles
   new_indices
   clear new_indices;
   save 'fn_path_full.dat' fn_path_full -ascii;
   save 'overall_best_fn.dat' overall_best_fn -ascii;
   save 'overall_best_od.dat' overall_best_od -ascii;
   save 'fn_path_best.dat' fn_path_best -ascii;

   for m=1:no_particles
    part = particles(m,:);
    cd mfiles/;
    CHANGEPARAMETERS(part,t);
    cd ../;
    system(['cp ./DynaMIT/linktime', int2str(m), '.dat ./DynaMIT/linktime.dat']);
    for n=1:1
     cd DynaMIT/;
     system('rm __equilibriumUnfinished.dat');
     system('rm __estimatedUnfinished.dat');
     system(['./DynaMIT_P ./dtaparam_P', int2str(t), '.dat']);
     cd ..;
     system([ResetLinktimes_EXEC,
' ./DynaMIT/linktime.dat ./DynaMIT/equilibrium_linktime.out ./DynaMIT/linktime_tem
p.dat']);
     system(['mv ./DynaMIT/linktime_temp.dat ./DynaMIT/linktime.dat']);
    end % for loop : 3
    system(['mv ./DynaMIT/linktime.dat ./DynaMIT/linktime', int2str(m), '.dat']);
   end % for loop : no_particles
end % for loop : intervals

wts = [];
for j=1:no_particles
   part = (particles(j,:))';
   cd mfiles/;
```

```
  system(['cp ../DynaMIT/linktime', int2str(j), '.dat ../DynaMIT/linktime.dat']);
  fit = snob_func_tt(part,no_intervals);
  cd ..;
  wts = [wts; 1.0/fit];
end % for loop : particles
wts = wts/sum(wts);

final_OD = [];
for j=1:no_ODs*no_intervals
  temp = sum(wts.*particles(:,j));
  final_OD = [final_OD; temp];
end % for loop
cd mfiles/;
final_fn = snob_func_tt(final_OD, no_intervals);
cd ../;
save final_OD.dat final_OD -ascii;
save final_fn.dat final_fn -ascii;
save final_particles.dat particles -ascii;
cd mfiles/;
```

Bibliography

- Abdulhai, B., Sheu, J. B., and Recker, W. (1999). Simulation of ITS on the Irvine FOT Area using the 'PARAMICS 1.5' Scalable Microscopic Traffic Simulator. Phase I: Model Calibration and Validation. Technical report, PATH research report, UCBITS-PRR-99-12.
- Antoniou, C. (2002). Development of ITS analysis tools for Lower Westchester County. Technical Memorandum on DynaMIT Enhancements, submitted to New York State Department of Transportation.
- Antoniou, C. (2004). On-Line Calibration for Dynamic Traffic Assignment. PhD thesis, Massachusetts Institute of Technology.
- Antoniou, C., Ben-Akiva, M. E., and Koutsopoulos, H. N. (2004). Incorporating Automated Vehicle Identification data into Origin-Destination Estimation. Transportation Research Record 1882, pp 37-44, Washington D.C.
- Antoniou, C., Ben-Akiva, M. E., and Koutsopoulos, H. N. (2006). Dynamic Traffic Demand Prediction Using Conventional and Emerging Data Sources. IEEE Proceedings Intelligent Transport Systems.
- Antoniou, C., Ben-Akiva, M. E., and Koutsopoulos, H. N. (2006). Non–linear Kalman Filtering Algorithms for On–line Calibration of Dynamic Traffic Assignment Models. Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC), Toronto, Canada.
- Antoniou, C., Wen, Y., Ramanujam, V., and Vaze, V. (2006). Development of ITS analysis tools for Lower Westchester County, Technical Memorandum on the Calibration of MITSIMLab for the Lower Westchester County Network, submitted to New York State Department of Transportation.
- Arfken, G. (1985). The Method of Steepest Descents. Mathematical Methods for Physicists, 3rd ed. Orlando, FL: Academic Press, pp. 428-436.
- Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing, 50(2):174–188.

- Asakura, Y., Hato, E., and Kashiwadani, M. (2000). OD matrices estimation model using AVI data and its application to the Han-Shin expressway network. Transportation, vol. 27, no. 4, pp. 419–438.
- Ashok, K. (1992). Dynamic Trip Table Estimation for Real Time Traffic Management Systems. Master's thesis, Massachusetts Institute of Technology.
- Ashok, K. (1996). Estimation and Prediction of Time-Dependent Origin-Destination Flows. PhD thesis, Massachusetts Institute of Technology.
- Ashok, K. and Ben-Akiva, M. E. (1993). Dynamic O-D matrix estimation and prediction for real-time traffic management systems. In Daganzo, C., editor, Transportation and Traffic Theory, pages 465–484. Elsevier Science Publishing.
- Ashok, K. and Ben-Akiva, M. E. (2000). Alternative Approaches for Real-Time Estimation and Prediction of Time-Dependent Origin-Destination Flows. Transportation Science, 34(1):21-36.
- Ashok, K. and Ben-Akiva, M. E. (2002). Estimation and prediction of time-dependent origin-destination flows with a stochastic mapping to path flows and link flows. Transportation Science, 36(2):184–198.
- Balakrishna, R. (2002). Calibration of the Demand Simulator in a Dynamic Traffic Assignment System. Master's thesis, Massachusetts Institute of Technology.
- Balakrishna, R. (2006). Off-line Calibration for Dynamic Traffic Assignment Models. PhD thesis, Massachusetts Institute of Technology.
- Balakrishna, R., Antoniou C., Ben-Akiva, M. E., Koutsopoulos, H. N., and Wen, Y. (2007). Calibration of Microscopic Traffic Simulation Models: Methods and Application. Accepted for publication in Transportation Research Record.
- Balakrishna, R., Ben-Akiva, M. E., and Koutsopoulos, H. N. (2006a) Time-Dependent Origin-Destination Estimation without Assignment Matrices. Second International Symposium on Transport Simulation.
- Balakrishna, R., Koutsopoulos, H. N., and Ben-Akiva, M. E. (2006b) Simultaneous Off-line Demand and Supply Calibration of Dynamic Traffic Assignment Systems. Presented at the 85th annual meeting of the Transportation Research Board.

- Balakrishna, R., Ben-Akiva, M. E., Koutsopoulos, H. N., and Toledo, T. (2004). Traffic Simulation Model Calibration Framework Using Aggregate Data. Proceedings of the Triennial Symposium on Transportation Analysis (TRISTAN V).
- Balakrishna, R., Koutsopoulos, H. N., and Ben-Akiva, M. E. (2005a). Calibration and Validation of Dynamic Traffic Assignment Systems. In Mahmassani, H. S., editor, Transportation and Traffic Theory: Flow, Dynamics and Human Interaction, pages 407-426. Proceedings of the 16th International Symposium on Transportation and Traffic Theory, Elsevier.
- Balakrishna, R., Koutsopoulos, H. N., Ben-Akiva, M. E., Ruiz, B. M. F., and Mehta, M. (2005b). A Simulation-Based Evaluation of Advanced Traveler Information Systems. Transportation Research Record.
- Barcelo, J., and Casas, J. (2002). Dynamic Network Simulation with AIMSUN. Proceedings of the International Symposium on Transport Simulation, Yokohama. Kluwer.
- Ben-Akiva, M. E., Davol, A., Toledo, T., Koutsopoulos, H. N., Burghout, W., Andreasson, I., Johansson, T., and Lundin, C. (2002). Calibration and Evaluation of MITSIMLab in Stockholm. Proceedings of the 81st Transportation Research Board Annual Meeting, Washington, D.C.
- Ben-Akiva, M. E., Bierlaire, M., Bottom, J., Koutsopoulos, H. N., and Mishalani, R. (1997). Development of a Route Guidance Generation System for Real-Time Application. Proceedings of the Eighth IFAC Symposium on Transportation Systems, Chania, Greece.
- Ben-Akiva, M. E., Bierlaire, M., Burton, D., Koutsopoulos, H. N., and Mishalani, R. (2001). Network State Estimation and Prediction for Real-Time Transportation Management Applications. Networks and Spatial Economics, 1(3/4):291-318.
- Ben-Akiva, M. E., Bierlaire, M., Burton, D., Koutsopoulos, H. N., and Mishalani, R. (2001). Network State Estimation and Prediction for Real-Time Transportation Management Applications. *Networks and Spatial Economics,* Vol. 1, No. 3/4, pp 293-318.

- Ben-Akiva, M. E., Bierlaire, M., Koutsopoulos, H. N., and Mishalani, R. (2002). Real- Time Simulation of Traffic Demand-Supply Interactions within DynaMIT. M. Gendreau, and P. Marcotte, editors, Transportation and Network Analysis: Miscellenea in honor of Michael Florian, pages 19-36. Kluwer.
- Ben-Akiva, M., and Lerman, S. (1985), Discrete Choice Analysis: Theory and Applications to Travel Demand, The MIT Press, Cambridge, MA.
- Box, M. J. (1965). A New Method of Constrained Optimization and a Comparison with Other Methods. Computer Journal, 8(1):42-52.
- Caliper (2007). About Transmodeler. http://caliper.com/transmodeler/default.htm. Accessed on 27 April 2007.
- Carey, M. (1986). A constraint qualification for a dynamic traffic assignment model. *Transportation Science*, 20(55-58).
- Cascetta, E., Inaudi, D., and Marquis, G. (1993). Dynamic estimators of origin destination matrices using traffic counts. *Transportation Science*, vol. 27, no. 4, pp. 363–373.
- Chui, C. K., and Chen, G. (1999). Kalman Filtering with Real-Time Applications. Springer-Verlag.
- Corana, A., Marchesi, M., Martini, C., and Ridella, S. (1987). Minimizing Multimodal Functions of Continuous Variables with the Simulated Annealing Algorithm. ACM Transactions on Mathematical Software, 13:262-280.
- Daganzo, C. (1994). The Cell Transmission Model: A Dynamic Representation of Highway Traffic Consistent with the Hydrodynamic Theory. Transportation Research, 28B(4):269-287.
- De Jong, K. A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. thesis, University of Michigan, Ann Arbor.
- Doucet, A., Freitas, N. D., and Gordon, N. (1974). "Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science, Springer.
- Federal Highway Administration (2007). Highway Statistics: Annual Summary and Reports. http://www.fhwa.dot.gov/policy/ohpi/hss/hsspubs.htm. Accessed on 17 May 2007.

- FHWA (2007). CORSIM, http://ops.fhwa.dot.gov/trafficanalysistools/corsim.htm. Accessed on 8 May 2007.
- Gelb, A., editor (1974). Applied Optimal Estimation. M.I.T. Press.
- Goffe, W. L., Ferrier, G. D., and Rogers, J. (1994). Global Optimization of Statistical Functions with Simulated Annealing. Journal of Econometrics, 60(1-2).
- Gordon, N. J. (2003). Beyond the Kalman Filter: Particle filters for tracking applications. Workshop on New Directions in Signal Processing in the XXI century, Lake Louise, Alberta, Canada.
- Gupta, A. (2005). Observability of Origin-Destination Matrices for Dynamic Traffic Assignment. Master's thesis, Massachusetts Institute of Technology.
- Henderson, J., and Fu, L. (2004). Applications of Genetic Algorithms in Transportation Engineering. Presented at the 83rd annual meeting of the Transportation Research Board.
- Hestenes, M. R., and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards* 49, 409–436.
- Huyer, W., and Neumaier, A. (2004). SNOBFIT - Stable Noisy Optimization by Branch and Fit. Submitted to ACM Transactions on Mathematical Software.
- INRO (2007). About EMME/2. http://www.inro.ca/en/products/emme2/index.php. Accessed on 4 May 2007.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Transportation ASME Journal of Basic Engineering, 82(1): pp 33-45.
- Kim, K. O., (2002). Optimization Methodology for the Calibration of Transportation Network Micro-Simulation Models. PhD thesis, Texas A&M University.
- Kim, K. O., and Rilett, L. R. (2003). Simplex-Based Calibration of Tra±c Microsimulation Models with Intelligent Transportation Systems Data. Transportation Research Record, 1855:80-89.
- Kim, K. O., and Rilett, L. R. (2004). A Genetic Algorithm Based Approach to Traffic Micro-Simulation Calibration Using ITS Data. Presented at the 83rd annual meeting of the Transportation Research Board.

- Kleijnen, J. P. C. (1987). Statistical Tools for Simulation Practitioners. Marcel Dekker.
- Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. SIAM Review, 45(3):385-482.
- Kunde, K. K. (2002). Calibration of Mesoscopic Traffic Simulation Models for Dynamic Traffic Assignment. Master's thesis, Massachusetts Institute of Technology.
- Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. SIAM Journal of Optimization, 9(1):112-147.
- Lee, D. H., Yang, X., and Chandrasekhar, P. (2001). Parameter Calibration for PARAMICS using Genetic Algorithm. Presented at the 80th annual meeting of the Transportation Research Board.
- Mahanti, B. P. (2004). Aggregate Calibration of Microscopic Traffic Simulation Models. Master's thesis, Massachusetts Institute of Technology.
- Mahmassani, H. S. (2002). Dynamic Network Traffic Assignment and Simulation Methodology for Advanced System Management Applications. Presented at the 81st annual meeting of the Transportation Research Board.
- Merchant, D. K., and Nemhauser, G. L. (1978). Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(200-207).
- Messmer, A., and Papageorgiou, M. (2001). Freeway Network Simulation and Dynamic Traffic Assignment with METANET Tools. Transportation Research Record, (1776):178-188.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics, 21:1087-1092.
- Mitchell, M. (1006). An introduction to Genetic Algorithms. Cambridge MIT Press.
- Nelder, J. A., and Mead, R. (1965). A Simplex Method for Function Minimization. Computer Journal, 7(4):308-313.

- Pitt, M. K., and Shephard, N. (1999). Filtering via simulation: Auxiliary particle filters. Journal of the American Statistical Association, 94(446).
- PTV (2006). http://www.english.ptv.de/cgi-bin/traffic/traf vision.pl. Accessed on 4 May 2006.
- Pindyck, R. S., and Rubinfeld, D. L. (1997). Econometric models and economic forecasts, 4th edition. Irwin McGraw-Hill, Boston MA.
- Smith, M., Duncan, G., and Druitt, S. (1995). PARAMICS: Microscopic Traffic Simulation for Congestion Management. Colloquium on Dynamic Control of Strategic Inter-Urban Road Networks, London, England.
- Sorenson, H. W. (1985) editor. Kalman Filtering: Theory and Application. IEEE Press, New York.
- Spall, J. C. (1988). A Stochastic Approximation Algorithm for Large-Dimensional Systems in the Kiefer-Wolfowitz Setting. In *Proc. IEEE Conf. on Decision and Control*, pp. 1544–1548.
- Spall, J. C. (1992). Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. IEEE Transactions on Automatic Control, 37(3):332-341.
- Spall, J. C. (1998a). An Overview of the Simultaneous Perturbation Method for Efficient Optimization. Johns Hopkins APL Technical Digest, 19(4):482-492.
- Spall, J. C. (1998b). Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization. IEEE Transactions on Aerospace and Electronic Systems, 34(3):817-823.
- Spall, J. C. (1999). Stochastic Optimization, Stochastic Approximation and Simulated Annealing. Webster, J. G., editor, Wiley Encyclopedia of Electrical and Electronics Engineering, pages 529-542. Wiley-Interscience.
- Thierens, D. (1999). Scalability Problems of Simple Genetic Algorithms," Journal of Evolutionary Computation, MIT Press, 7(4):331-352.
- Toledo, T. (2003). Integrated Driving Behavior Modeling, PhD Thesis, Massachusetts Institute of Technology, February.
- Toledo T., and Koutsopoulos, H. N. (2004). Statistical Validation of Traffic Simulation Models. Transportation Research Record 1876, pp 142-150.

- UMD (2007). DYNASMART. http://www.dynasmart.umd.edu/index.html, Accessed 17th May 2007.
- Van der Zijpp, N. J. (1997). Dynamic OD-Matrix Estimation From Traffic Counts and Automated Vehicle Identification Data. Transportation Research Record, No. 1607, pp. 87–94.
- Yagar, S. (1971). Dynamic traffic assignment by individual path minimization and queueing. *Transportation Science*, 5(179-196).
- Yang, Q., and Koutsopoulos, H. N. (1996). A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Assignment Systems. *Transportation Research 4C(3)*, pp. 113-129.
- Yang, Q., Koutsopoulos, H. N., and Ben-Akiva, M. E. (2000). A Simulation Model for Evaluating Dynamic Traffic Management Systems. *Transportation Research Record*, No. 1710, pp 122-130.
- Ypma, T. J. (1995). Historical development of the Newton-Raphson method. *SIAM Review* 37 (4), 531–551.
- Zhou, X., and Mahmassani, H. S. (2006). Dynamic origin destination demand estimation using automatic vehicle identification data. IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 1, pp. 105-114.