Obtaining Dense Road Speed Estimates from Sparse GPS Measurements

Andrew Phan

Masters in Engineering

Department of Electrical and Computer Engineering

McGill University

Montreal, Quebec

2008-08-29

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Masters in Engineering

DEDICATION

This document is dedicated to ambulance dispatchers around the world whose timely decisions have saved (and continue to save) countless lives.

ACKNOWLEDGEMENTS

I thank my adviser, Professor Frank Ferrie, for his invaluable guidance and advice, my parents for always encouraging me to go that extra yard, my parents-in-law for taking care of me these past few years so that I only had to worry about my work, and my beloved wife, Olivia, whose unfailingly support never ceases to amaze me despite my unending and sometimes costly projects. I also wish to extend my thanks to my colleagues at CIM who took valuable time out of their busy schedules to provide me with constructive feedback.

ABSTRACT

A major challenge for traffic management systems is the inference of traffic flow in regions of the network for which there is little data. In this thesis, GPS-based vehicle locator data from a fleet of 40-60 roving ambulances are used to estimate traffic congestion along a network of 20,000 streets in the city of Ottawa, Canada. Essentially, the road network is represented as a directed graph and a belief propagation algorithm is used to interpolate measurements from the fleet. The system incorporates a number of novel features. It makes no distinctions between freeways and surface streets, incorporates both historical and live sensor data, handles user inputs such as road closures and manual speed overrides, and is computationally efficient - providing updates every 5 to 6 minutes on commodity hardware. Experimental results are presented which address the key issue of validating the performance and reliability of the system.

ABRÉGÉ

Un défi important en lien avec les systèmes de gestion de la circulation routière est de définir la situation actuelle du réseau routier dans les régions où peu de données sont disponibles. Les données provenant d'une flotte de 40-60 ambulances munies d'un GPS ont été utilisées afin d'estimer la congestion routière sur un réseau de plus de 20 000 rues dans la ville d'Ottawa, au Canada. Essentiellement, le réseau routier est représenté par un graphe orienté et un algorithme de propagation de confiance est utilisé pour interpoler les données provenant de la flotte d'ambulances. Ce système comprend des caractéristiques innovatives. Le système ne fait aucune distinction entre les autoroutes et les rues, il intègre les données archivées et actuelles, il gère les informations entrées par l'utilisateur au central concernant les fermeture des routes et les changements de vitesse sur le réseau routier et il est efficace dans ses calculs puisqu'il fournit des mises à jour de l'état du réseau routier toutes les 5-6 minutes sur un ordinateur standard. Les résultats de l'expérience, la validation de la performance ainsi que la fiabilité du système sont présentés.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

Maintaining the state of a complex road network given limited sensor input is a key challenge to the design of modern traffic control systems. In this paper we address the specific problem of inferring traffic density along each of 20,000 streets in the greater city of Ottawa, Canada, from GPS-based vehicle locator data supplied by a fleet of 40-60 roving ambulances. The context for this work is a system for managing the redeployment of a fleet of ambulances operated by the Ottawa Paramedic Service (OPS).

1.1 Project Overview

Typically, when an emergency call is received that requires the services of an ambulance, the OPS dispatcher must first decide which available ambulance to deploy. This task is not particularly difficult as the number of ambulances to take into consideration is limited by the maximum allowable response time of 8 minutes. After an ambulance has been deployed however, a gap in the coverage may occur in which case the remaining available ambulances need to be redeployed in order to maintain coverage. The redeployment task is much more difficult because now all the ambulances must be taken into consideration in addition to operating constraints such as load balancing, crew scheduling, and minimizing movement. The goal of the project, RISER[1] , is

[1] Rapid Intelligent Scheduling for Emergency Responders, a joint project led by CAE Inc. under the Precarn Inc. CORE program, with the participation of Actenum Inc., McGill University, Simon Fraser University, and the Ottawa Paramedic Service.

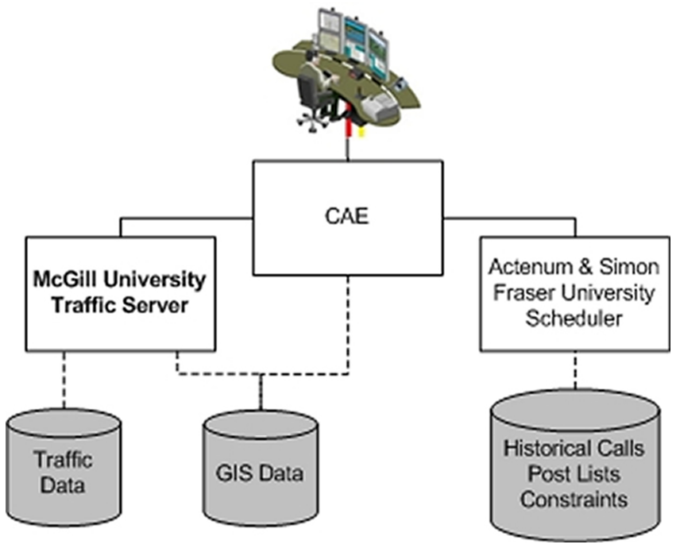

CAE
McGill University Traffic Server
Actenum & Simon Fraser University Scheduler
Traffic Data
GIS Data
Historical Calls Post Lists Constraints

Figure 1–1: Proposed Ambulance Dispatch System

to develop a system capable of dynamically positioning the fleet so as to guarantee response time while respecting operating constraints. As can be seen in Fig. 1–1, the system relies on a traffic server module to provide an estimate of the traffic congestion along each of the approximately 20,000 streets in the city road network. These data are used to estimate time of arrival (ETA) for any path through the network, which in turn is used for determining optimal redeployment of the fleet by the scheduling engine.

1.2 The Traffic Server

The focus of this thesis is the traffic server, which uses a combination of sparse measurements, historical traffic data, and a GIS database to determine the likely traffic flow state at any location in the network. The key idea of the thesis is that given prior information in the form of date/time indexed historical data, one can reliably interpolate a set of sparse, real-time measurements, and obtain reasonable estimates over the extent of the network. This is achieved by means of a belief propagation algorithm which operates on a representation of the traffic network as a directed graph and serves to integrate information from different sources. Here we consider only vehicle locator data and user inputted speeds, but the algorithm is sufficiently general to incorporate any available source of information provided that i) it is geo-referenced to a particular location in the network, and ii) that it can be expressed in terms of relative congestion (described shortly). The particular belief propagation algorithm we use is Relaxation Labelling (RL), which has been widely used in the computer vision field [1, 2, 3].

1.3 Organization of the Thesis

The organization of the thesis is as follows. Chapter 2 formally defines the problem and outlines the contributions of this work. Chapter 3 reviews the current state of the art. Chapter 4 describes the solution we propose. Chapter 5 details the experimental methodology and the validation results. The thesis ends with a discussion in Chapter 6 and suggestions for future work in Chapter 7.

CHAPTER 2
Contributions

There are a number of contributions that, when put together, make the work accomplished in this thesis unique.

2.1 Interpolating Sparse and Noisy Sensor Data

The main contribution of this work is the interpolation of sparse and locally inconsistent sensor data. In the city of Ottawa, a central computer tracks the positions in 11 second intervals of a fleet of 40-60 ambulances equipped with GPS devices. Their positions are converted to travel speeds which in turn are used to infer local road congestions. As there are at least 20,000 streets in Ottawa, one problem is that the number of probes supplying road congestion information is orders of magnitude lower than the number of streets in the city. Although a user may add data by manually inputting road speeds into the system, the user is not expected to do so often which does not alleviate the sparsity of the data. Another problem is that because the sensor data are noisy and have large variance, local road congestions are bound to be inconsistent. For example, a given road may appear to be free flowing according to its historical data but could in fact be congested according to the historical data of its neighbouring roads. Dense locally consistent speed estimates are obtained from sparse and locally inconsistent speeds via the Relation Labelling (RL) algorithm using local constraints as defined by the road network topology (i.e. linkages, number of lanes and speed limits).

2.2 Accommodating Arbitrary Data Sources

The traffic server is sufficiently general that it can accommodate arbitrary sensor information provided that sensor data satisfy two conditions. First, it must be geo-referenced to a particular road segment. Second, it must be expressed in terms of relative congestion.

2.3 Incorporating Measures of Confidence

In contrast to the other methods, we take care to incorporate measures of confidence to represent uncertainties in the sensor data and in the estimates that we provide. For the sensor data, this is achieved by means of a Gaussian distribution where the shape of the Gaussian determines the confidence in the data. For predicting the potential ambulance speed of every road segment in the network, we obtain a distribution of likely speeds where the maximum likelihood estimate corresponds to the most likely ambulance speed and the shape of the distribution indicates confidence in the estimate. Such information may be desirable for advanced route planning systems where, given two routes with similar travel times but different certainties, the route with the higher certainty should be favoured.

2.4 Scale and Performance

The traffic server is able the model the road congestion for the entire city of Ottawa, Canada, including both freeways and surface streets in under 5-10 minutes depending on the model parameters. During the main execution loop, almost all of the computation time is dedicated to the interpolation step. On commodity hardware, we are able to update the road congestion for Ottawa every 5-10 minutes depending on the model parameters which are determined by the needs of the ambulance dispatchers. For larger cities, we can take advantage of the parallel nature of RL algorithm and split the processing over multiple cores using readily available multi-core processors in order to minimize processing time.

2.5 Validation

We developed a thorough and automated process to validate the model using K-folds cross-validation. Challenges due to the size of the road network and the quantity of the historical data had to be overcome. As explained later, it takes 10 days or 240 hours of processing time to perform 5-fold cross-validation for each set of parameters.

2.6 Easily Portable to Other Cities

The traffic server is sufficiently flexible that it can be ported to any arbitrary city with little user modifications. All that is required is for the road network database of the new city to follow the same widely used ESRI Shapefile format and to contain the same key attributes.

2.7 Locally Adaptable

The traffic server can allow a user to modify the road network without rendering the model useless. For instance, the user may close a road off from circulation or temporarily override historical data by manually inputting live speeds. Examples of when a road closure may be necessary are during parades or road work while examples of speed overrides are in the case of accidents or stalled vehicles. The advantage of supporting such modifications is that the system is more dynamic and can be adjusted to closely model actual conditions.

2.8 Publication

The design and results presented in this thesis are summarized in [4]:

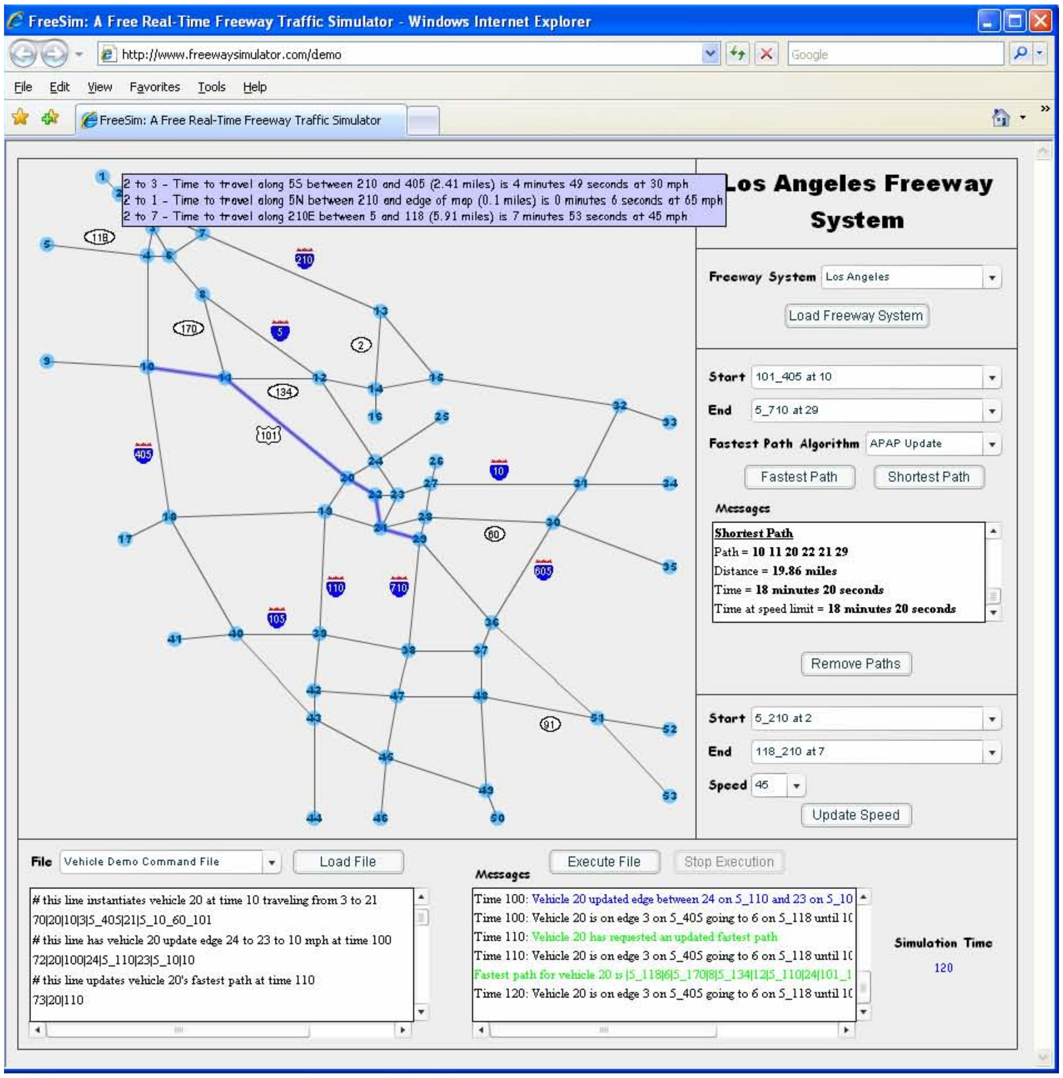
A. Phan and F.P. Ferrie. Obtaining Dense Road Speed Estimates from Sparse GPS Measurements. *IEEE Intelligent Transportation Systems Conference*, pages 157-162, 2008.

CHAPTER 3
Related Work

In the literature, traffic models are generally classified as either microscopic or macroscopic. A microscopic model tracks the movement of individual vehicles and generally includes a model for the driver, the vehicle and how the vehicle interacts with its surroundings. A macroscopic model, on the other hand, tends to treat congestion as a fluid and represents the dynamics of traffic flow as a group rather than as individual vehicles. Below are several systems that we believe are representative of the current state of the art in traffic modelling.

3.1 FreeSim (2007)

FreeSim [5, 6] shown in Fig. 3–1 is a framework that offers both macroscopic and microscopic traffic modelling of freeways. It requires that a large number of vehicles in the road network have a GPS device that transmits their current speed and location to a central server. The central server then uses this information to update the travel speed of every road segment in the road network database. At any time during the simulation, a user may request the quickest route to get from one location to another and the system responds by using one of the six routing algorithms implemented to compute quickest travel times. Only live data are used or necessary because it is assumed that a large enough number of vehicles are transmitting and updating the central server regularly. FreeSim has been tested on Los Angeles' freeways using user-generated data and live data from the California Department of Transportation. We found FreeSim to be special because few systems allow for both macroscopic and microscopic modelling. Their main limitations are that they do not interpolate sparse sensor data, do not incorporate historical data and do not consider surface streets.


FreeSim: A Free Real-Time Freeway Traffic Simulator - Windows Internet Explorer
http://www.freewaysimulator.com/demo
File Edit View Favorites Tools Help
FreeSim: A Free Real-Time Freeway Traffic Simulator
Los Angeles Freeway System
Load Freeway System
Fastest Path
Shortest Path
Remove Paths
Update Speed
Load File
Execute File
Stop Execution

2 to 3 - Time to travel along 5S between 210 and 405 (2.41 miles) is 4 minutes 49 seconds at 30 mph
2 to 1 - Time to travel along 5N between 210 and edge of map (0.1 miles) is 0 minutes 6 seconds at 65 mph
2 to 7 - Time to travel along 210E between 5 and 118 (5.91 miles) is 7 minutes 53 seconds at 45 mph

Freeway System: Los Angeles

Start: 101_405 at 10
End: 5_710 at 29
Fastest Path Algorithm: APAP Update

Messages

Shortest Path
Path = **10 11 20 22 21 29**
Distance = **19.86 miles**
Time = **18 minutes 20 seconds**
Time at speed limit = **18 minutes 20 seconds**

Start: 5_210 at 2
End: 118_210 at 7
Speed: 45

File: Vehicle Demo Command File

```
# this line instantiates vehicle 20 at time 10 traveling from 3 to 21
70|20|10|3|5_405|21|5_10_60_101
# this line has vehicle 20 update edge 24 to 23 to 10 mph at time 100
72|20|100|24|5_110|23|5_10|10
# this line updates vehicle 20's fastest path at time 110
73|20|110
```

Messages

```
Time 100: Vehicle 20 updated edge between 24 on 5_110 and 23 on 5_10
Time 100: Vehicle 20 is on edge 3 on 5_405 going to 6 on 5_118 until 1(
Time 110: Vehicle 20 has requested an updated fastest path
Time 110: Vehicle 20 is on edge 3 on 5_405 going to 6 on 5_118 until 1(
Fastest path for vehicle 20 is |5_118|6|5_170|8|5_134|12|5_110|24|101_1
Time 120: Vehicle 20 is on edge 3 on 5_405 going to 6 on 5_118 until 1(
```

Simulation Time: 120

Figure 3–1: FreeSim Graphical User Interface

3.2 Corsim (2007)

Corsim [7, 8] is a microscopic traffic simulator sponsored by the United States Federal Highway Administration and consists primarily of two traffic models: Netsim for modelling surface streets and Fresim for modelling freeways. It can model control devices (e.g traffic signals, ramp metering etc.), vehicles and driver behaviour, and includes the ability to accommodate complicated road geometries. Unlike other systems, it has undergone extensive validation and has perhaps the largest pool of developers. However, as is generally the case with microscopic models, there is a limit on both the maximum number of sensors and the maximum size of the road network. Furthermore, it is unclear from the literature if they handle the interpolation problem or if they use any historical data.

3.3 Li (2006)

A method for short-term traffic forecasting using type-2 fuzzy logic is presented in [9]. The authors propose a rule-based system that generates a range of values that allows the user to gauge the uncertainty of traffic estimates where a large range is more unreliable than a small range. Historical and live data are used to construct the fuzzy rules. Their approach is interesting because there is no fitting process and, as a result, no risk of over-fitting the model. For validation, the authors collected data for a 7-mile freeway stretch of Interstate 880 in Alameda County, California. A total of 24 data sets was obtained where each data set corresponds to one day from 5AM-10AM and 2PM-8PM. A total of 30 runs were performed and each run consisted of randomly selecting a test data set and randomly selecting 15 of the remaining data sets in order to construct the fuzzy rules. Their system's main strength is in the handling of uncertainty but the downside is that there remains much work to be done in terms of scaling it to 1) work for an entire city and 2) handle months if not years of data.

3.4 Chrobok (2001)

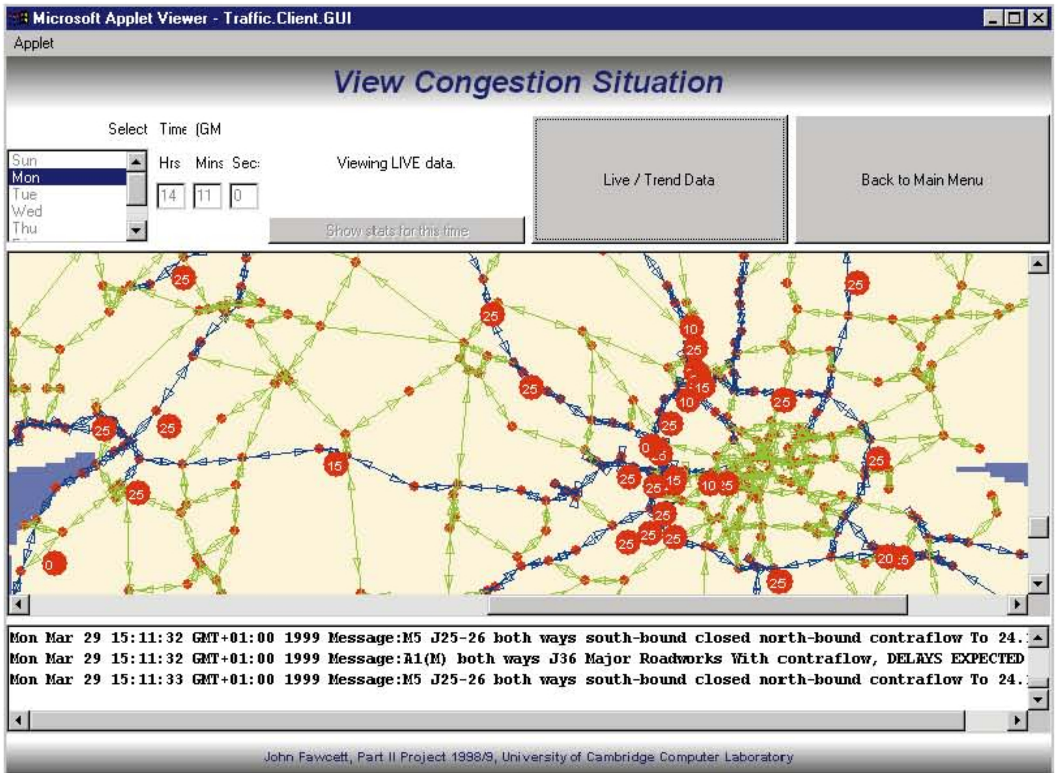
A variant of the classical cellular automaton model [10] (2001) is presented in [11] for modelling freeways. It is a microscopic traffic model that divides roads into cells such that each cell may contain no more than one vehicle.

Vehicles are modelled according to rules while lane changes are a function of the speed of the vehicle ahead and on the number of empty cells around. For validation, the authors used three years of inductive loop data of the North Rhine-Westphalia freeway network for which they simulated 14,000,000 cells on 2,500 kilometres of freeway. The system uses both historical and live inductive loop counts. When the historical data indicates insufficient vehicle counts on a given segment, it fills up the available (empty) cells in the vicinity of the inductive loop with vehicles until the number of counts matches the number of counts in the historical data or until there are no available cells left. Unlike other systems mentioned in this literature review, their model can extrapolate estimates into areas with little or no sensors and considers 4 classes of traffic patterns: Monday-Thursday (except holidays and days before holidays), Friday and days before holidays, Saturday except holidays, and Sunday and holidays. Their main drawback is their microscopic model that limits their scope to only freeways.

3.5 Adaptive Routing (2000)

Adaptive Routing [12] shown in Fig. 3–2 is the only system in this literature review that integrates both historical and live road congestion information for both freeways and surface streets. Their system is centred around estimating the travel speed of every road segment using a geometrically weighted average according to the time of day. This is accomplished by first discretizing a 24-hour day into 15 minute intervals and then computing, for each time interval, a running average using live speeds. When no new data are available due to the sparseness of the sensor data, they generate synthetic speed data in order to prevent the average speed from decreasing monotonically. They explore many different routing algorithms and, for validation, use the Trafficmaster [13] service which provides them with road speed data of the United

Kingdom. Their system is perhaps the most similar to the one we propose because they also average speed data. However we take their design to the next level by interpolating the sparse sensor data rather than generating synthetic data.



```
Mon Mar 29 15:11:32 GMT+01:00 1999 Message:M5 J25-26 both ways south-bound closed north-bound contraflow To 24.
Mon Mar 29 15:11:32 GMT+01:00 1999 Message:A1(M) both ways J36 Major Roadworks With contraflow, DELAYS EXPECTED
Mon Mar 29 15:11:33 GMT+01:00 1999 Message:M5 J25-26 both ways south-bound closed north-bound contraflow To 24.
```

John Fawcett, Part II Project 1998/9, University of Cambridge Computer Laboratory

Figure 3–2: Adaptive Routing Graphical User Interface

3.6 Dicaf (1998)

Dicaf [14] is designed to model the traffic flow of freeways. Unlike the other systems in this literature review, they alone employ a distributed scalable architecture. Contrary to the conventional central traffic server design, the authors propose installing many inexpensive distributed traffic management centres (DMTCs). Each DMTC is responsible for collecting and processing congestion data in its region, computing a congestion measure based on density and average speed, and sending this value to neighbouring DMTCs who in turn send it to their neighbours until every DMTC has been updated with the latest congestion information. The coverage area of each DMTC is determined by its processing power which is dependent not on a coverage distance but on the number of vehicles and the number of segments that it must monitor. When a vehicle needs to find the fastest path to its destination, its on board computer communicates with the nearest available DMTC, obtains congestion measures (CMs) for all the relevant segments and computes the quickest route. Because it takes a while for new CMs to transmit through the entire network, the most accurate CMs are the ones being controlled by the same DMTC and the accuracy drops off progressively as more and more distant CMs are accessed. As a result, the quickest route may change several times as the vehicle approaches its destination and accesses more up-to-date CMs. The advantage of their system is that each DMTC is simple and inexpensive, the overall system will continue to work even though individual DMTCs break down, and most importantly it scales well with increasing number of vehicles and road segments. Dicaf was tested in a virtual environment composed of 65 computers where the authors introduced a maximum of 45,000 vehicles and ran 400 simulations. Their system's main attraction is in the idea of a scalable, robust and fast architecture although it is currently not a real-world operational system. Furthermore, it is unclear from the literature whether or not the authors interpolate the sparse sensor data in computing the CMs of each DMTC.

3.7 Other works

There are other systems not covered in the preceding literature review but worth mentioning. If interested, the reader is invited to refer to the following list of related works:

- Vatsim [15, 16] - A vehicle and traffic simulator that facilitates developing intelligent transportation system control strategies involving multiple vehicles.
- Mitsim [17] - A microscopic traffic simulator for the evaluation of dynamic traffic management systems.
- Transims [18] - A transportation analysis and simulation system that helps transportation planners or engineers develop roadways by trying to predict displacements of individual households and travelers.
- Paramics [19] - A parallel microscopic traffic simulator that aims to provide road network planners with a range of tools.
- Vissim [20] - A simulation tool for the design of traffic actuated control systems.

Each system presented above has its strengths and weaknesses. Unfortunately, too few attempt to interpolate sparse traffic data. Given the current limited availability of road congestion data and the numerous sensor modalities that exist, we consider the ability of a traffic system to interpolate sparse and arbitrary traffic data to be of utmost importance. In the next chapter, we describe a system whose strength lies in its ability to interpolate sparse traffic measurements coming from any number of sensors or sensor modalities.

CHAPTER 4
Proposed Solution

In the literature, microscopic models are mostly used because they are better able to handle complicated road geometries and features such as traffic lights or on-ramp metering [7]. However, a macroscopic traffic model has the advantage when it comes to large scale designs thanks to the reduced number of variables [21]. Because we hope to estimate potential ambulance speeds for every road segment in a city, we use a macroscopic model to keep the large scale implementation as simple as possible. As such, the system we propose can handle the entire city of Ottawa including both freeways and surface streets, is not limited by the number of sensors and is updated every 5 to 6 minutes which is acceptable for the needs of the ambulance dispatchers.

Before delving into the innards of the traffic server, we offer the following high level description as depicted in Fig. 4–1. Given a road network, the traffic server takes whatever road data is available and outputs a potential ambulance speed for every road segment in the network. To accomplish this task, we begin by representing the road network as a directed graph where road segments in the road network correspond to nodes in the directed graph. Then we turn to almost a year's worth of historical sensor data to initialize the graph according the current or simulation date and time. Nodes with noisy or sparse historical data in time are initialized with less confidence. Conversely, nodes with abundant historical samples and less variance are initialized with higher confidence. While nodes without any historical data are initialized to a neutral state. Next, we turn to the live sensor data and the user entered data to complete the initialization of the graph. Specifically, if a node has access to both historical and live data, then the node is initialized by means of a weighted average between the two with more weight given to the live data. Additionally, the confidence of the initialization is also higher assuming that the live data are more reliable than the historical data. If a node has been manually initialized by the user, then whatever historical or live data is typically ignored and the confidence of the initialization is maximal unless specified otherwise by the user. Finally, we use a belief propagation algorithm to interpolate the sparse data and obtain dense locally consistent road speed estimates.

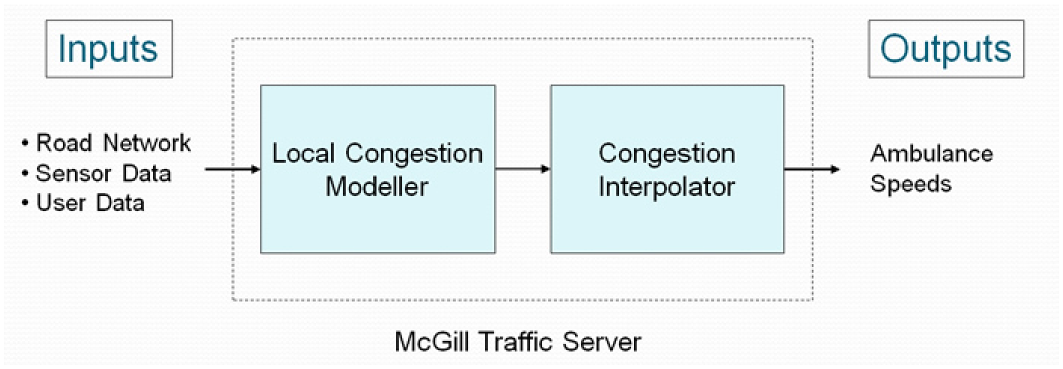


Figure 4–1: Simplified Block Diagram

4.1 Represent the Road Network as a Directed Graph

The first step is to represent the road network as a graph representation that will serve as the basis for subsequent computations. The traffic server accepts any road network database as long as it is given in the widely used geospatial vector data Environmental Systems Research Institute (ERSI) shapefile format. To extract the street geometry and attributes, we use the Shapefile C Library available online [22]. As indicated by their name, shapefiles contain shapes such as roads, rivers, power lines and railroads. Each shape has a set of vertices which are typically connected to form line segments. In general, a shape starts or stops at an intersection. As such, a long boulevard

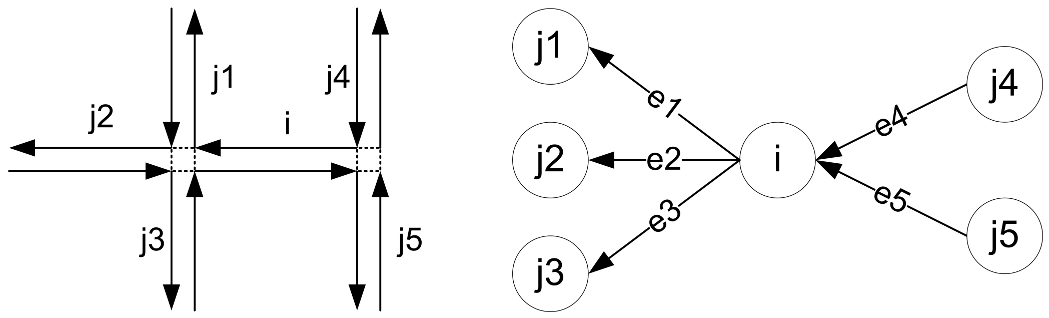


Figure 4–2: Simple Road Network (left) and Corresponding Graph Representation from the Perspective of Node i (right). Nodes j^1 through j^5 are neighbours of i and e^1 through e^5 are the corresponding edges.

may consist of many shapes if it spans many intersections. Each shape also has any number of unique attributes. 7 attributes are currently used and are described below.

1 TYPE: Attribute indicating the type of the shape. Possible types are roads, rivers, power lines and railroads.
2 ONEWAY: Attribute indicating the direction of the road assuming that the shape is of type road. If the attribute is empty or null, then the shape is bidirectional otherwise the shape is unidirectional. If the attribute contains "FT", then the shape originates from the road whose name is indicated in the "FROM_STREET" attribute and points to the road whose name is indicated in the "TO_STREET" attribute. Conversely, if the attribute contains "TF", then the shape originates from the road whose name is indicated in the "TO_STREET" and points to the road whose name is indicated in the "FROM_STREET" attribute.
3 FROM_STREET: Assuming the shape is a road, this attribute contains a road name and is only used if the "ONEWAY" attribute is "FT" or "TF".
4 TO_STREET: Assuming the shape is a road, this attribute contains a road name and is only used if the "ONEWAY" attribute is "FT" or "TF".
5 LANES: Assuming the shape is a road, this attribute contains the road's number of lanes.
6 ROAD_NAME: Assuming the shape is a road, this attribute contains the road's name.
7 SPD_LIMIT: Assuming the shape is a road, this attribute contains the road's speed limit.

We use the Boost Graph C++ library available online [23] to represent the road network as a graph composed of nodes and edges, described next, as in Fig. 4–2.

4.1.1 Nodes

Each node in the graph corresponds to a single road segment with a unique direction. As such, a two-way street gives rise to two nodes or road segments whose directions are 180° apart while a one-way street results in a single road segment. In Fig. 4–2, nodes j^1 through j^5 are considered to be neighbours of node i. To indicate the direction of vehicle flow, we may add a prefix 'in' or 'out' as qualifiers and refer to neighbours as in-neighbours or out-neighbours. In Fig. 4–2, nodes j^1 through j^3 are considered to be out-neighbours of node i while nodes j^4 and j^5 are considered to be its in-neighbours.

4.1.2 Edges

Nodes are linked to each other via edges. Since we have a directed graph, each edge has a particular orientation. To indicate the origin and destination of an edge, we refer to the originating node as a source and the destination node as a target. In Fig. 4–2, node i is the source of edges e^1 through e^3 and is also the target of edges e^4 through e^5. To indicate the position of an edge relative to a node i, we may add a prefix 'in' or 'out' as qualifiers and refer to edges as in-edges or out-edges of i. In Fig. 4–2, node i has 3 out-edges, e^1 through e^3, and 2 in-edges, e^4 through e^5.

Each edge has a weight bound between 0 and 1 that defines the likelihood of vehicle trajectories. In Fig. 4–2, we may refer to the edge weight of e^1 as one of two interchangeable notations: EW_{e^1} or $\mathrm{EW}_{i \to j^1}$, where i is the source of e^1 and j^1 its target.

In principle, edge weights should vary according to the time of day and the date, and be computed using historical and live turn ratios. As we do not have access to such information, we compute static edge weights using a generalized assumption about the behaviour of vehicles at each intersection. Explicitly, we assume that vehicles like to travel in the path with the least resistance or, conversely, the path with the most conductivity, where the conductivity of an out-edge is a function of the deviation angle, the number of lanes and the speed limit between a node and its out-neighbours. The out-edge with the greatest conductivity has the smallest deviation angle, the target road segment with the greatest number of lanes and highest speed limit. Note that this assumption does not always hold because, at times, on-ramps or off-ramps are favoured over main arteries, but doing so allows us to form a baseline estimate that will serve to evaluate the model's performance as more data is eventually included. Note that the static edge weights are generally only computed once during the initialization process unless the road network is altered due, for instance, to a road closure in which case they must be re-computed.

Numerically, if a node i has out-edges $\{e^1, \ldots, e^n\}$ whose targets are out-neighbours $\{j^1_{\text{out}}, \ldots, j^n_{\text{out}}\}$ respectively, then the edge weight of e^k, EW_{e^k}, is defined as:

$$\text{EW}_{e^k} = \frac{\text{Conductivity}_{e^k}}{\sum_{m=1}^{n} \text{Conductivity}_{e^m}}. \tag{4.1}$$

where $\text{Conductivity}_{e^k}$, the conductivity of edge e^k, is

$$\text{Conductivity}_{e^k} = \text{RCap}_{j^k_{\text{out}}} \cdot e^{-\beta \cdot \text{Deviation Angle}_{e^k}}, \tag{4.2}$$

$\text{RCap}_{j^k_{\text{out}}}$, the road segment capacity of j^k_{out}, is

$$\text{RCap}_{j^k_{\text{out}}} = \text{Number of Lanes}_{j^k_{\text{out}}} \cdot (\text{Speed Limit}_{j^k_{\text{out}}} + 10) \tag{4.3}$$

and β, the decay term, is

$$\beta = \frac{-\ln 0.1}{90}. \tag{4.4}$$

Note that (4.3) is a virtual road segment capacity and not the actual road segment capacity since the latter is not known. Note also that the Deviation Angle_e is the angle formed between road segment i and the target of e and that β is a constant obtained empirically such that a greater bend in the road results in a smaller conductivity value. In this case, we make it so that a deviation angle of 90 degrees results in a conductivity of 10% of the road capacity. Note also that the "+ 10" in the calculation of Road Capacity$_t$ is due to the nominal travel speed in Canada which we decided, based on personal driving experience, is often 10 to 20 km/h above the posted speed limit. Also, if a node has only one out-edge, then we assume that vehicles have only one trajectory choice in which case the sole out-edge has a weight of 1. If a node has E out-edges, where E is often greater than 1 in a typical road network, then the sum of the node's out-edge weights should equal 1 such that $\sum_{k=1}^{E} \mathrm{EW}_k = 1$. An exception to this rule would be in the case of a vehicle sink, for instance due to a stadium parking lot, in which case the sum may be allowed to be less than 1. Out-edges whose targets are dead ends have a weight of 0 because we assume that there is little or no relationship between road congestion and dead end streets. That is, the amount of vehicles entering a dead end road segment is too few to impact the congestion of neighbouring roads.

4.2 Estimate the Local Congestion at Each Node

The second step is to determine the most likely travel speed for each road segment or node based on the available sensor and user data. To accomplish this task, we construct a speed profile for each road segment which is essentially a trend in ambulance speeds over time. Note that the traffic server can make use of any sensor data as long as the data can be geo-referenced to a particular location in the road network and can be expressed in terms of a relative congestion term, described below.

4.2.1 Register the Data

The only road congestion data that we currently consider in this work is the AVL data, graciously provided by OPS. Specifically, we have access to almost a year's worth (February 1 2007 to December 20 2007) of historical ambulance AVL data which results in approximately 18 million records in the form of the following comma separated values:

```
4150,2/1/2007 12:00:25 AM,45.37018,-75.7697,
4261,2/1/2007 12:00:26 AM,45.0977,-75.2085,
4388,2/1/2007 12:00:27 AM,45.41693,-75.68886,
4172,2/1/2007 12:00:29 AM,45.37,-75.68277,
4409,2/1/2007 12:00:30 AM,45.56376,-74.62932,
4242,2/1/2007 12:00:32 AM,45.2902,-75.2289,
4347,2/1/2007 12:00:35 AM,45.37945,-75.65529,
4261,2/1/2007 12:00:37 AM,45.0988,-75.2057,
4285,2/1/2007 12:00:39 AM,45.38595,-75.68026,
4409,2/1/2007 12:00:41 AM,45.56128,-74.63054,
4388,2/1/2007 12:00:39 AM,45.41708,-75.68887,
4242,2/1/2007 12:00:43 AM,45.2895,-75.2303,
4347,2/1/2007 12:00:46 AM,45.37967,-75.65216,
...
```

Each line corresponds to a single record indicating the ambulance's id, the measurement date and time, and the GPS latitude and longitude coordinates.

Since the AVL data only gives the positions of ambulances every 10 to 12 seconds on average, we use 3 consecutive positions in time to reliably estimate the ambulance's direction and register it to a specific road segment. Explicitly, if we know where the ambulance was at time t_{t-1} and where it will be at time t_{t+1}, then we can determine with a high degree of confidence on which road segment the ambulance is travelling at time t_t. We estimate the ambulance travel speed using a straight line approximation by computing the distance travelled and dividing this value by the time interval. For time intervals on the order of 10 to 12 seconds, the straight line approximation provides a satisfactory baseline though there is room for improvement by using a more sophisticated road following algorithm. Any invalid speeds due to GPS tracking error are filtered in this process such that, from the original 18 million raw AVL samples, we obtain approximately 13 million useful ambulance speed samples that we refer to as the *processed* AVL data.

Unfortunately, the AVL data does not distinguish deployment speeds with lights or sirens on from redeployment speeds with lights and sirens off. Hopefully, the information will eventually be made available so that we can consider the two classes of data. For now, we simply assume that all AVL speeds are redeployment speeds.

4.2.2 Construct Historical Trends for Each Node

Before analyzing trends in ambulance speeds, we must segment the data. Currently, we split the processed AVL data into two classes, weekdays and weekends, so that the model can be trained to give appropriate speed estimates given the day of the week and the time of day. In the future more sophisticated data clustering could be used to improve the estimates such as splitting the data according to weather, holidays or special events such as festivals.

Next we construct two speed profiles (one for each data class) and investigate two different methods for constructing the trend in ambulance speeds. The first method uses linear regression to estimate the coefficients of a polynomial fit to the sample data,

$$\text{Speed}(t) = w_0 t^0 + w_1 t^1 + w_2 t^2 + ... + w_p t^p. \tag{4.5}$$

The second method, bin averaging, does not attempt to fit a curve to the data. Rather, we divide a day into 15 minute bins and simply take the mean of each bin to obtain the local speed estimate. Fig. 4–3 shows the weekday speed profiles of 4 different road segments with abundant AVL samples using both methods.

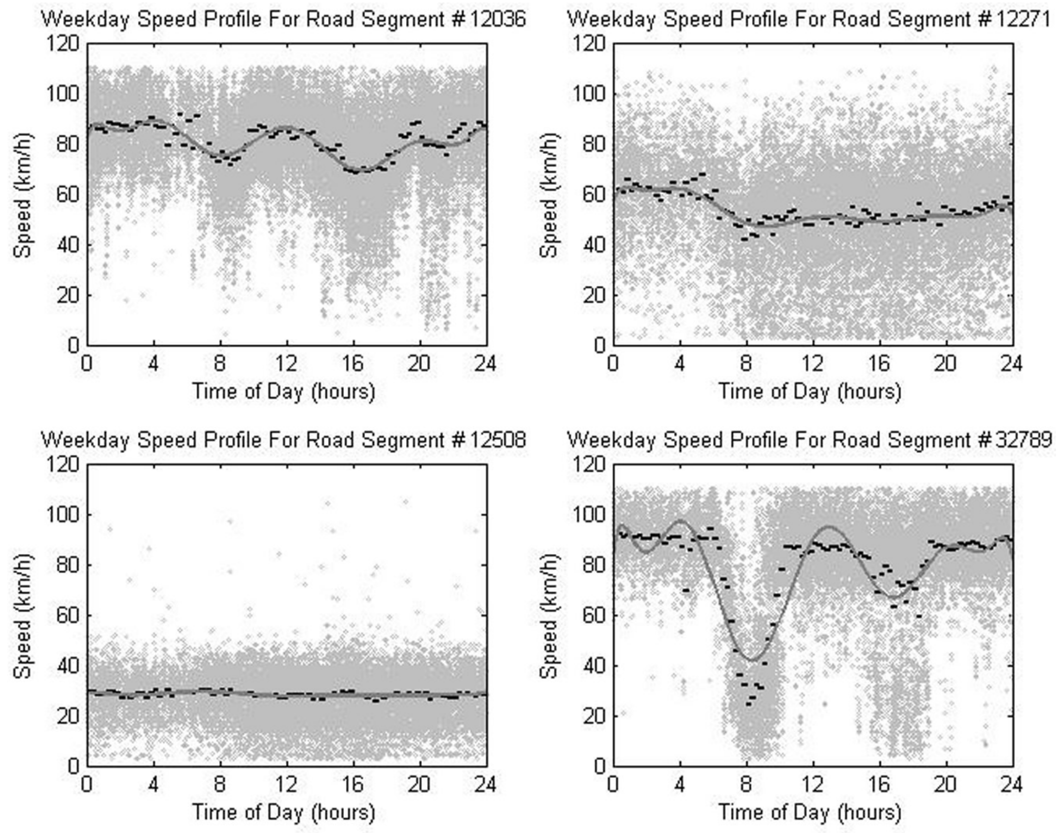


Figure 4–3: Weekday Speed Profiles for 4 Different Road Segments With Many Samples. Light grey points show the processed AVL speed samples. The black dashed lines is the fit due to bin averaging while the dark grey continuous line is the polynomial fit.

4.2.3 Express Measurements in Terms of Relative Congestion

Armed now with speed profiles, we address the problem of merging different sensors by converting arbitrary sensor measurements to a relative congestion term bound between 0 and 1 using an equation that may be unique for each sensor. For the AVL data, we use

$$\text{Relative Congestion}(t) = 1 - \frac{\text{Speed}(t)}{\text{Speed Limit}}, \tag{4.6}$$

but for other sensors such as inductive loops the Relative Congestion(t) might be obtained directly from counts rather than from vehicle speeds.

4.2.4 Determine the Local Congestion of Each Node

Typically, the local congestion of each node is dependent on the available sensor data. If a node has access to no sensor data, then its local congestion is unknown. If there is only historical sensor but no live data, then we obtain the local congestion by referring to the node's speed profile. If there is both historical and live sensor data, then the local congestion is a weighted average of the two relative congestions with less weight given to the more uncertain of the two. Note that the live AVL feed is not yet available, but that it will not be difficult to accommodate in terms of implementation because it only involves choosing a weight.

Note also that the relative congestions that serve as local estimates are sparse and noisy. Since ambulances drive over some roads more often than others, some road segments may have tens of thousands of AVL samples while others have none. Indeed, as can be seen in Fig. 4–3, there is a great deal of variance in the AVL data. Both the sparsity and the variance of the data contribute to the likelihood that relative congestions of neighbouring road segments are locally inconsistent.

4.2.5 Handle User Overrides

Although the traffic server is designed to require as little user data or intervention as possible, it may be necessary for the user to input changes when the current state starts to differ greatly from the historical data. Fortunately, the traffic server is locally adaptable and two types of user overrides are currently supported.

We refer to the first type as road closures. As such, a user may close off entire road segments such that affected vehicles are forced to find alternate routes along neighbouring roads. Road work, festivals or parades are examples of typical road closures. The effect on the model is that edge weights must be updated to account for the change in vehicle trajectory likelihoods. Note however that any change to linkages in the road network puts into question the usefulness of the historical data. Minor road closures that affect few vehicles have little or no effect on the model whereas major road closures that affect many vehicles should be accompanied by speed overrides, explained next, in the vicinity of the road closure to compensate for invalid historical data.

We refer to the second type as speed overrides. As such, a user may manually set a fixed local speed for a given road segment instead of using the one from its speed profile. Perhaps an accident or a stalled vehicle has caused a major traffic jam. By default, we assume that the user entered speed has 100% certainty though it is possible to specify a lower certainty value and compute a weighted average with sensor data, if available.

4.3 Interpolate Local Congestions

The last step is to interpolate the sparse local estimates and resolve any local inconsistencies that may arise from the previous step. To accomplish this interpolation task, we use a belief propagation network implemented using a Relaxation Labelling algorithm (RL) [24]. The underlying goal is to use the road network topology (number of lanes, speeds limits and linkages) to determine how likely each road segment is congested based on how likely its neighbouring road segments are congested. We chose the RL algorithm for interpolation because it is relatively simple to implement and offers a baseline to which we can compare eventual more complex strategies. Furthermore, it allows us to split a complex computation into many simple and parallel computations, uses context to compensate for noise and ambiguity, and is deterministic.

4.3.1 Setting Up the Relaxation Labelling Problem

Before describing the algorithm, we begin by detailing the basic RL components as applied to the road network.

Labels

In RL, each node has a set of m labels $\{\lambda^1, \ldots, \lambda^m\}$ where each label corresponds to a range of relative congestion. For example, if m is 5, then labels $\{\lambda^1, \lambda^2, \lambda^3, \lambda^4, \lambda^5\}$ correspond to relative congestions {[0-0.2], (0.2-0.4], (0.4-0.6], (0.6-0.8], (0.8-1.0]}, respectively. In turn, each label has a weight bound between 0 and 1 that defines the likelihood of the label. The label with the greatest weight indicates the most likely relative congestion. The notation we use is $p_i(\lambda^k)$ which corresponds to the weight of node i's label λ^k. Note that $p_i(\lambda^k)$ corresponds to probability measures, but are closer to normalized likelihoods. As such, the sum of each node's weights should be 1 such that $\sum_{k=1}^{m} p_i(\lambda^k) = 1$. Initial label weights depend on the historical and live speed data and are assigned at the beginning of the RL algorithm's iterative process. During the RL iterations, the label weights are incrementally changed until convergence is reached. According to [24], convergence is extremely likely as long as we use a "proper updating rule and reasonable compatibility values".

The Compatibility Matrix

A compatibility matrix contains elements that define the relationship between labels of neighbouring nodes. For example, if each node in the network has m labels, λ is the label of node i, λ' is the label of node j, and i and j are neighbours, then $r_{ij}(\lambda, \lambda')$ is an element in an $m \times m$ compatibility matrix that defines the relationship between the labels of i and the labels of its neighbour j. If the two labels greatly support each other, then $r_{ij}(\lambda, \lambda')$ should be large and positive. If the labels greatly inhibit each other, then $r_{ij}(\lambda, \lambda')$ should be large and negative. Otherwise, if there is no relation between the two labels, then $r_{ij}(\lambda, \lambda')$ should be 0.

In practice, node i with label λ has multiple neighbours j^1, j^2, $\ldots$, j^n with labels λ^1, λ^2, $\ldots$, λ^n, respectively, where the number of neighbours is typically 2-6. In this case, the compatibility matrix is a multidimensional array with elements $r_{ij^1 \cdots j^n}(\lambda, \lambda^1, \cdots, \lambda^n)$. The same rules as before apply. Namely, if labels $\lambda, \lambda^1, \ldots, \lambda^n$ greatly support each other, then $r_{ij^1 \cdots j^n}(\lambda, \lambda^1, \cdots, \lambda^n)$ should be large and positive. If the labels greatly inhibit each other, then $r_{ij^1 \cdots j^n}(\lambda, \lambda^1, \cdots, \lambda^n)$ should be large and negative. If there is no relation between the labels, then $r_{ij^1 \cdots j^n}(\lambda, \lambda^1, \cdots, \lambda^n)$ is 0. Note also that because few intersections in a road network are exactly alike, it is necessary to construct a unique compatibility matrix for each road segment in the road network.

There are several possible ways of computing $r_{ij^1 j^2 \cdots j^n}(\lambda, \lambda^1, \lambda^2, \cdots, \lambda^n)$ numerically. If sufficient data were acquired, then we could build the compatibility matrices by analyzing the congestion patterns at different times of the day though such a scenario would also require that we shorten the title of this thesis to "Obtaining Road Speed Estimates From GPS Measurements". Instead, we use a vehicle flow model (VFM) based on Kirchhoff's Current Law (KCL) to estimate $r_{ij^1 j^2 \cdots j^n}(\lambda, \lambda^1, \lambda^2, \cdots, \lambda^n)$ using a compatibility function and whatever data are available. The VFM can be read as follows: "The sum of vehicles entering a road segment is equal to the sum of vehicles exiting that road segment". Note that the VFM can cope with minor losses or additions of vehicles which is to be expected given that vehicles unlike electrons have a tendency to park or un-park. But when there are significant losses or additions a user must compensate by manually inputting a vehicle source or a sink into the road network. The compatibility function, on the other hand, generates a compatibility value based on the vehicle counts of neighbouring road segments. In fact, we use two compatibility functions to generate the desired compatibility values because, as will be shown shortly, road segments are affected by the vehicle counts of their in-neighbours differently than by the vehicle counts of their out-neighbours.

To be able to make use of the VFM, we need to express labels, which correspond to relative congestions, in terms of vehicle counts to be summed at the input and output of road segments. To accomplish this task, we assume that the relationship between road congestion and vehicles counts is linear. We recognize that this relationship needs to be refined but it will also serve as a baseline for comparison to future improvements. If a node i has relative congestion, RCon_i and road capacity, RCap_i, then we express the vehicle counts of i, VC_i, as follows:

$$\mathrm{VC}_i = \mathrm{RCon}_i \times \mathrm{RCap}_i. \tag{4.7}$$

Note that if the relative congestion is at its highest (i.e. 1), then we obtain the maximum number of vehicles on any road segment which is the road segment's capacity as was defined in (4.3). Note that these are not actual vehicle counts but are in fact vehicle-like units that we may refer to as virtual vehicle counts or simply vehicle counts for convenience. The VFM that we propose will hold as long as do not mix actual with virtual vehicle counts. If we wish to express units in terms of actual vehicle counts, then we must replace the road segment capacity in (4.3) by the actual maximum number of vehicles for each road segment.

The two compatibility functions are obtained as follows. If node i with label λ has in-neighbours $j_{\text{in}}^1 \dots j_{\text{in}}^n$ with labels $\lambda_{\text{in}}^1 \dots \lambda_{\text{in}}^n$ respectively and out-neighbours $j_{\text{out}}^1 \dots j_{\text{out}}^n$ with labels $\lambda_{\text{out}}^1 \dots \lambda_{\text{out}}^n$ respectively, then the two compatibility functions are

$$r_{ij_{\text{in}}^1 \cdots j_{\text{in}}^n}(\lambda, \lambda_{\text{in}}^1 \dots \lambda_{\text{in}}^n) = f(|\text{EVC-in}_i - \text{VC}_i|) \tag{4.8}$$

and

$$r_{ij_{\text{out}}^1 \cdots j_{\text{out}}^n}(\lambda, \lambda_{\text{out}}^1 \dots \lambda_{\text{out}}^n) = f(|\text{EVC-out}_i - \text{VC}_i|) \tag{4.9}$$

where EVC-in_i and EVC-out_i are the expected vehicle counts from node i's in-neighbours and out-neighbours respectively, defined as

$$\text{EVC-in}_i = \sum_{k=1}^{n} \text{VC}_{j_{\text{in}}^k} \times \text{EW}_{j_{\text{in}}^k \to i} \tag{4.10}$$

and

$$\text{EVC-out}_i = \sum_{k=1}^{n} \text{VC}_{j_{\text{out}}^k} \times \frac{\text{RCap}_i \times \text{EW}_{i \to j_{\text{out}}^k}}{\max \text{EVC-in}_{j_{\text{out}}^k}}. \tag{4.11}$$

Note that $\max \text{EVC-in}_{j_{\text{out}}^k}$ is the maximum expected vehicle counts from node j_{out}^k's in-neighbours and occurs when the vehicle counts of its in-neighbours are equal to their road capacities. The function f takes as arguments the difference between the expected and observed vehicle counts, and can be any linear or non-linear function that monotonically decreases such as $f(x) = \alpha \cdot e^{-\beta \cdot x}$, $f(x) = m \cdot x + b$ where α, β and m are constants or an empirically derived lookup table (LUT). Finally, there is a boundary case that must be accounted for: when the expected vehicle counts are greater than the road segment's capacity, and the relative congestion of i is at its highest (i.e. 1). In such a case, $r_{ij^1_{\text{in}}\cdots j^n_{\text{in}}}(\lambda, \lambda^1_{\text{in}} \ldots \lambda^n_{\text{in}})$ and $r_{ij^1_{\text{out}}\cdots j^n_{\text{out}}}(\lambda, \lambda^1_{\text{out}} \ldots \lambda^n_{\text{out}})$ should increase rather than decrease as the difference $|\text{EVC-in}_i - \text{VC}_i|$ increases.

4.3.2 The Relaxation Labelling Algorithm

Detailed next are the 8 steps of the RL algorithm.

RL 1: Initialize the iteration counter $k = 0$ and the label weights $\bar{p}^k$ according to the current time and day, t_{curr}.

For a given road segment at t_{curr}, we initialize the label weights using a normal distribution $N(\mu, \sigma^2)$, where the mean μ is the relative congestion (computed in the Local Congestion Modeller Module). The variance σ^2 of the distribution is used to adjust the confidence of the local congestion. If a node has only historical data, then the variance of the distribution is a function of the variance and the number of historical samples. If there are many samples and the AVL variance is low, then we expect the label weights to have a more peaked distribution. On the other hand, if there are few samples or the AVL variance is high, then the label weights should have a flatter distribution. In the extreme case that the road segment has no historical data, then we initialize the label weights using a uniform distribution such that each label is equally possible. If live sensor data are available or a user has manually entered a speed override, then we initialize the label weights using a highly peaked distribution. By default, user entered data has a variance of zero such that the distribution resembles an impulse, but it is equally possible to specify a lower certainty value.

RL 2: Compute the support for every label of every node in the graph, $\bar{q}^k$.

The support for a given label of a given node is dependent on the label weights of neighbouring nodes and on the road network topology in the form of the two aforementioned compatibility functions. Consider the following simple example of calculating support before considering the more complex case. If each node has m labels such that node i with label λ has neighbour j with labels λ' and compatibility function $r_{ij}(\lambda, \lambda')$, then the support for λ can be expressed as

$$q_i(\lambda) = \sum_{\lambda'} r_{ij}(\lambda, \lambda') p_j(\lambda'). \tag{4.12}$$

Since node i can have multiple neighbours j^1, j^2, ... , j^n with labels λ^1, λ^2, ... , λ^n respectively, where n is often 2-6, then the equation for support becomes

$$\begin{aligned} q_i(\lambda) = &\sum_{\lambda^1} \sum_{\lambda^2} \cdots \sum_{\lambda^n} r_{ij^1 j^2 \cdots j^n}(\lambda, \lambda^1, \lambda^2, \cdots, \lambda^n) \cdot p_{j^1}(\lambda^1) \cdot \\ & p_{j^2}(\lambda^2) \cdot \ldots \cdot p_{j^n}(\lambda^n). \end{aligned} \tag{4.13}$$

Also, since we use the VFM to estimate the compatibilities from two compatibility functions because road segments are affected by the vehicle counts of the in-neighbours differently than by the vehicle counts of their out-neighbours, then the equation for support becomes

$$q_i(\lambda) = q\text{-in}_i(\lambda) \times q\text{-out}_i(\lambda), \tag{4.14}$$

where

$$\begin{aligned} q\text{-in}_i(\lambda) = &\sum_{\lambda^1_{\text{in}}} \sum_{\lambda^2_{\text{in}}} \cdots \sum_{\lambda^y_{\text{in}}} r_{ij^1_{\text{in}} j^2_{\text{in}} \cdots j^y_{\text{in}}}(\lambda, \lambda^1_{\text{in}}, \lambda^2_{\text{in}}, \cdots, \lambda^y_{\text{in}}) \cdot p_{j^1_{\text{in}}}(\lambda^1_{\text{in}}) \cdot \\ & p_{j^2_{\text{in}}}(\lambda^2_{\text{in}}) \cdot \ldots \cdot p_{j^y_{\text{in}}}(\lambda^y_{\text{in}}), \end{aligned} \tag{4.15}$$

$$\begin{aligned} q\text{-out}_i(\lambda) = &\sum_{\lambda^1_{\text{out}}} \sum_{\lambda^2_{\text{out}}} \cdots \sum_{\lambda^z_{\text{out}}} r_{ij^1_{\text{out}} j^2_{\text{out}} \cdots j^z_{\text{out}}}(\lambda, \lambda^1_{\text{out}}, \lambda^2_{\text{out}}, \cdots, \lambda^z_{\text{out}}) \cdot p_{j^1_{\text{out}}}(\lambda^1_{\text{out}}) \cdot \\ & p_{j^2_{\text{out}}}(\lambda^2_{\text{out}}) \cdot \ldots \cdot p_{j^z_{\text{out}}}(\lambda^z_{\text{out}}) \end{aligned} \tag{4.16}$$

and y, z are the number of in-neighbours and out-neighbours respectively.

RL 3: Compute the update direction for the label weights, $\bar{u}^k$.

It is shown in [24] that $\bar{u}$ is the normalized projection of the support vector, $\bar{q}$, onto the set of all tangent vectors at $\bar{p}$, $T_{\bar{p}}$, where $\bar{p}$ is in the interior of the assignment space. The algorithm to obtain $\bar{u}$ is originally described in [24] Appendix A and is reproduced in Algorithm 1 for convenience and completeness. The inputs to the algorithm are the label weights of every node, p, the support vector of every node, q, the number of nodes n and the number of labels m.

Algorithm 1 Projection Operator to Obtain $\bar{u}$

```
1: procedure PROJECTION OPERATOR($p, q, n, m$)
2:     for $i$ in $[1 \ldots n]$ do
3:         $D := \{k \text{ in } [1 \ldots m] \mid p_i(k) = 0\}$;    ▷ D is an array that contains (discarded) labels whose weights are 0. If none of the label weights are 0, then D remains empty.
4:         $S := \{\ \}$;    ▷ S determines the stopping criterion of the loop in the next step and is initialized to an empty array.
5:         loop    ▷ Loop executes at most $m + 1$ times and terminates when QUIT is executed.
6:             $N_s := \#S$;    ▷ Set $N_s$ to be the number of elements in array $S$.
7:             $t := \frac{1}{m - N_s} \sum_{\substack{k=1 \\ k \notin S}}^{m} q_i(k)$    ▷ $t$ is a threshold term.
8:             $S := \{k \text{ in } D \mid q_i(k) < t\}$;    ▷ $S$ contains the discarded labels in D whose support $q_i(k)$ is less than threshold $t$.
9:             if $\#S = N_s$ then
10:                QUIT    ▷ Break out of indefinite loop.
11:            end if
12:        end loop
13:        for $k$ in $[1 \ldots m]$ do
14:            $u_i(k) = \begin{cases} 0 & k \in S \\ q_i(k) - t & \text{otherwise} \end{cases}$    ▷ Compute vector $\bar{u}_i$.
15:        end for
16:        $\bar{u}_i = \frac{\bar{u}_i}{\|\bar{u}_i\|}$;    ▷ Normalize $\bar{u}_i$.
17:    end for
18: end procedure
```

RL 4: If $\bar{u}^k = \bar{0}$ such that the projection computed previously is zero then go to RL 8.

If the projection is zero then we have reached a consistent labelling and can conclude the RL process.

RL 5: Otherwise, update the label weights $\bar{p}^{k+1} = \bar{p}^{k} + h\bar{u}^{k}$, where $0 < h \leq \alpha_k$ and α_k is a small valued maximum step size that may decrease as k increases to speed convergence.

Each iteration moves the label weights $\bar{p}$ incrementally in the projection direction.

RL 6: If $k = k_{\mathbf{max}}$ such that the iteration counter has reached the maximum iteration count then go to RL 8

We observed consistent labellings after 8-10 iterations. When the maximum number of iterations is reached, then conclude the RL process.

RL 7: Otherwise, increment the iteration counter $k = k + 1$ and go back to RL 2.

Perform another RL iteration until a stop condition is reached.

RL 8: Obtain the final labelling and go back to RL 1 to resume the entire process for the new current date and time, $t_{\mathbf{curr}'}$.

When the RL process ends for t_{curr}, each node has a distribution of label weights that is locally consistent with its neighbouring nodes. The label with the highest weight corresponds to the most likely relative congestion interval. The centre of the interval is converted to a potential ambulance speed using

$$\text{Speed} = \text{Speed Limit} \cdot (1 - \text{Relative Congestion}) \qquad (4.17)$$

and the entire RL algorithm starts over again with the new date and time in order to keep the road network up-to-date with the latest speeds.

The traffic server outputs the most likely ambulance speed for every road segment in the road network at any given day and time. Unlike the other systems in the literature review, we do not implement any routing algorithm, the reason being simply that it was not a requirement of the traffic server module. For a complete system overview, we provide a detailed flowchart in Appendix A.

CHAPTER 5
Experimental Methodology and Results

5.1 K-Fold Cross-Validation

To validate our system, we employ a common technique known as K-fold cross-validation. First, we split the historical data randomly into k evenly sized parts called folds. Then we choose one of the folds to act as the test set and train the model on the remaining k-1 folds. We repeat this process k times using each fold only once as the test set and always training on the remaining folds. As such, each iteration randomly excludes a percentage of the data from the dataset for initialization of the graph and then uses this excluded data to test the resulting model. This way we can gauge the stability of the algorithm with respect to variations in the input data. Typically 5 folds are enough to conclude whether or not the model's performance is stable.

Due to the large number of data samples in the test set (approximately 13 million) and the time it takes to update the model (5 minutes per update), we ran the traffic server over a continuous span of 24 hours per fold per data class and to obtain the simulated results. Then each sample in the test set is matched to the nearest sample in the simulated results and the difference between the simulated ambulance speed and the test set's speed is used as the estimate error. Using this method of validation has the significant advantage that the simulation time is the same regardless of the size of the road network or the size of the test set. Using 5 folds ($k = 5$), it takes 10 days ($= 5$ folds $\times$ 2 data classes) of continuous processing time to validate the system.

Table 5–1: K-Fold Cross-Validation Simulation Results for Weekdays (top) and Weekends (bottom) using Polynomial Regression, 9 RL Iterations, 7 RL Labels. MAE: Mean Average Error; RMSE: Root Mean Squared Error; STD: Standard Deviation.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.75	12.75	12.76	12.75	12.78	12.76
RMSE (km/h)	17.06	17.05	17.07	17.05	17.09	17.06
STD (km/h)	11.34	11.33	11.34	11.32	11.34	11.33
MAE (labels)	1.26	1.26	1.27	1.27	1.27	1.27
RMSE (labels)	1.82	1.82	1.82	1.82	1.82	1.82
STD (labels)	1.31	1.31	1.31	1.31	1.31	1.31

Fold k	1	2	3	4	5	Mean
MAE (km/h)	13.12	13.13	13.11	13.12	13.14	13.12
RMSE (km/h)	17.96	17.97	17.97	17.96	17.98	17.97
STD (km/h)	12.26	12.28	12.28	12.26	12.27	12.27
MAE (labels)	1.29	1.29	1.29	1.29	1.29	1.29
RMSE (labels)	1.93	1.93	1.93	1.93	1.93	1.93
STD (labels)	1.43	1.43	1.43	1.43	1.43	1.43

5.2 Test Cases

We ran 4 test cases totalling 40 days of processing time that was distributed over multiple commodity laptops. We were interested in tweaking mainly two parameters: 1) the number of RL labels and 2) the choice of speed profile building between polynomial regression and bin averaging. In every test case below, we use 5 folds for cross-validation and express the error in terms of both labels and speed (kilometres per hour).

5.2.1 7 RL Labels and Polynomial Regression

Using 9 RL iterations, 7 RL labels and polynomial regression for constructing local speed profiles, we obtain the results shown in Table 5–1.

5.2.2 7 RL Labels and Bin Averaging

Using 9 RL iterations, 7 RL labels and bin averaging for constructing local speed profiles, we obtain the results shown in Table 5–2.

Table 5–2: K-Fold Cross-Validation Simulation Results for Weekdays (top) and Weekends (bottom) using Bin Averaging, 9 RL Iterations, 7 RL Labels. MAE: Mean Average Error; RMSE: Root Mean Squared Error; STD: Standard Deviation.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.18	12.17	12.18	12.18	12.19	12.18
RMSE (km/h)	16.25	16.24	16.26	16.24	16.26	16.25
STD (km/h)	10.76	10.75	10.76	10.75	10.76	10.76
MAE (labels)	1.22	1.22	1.22	1.22	1.22	1.22
RMSE (labels)	1.76	1.76	1.76	1.76	1.76	1.76
STD (labels)	1.27	1.27	1.27	1.27	1.27	1.27

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.24	12.25	12.22	12.23	12.27	12.24
RMSE (km/h)	16.69	16.69	16.66	16.67	16.71	16.68
STD (km/h)	11.35	11.35	11.32	11.33	11.35	11.34
MAE (labels)	1.23	1.23	1.22	1.22	1.23	1.23
RMSE (labels)	1.84	1.83	1.83	1.83	1.84	1.83
STD (labels)	1.37	1.37	1.36	1.36	1.37	1.37

5.2.3 9 RL Labels and Polynomial Regression

In the hopes of reducing our error, we tried relaxing the update time interval constraint from 5-6 minutes to 10-11 minutes in order to increase the number of RL labels from 7 to 9 thus reducing the coarseness or interval size of our estimates. Using 9 RL iterations, 9 RL labels and polynomial regression, we obtain the results shown in Table 5–3.

5.2.4 9 RL Labels and Bin Averaging

Finally, using 9 RL iterations, 9 RL labels and bin averaging, we obtain the results shown in Table 5–4.

5.3 Key Results and Observations

For computing local estimates, we notice that bin averaging consistently leads to better results compared to polynomial regression. The reason is not evident for roads with a great number of AVL samples as in Fig. 4–3 but, for

Table 5–3: K-Fold Cross-Validation Simulation Results for Weekdays (top) and Weekends (bottom) using Polynomial Regression, 9 RL Iterations, 9 RL Labels. MAE: Mean Average Error; RMSE: Root Mean Squared Error; STD: Standard Deviation.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.63	12.62	12.64	12.62	12.63	12.63
RMSE (km/h)	17.00	16.99	17.01	16.99	17.00	17.00
STD (km/h)	11.37	11.37	11.39	11.37	11.38	11.38
MAE (labels)	1.58	1.58	1.59	1.58	1.59	1.58
RMSE (labels)	2.30	2.30	2.30	2.30	2.30	2.30
STD (labels)	1.66	1.66	1.67	1.66	1.66	1.66

Fold k	1	2	3	4	5	Mean
MAE (km/h)	13.03	13.04	13.02	13.02	13.05	13.03
RMSE (km/h)	17.92	17.94	17.91	17.90	17.94	17.92
STD (km/h)	12.30	12.31	12.30	12.29	12.31	12.30
MAE (labels)	1.63	1.63	1.63	1.62	1.63	1.63
RMSE (labels)	2.44	2.44	2.44	2.44	2.44	2.44
STD (labels)	1.82	1.82	1.82	1.82	1.82	1.82

Table 5–4: K-Fold Cross-Validation Simulation Results for Weekdays (top) and Weekends (bottom) using Bin Averaging, 9 RL Iterations, 9 RL Labels. MAE: Mean Average Error; RMSE: Root Mean Squared Error; STD: Standard Deviation.

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.21	12.21	12.09	12.09	12.09	12.14
RMSE (km/h)	16.41	16.40	16.28	16.27	16.27	16.33
STD (km/h)	10.97	10.95	10.90	10.89	10.89	10.92
MAE (labels)	1.54	1.54	1.53	1.53	1.53	1.53
RMSE (labels)	2.24	2.24	2.22	2.22	2.22	2.23
STD (labels)	1.63	1.62	1.61	1.61	1.61	1.62

Fold k	1	2	3	4	5	Mean
MAE (km/h)	12.40	12.40	12.29	12.30	12.32	12.34
RMSE (km/h)	16.99	17.01	16.88	16.88	16.90	16.93
STD (km/h)	11.61	11.63	11.57	11.56	11.57	11.59
MAE (labels)	1.56	1.56	1.54	1.54	1.55	1.55
RMSE (labels)	2.34	2.35	2.32	2.33	2.33	2.33
STD (labels)	1.75	1.75	1.74	1.74	1.74	1.74

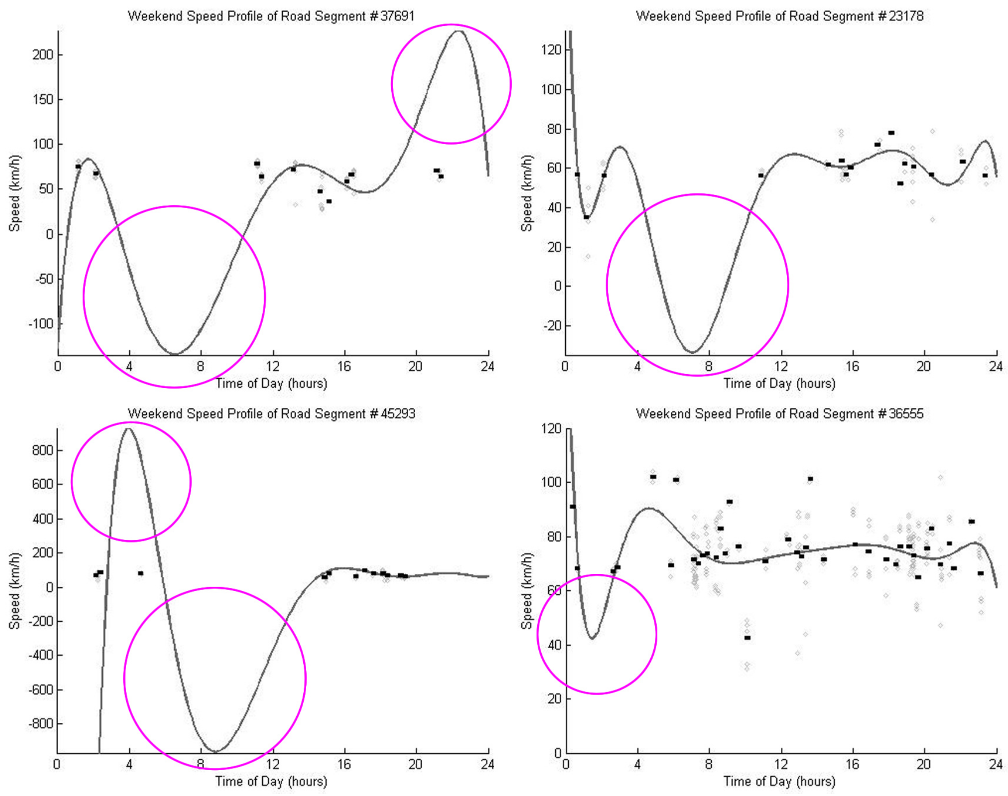


Figure 5–1: Weekday Speed Profiles for 4 Different Road Segments With Few Samples. Light grey points show the processed AVL speed samples. The black dashed lines is the fit due to bin averaging while the dark grey continuous line is the polynomial fit. Circled are the regions where polynomial regression does poorly.

roads with too few AVL samples as in Fig. 5–1, polynomial regression is too sensitive to outliers and results in a poor fit.

Increasing the number of labels also did not produce the expected improvements in the MAE or the root mean squared error (RSME). One possible explanation is the large variance in the AVL data as can be seen in Fig. 5–2. In which case, it is better to use 7 RL labels rather than 9 since it takes half the processing time.

The traffic server has a mean absolute errors (MAEs) of 12-13 km/h with a standard deviation (STD) of 10-12 km/h. We observe also that there is very little variation in the traffic server estimates across the different folds. Finally,

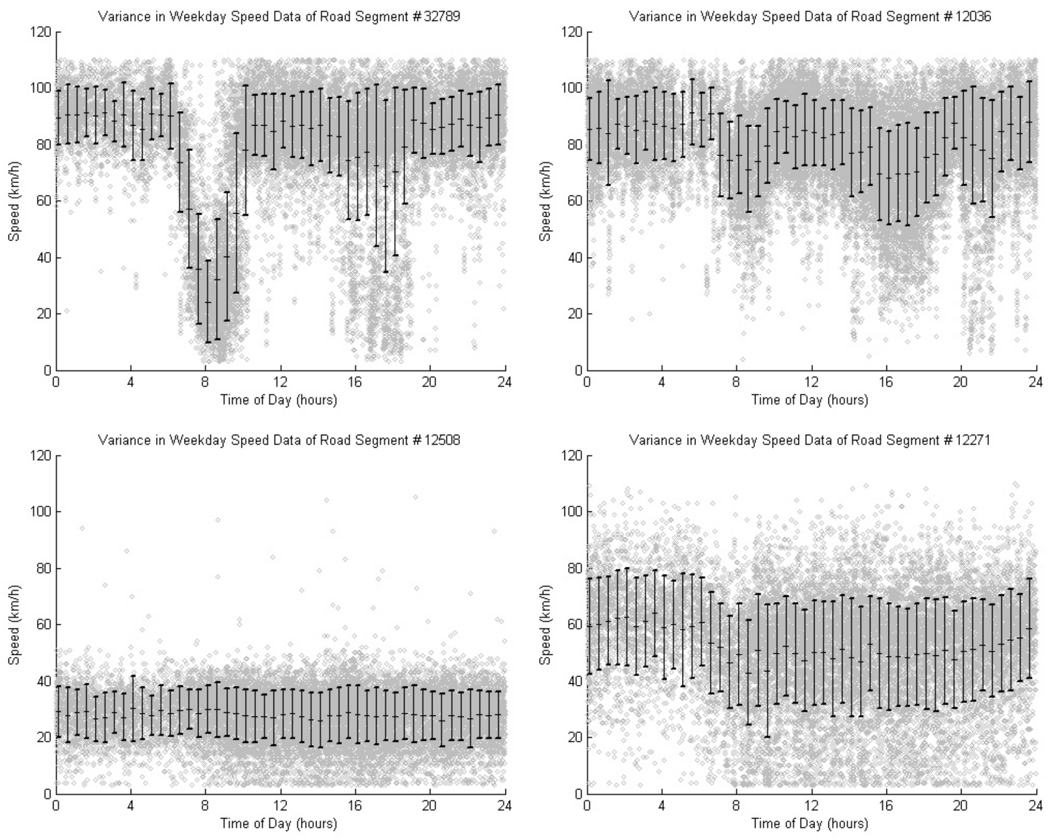


Figure 5–2: Standard Deviation of the Speed Data for 4 Different Road Segments. Light grey points show the processed AVL speed samples. Black vertical lines indicate the standard deviation σ of the data where the variance is defined as the standard deviation squared, σ^2.

we want to point out that in all the experimental results obtained, we used real AVL data samples and not synthetic data.

CHAPTER 6
Discussion

6.1 Validation Results

The mean absolute errors (MAEs) we obtained of 12-13 km/h may seem large but, at the same time, they are remarkably consistent. We offer several reasons to explain the magnitude of the MAEs. The first reason is the great deal of variance in the AVL data. Though we compensated by using the variance in the AVL data to initialize the distribution of label weights before iterating through the RL algorithm, it remains one of the biggest sources of error. Second is the inadequate road network database of Ottawa plagued with errors and lacking critical information (more on this below). CAE is actively working on obtaining a more complete and correct database, but in the meantime, the shortcomings in the road network database are a significant source of error. Lastly are the various assumptions that were made throughout this work. Namely, the conversion from ambulance GPS coordinates to ambulance speeds, the registration of ambulance speeds to road segment for speed profile building, the conversion from ambulance speeds to road congestion, the linear assumption made when estimating vehicle counts from road congestion, the computation of edge weights, the initialization of RL labels using a gaussian distribution based on the mean and variance of the data, the vehicle flow model for estimating compatibilities between labels, the RL algorithm itself for leading to an optimal real world solution, and the reverse process of going back to ambulance speeds from RL labels after the RL algorithm is complete. Anyone of these assumptions can contain errors which may propagate from the beginning to the end of the system.

Despite these assumptions, the results we obtained are great and can only get better using the current implementation as a baseline. Overall, given the sparse nature of the measurements and the coarse quantization of speed ranges, the results obtained were much better than expected - more than sufficient to estimate travel times between any two nodes in the network.

6.2 Sources and Sinks

Sources and sinks are special cases. A user may enter a source or a sink to account for a sudden and significant appearance or disappearance of vehicles. An example for both scenarios is a stadium parking lot before and after a big game. When hundreds if not thousands of drivers and their passengers arrive to park their vehicles, the user should input a sink. In general, the edge weights that leave a given road segment sum up to 1. However, in the case of a sink, the edge weights should sum up to a value less than 1. As such, the overwhelming number of vehicles will not propagate as strongly on to connecting road segments. At the end of the big game when it is time for the spectators to leave, the user may enter a source by inputting a low speed override for the affected road segments. We do not increase the edge weights to allow their sum to be greater than 1 because this may lead to unpredictable results during the RL algorithm.

6.3 Available Data Sources

For this project, we had access to a number of different data sources that we describe next.

6.3.1 Traffic Cameras

The original goal was to use image sequences from the existing traffic camera network, but this proved unfeasible for many reasons. Most notably is the lack of a uniform infrastructure for routing data to a central server. Also, lack of access to camera parameters such as pose and zoom makes the design of robust estimation procedures problematic.

6.3.2 Inductive Loops

The inductive loop count data was more promising because we had access to almost 200 loops spanning 5 years. Unfortunately, we had an issue with the registration of the sensor data for although we were given the nearest intersection and the road segment direction, we did not know in which lane or where along the road segment the loops were located. Because different lanes do not necessarily receive the same number of vehicle counts despite having approximately the same travel speed, there is a high probability that different loops of the same road segment contradict one another, especially if one of the loops is located in a turning lane. Another issue is the lack of ground truth that we would be required to convert vehicle counts into travel speeds. Without the exact loop positions and the ground truth, we have no way of reliably converting loop counts into average travel speeds unless we implement a microscopic traffic model which, as we explained at the beginning of Chapter Section 4, is an approach we want to avoid if possible.

6.3.3 AVL GPS Data

In the end, we used readily available AVL data which turned out to be a more practical solution. Unlike the inductive loops which are statically located and potentially contradictory, we do not need to know the exact location of the ambulance on the road segment because we assume that the ambulance is travelling on whatever lane is available and that the average speed of the ambulance for a given lane is the same for each lane. This technology can also be expanded to other fleets such as municipal vehicles or taxis to produce a denser initialization of the graph. Unfortunately, due to privacy concerns, we cannot make the data we used available to the research community for standardized testing.

6.4 Shapefile Issues

As mentioned before, the traffic server has a number of critical issues that CAE is currently working on resolving.

6.4.1 Linkages

The traffic server relies heavily on road linkage information which are currently inferred from the geometric properties within the shapefiles. Unfortunately, we observed a number of streets that did not connect where they were supposed to. The problem is that if there is a large enough drift - in some cases, we observed drifts ranging from 1 to 10 metres - then the intersections become difficult to discern and edge weights are computed erroneously. For display purposes, such drifts may be acceptable because they are barely perceptible when employing a bird's eye view of the city. But for the purposes of traffic modelling, this is unacceptable.

Supposing that the observed drift is unavoidable due to the inherent inaccuracies in the sensor, we propose that the road linkages be included in the shapefile attributes rather than be extracted from the shapefile geometric information. As such, each street or shape would contain the unique road ids of its neighbours. Though this may increase both the size of the shapefiles and the loading time, we do not consider this to be an issue because we only process the shapefiles once during the initialization of the system.

6.4.2 One-way Streets

Another shapefile attribute that is crucial to modelling vehicle flow is the one-way street information. Unfortunately, at the time of completing this work, we had to assume that all streets were bi-directional.

At one point, Ottawa's Fire Department did provide shapefiles with one-way information. But they lacked speed limits which were already available in the current shapefiles. Note that CAE tried to merge the two shapefiles but were unsuccessful because they differed too much.

6.4.3 Number of Lanes

The number of lanes is essential in computing the edge weights. Unfortunately, the way it is provided in the shapefiles is ambiguous. Indeed, the number of lanes of a given street is represented by a single attribute. When this attribute is odd, for instance 5, which direction has 2 lanes and which has 3? Or is it 1 lane in one direction and 4 in the other? The obvious solution is to use two attributes for number of lanes - one for each direction. In the mean time, we assume that each direction has half the number of lanes rounded downwards such that 5 lanes corresponds to 2 in each direction.

6.4.4 Speed Limits

Fortunately compared to the other shapefile attributes, the speed limit attribute is perhaps the most reliable and complete one though it is not without its shortcomings. For the 20,000 streets in Ottawa, approximately 1,000 streets are without a speed limit. For these streets, we simply assume a speed limit of 50 km/h and hope that we are correct for most of these streets. Eventually, we hope that it will no longer be necessary to make such an assumption, in which case, we expect the errors obtained during validation to decrease.

6.5 Scale and Performance

The traffic server was designed with the issue of scale in mind. Indeed, there is no limit on the number of AVL samples in the historical data since the heavy number crunching is done offline and the runtime calculations are neglible. On commodity hardware, the traffic server is able to update the state of the entire road network including both freeways and surface streets in

5-6 minute intervals using 7 RL labels and 10-11 minute intervals using 9 RL labels.

During runtime, the bulk of processing is dedicated to computing the support of each node's labels (step 2 of the RL algorithm). Depending on the number of in-edges and out-edges of each node in the graph, the processing time may increase or decrease. In the shapefiles of Ottawa we used, many streets are composed of multiple short shapes or segments. As a result, there is a large number of road segments with only one out-edge and in-edge. When we compute the support for these road segments with only one out-edge or in-edge, the computation is very quick. The lengthy calculations for support are due mostly to road segments with multiple out-edges or multiple in-edges.

To reduce the number of computations during runtime when a node has more than 3 in-edges, we only consider the 3 in-edges with the greatest number of vehicles and ignore the rest. Similarly, we impose the same constraint on the number of out-edges. In essence, we assume that few roads, if any, have more than 3 significant sources of vehicles at its input or at its output and we argue that the loss in model accuracy is negligible compared the the speedup.

The question that begs to be asked is: what happens if we double the size of the city or, conversely, if we half it? The complexity of the traffic server is on average $O(2n^2)$ where n is the number of road segments. Note that the multiplication by a factor of 2 is due to the two edge types (in- versus out-edges) in the calculation of support whereas the exponent of 2 refers to the average number of in- or out-edges per node. Accordingly, we observe that the performance of the system is closely tied with the size of the city. For larger cities, we can take advantage of the parallel nature of RL algorithm and split the processing over multiple processing cores using readily available multi-core processors in order to minimize processing time.

CHAPTER 7
Future Work

There remains much to do in terms of improving the accuracy of the traffic server.

7.1 Model Sources and Sinks

Perhaps the most effective and easiest improvement is to explicitly model sources and sinks by providing a source and sink edge incident on every node. Doing so would allow certain neighbouring nodes to remain locally inconsistent instead of erroneously interpolating road congestion across neighbours. Unlike electrons, large number of vehicles may suddenly park or appear in the flow at the onset or conclusion of a typical work day or of a special event such as a hockey game or concert.

7.2 Incorporate Statistics at Intersections

Perhaps the most important suggestion in terms of improving the results we obtained would be to incorporate statistics at intersections. Using intersection data such as turn ratios, we could compute dynamic edge weights that would vary according to the actual or simulation date and time.

7.3 Replace Linear Relationship Between Congestion and Number of Vehicles

Another improvement would be to replace the relationship between road congestion and number of vehicles by a non-linear one. The reason being that the assumption of linearity between the two that we make may be too far from reality and may consequently result in a poor performance of the VFM to estimate the compatibility matrix.

7.4 Incorporate Reserved Lanes

The estimates can be improved by including reserved lane information because intersecting regions where reserved lanes begin or rejoin normal lanes and during rush hour are problematic. The reason being that we expect ambulance speeds at these regions to be discontinuous during rush hour, when normal lanes are much more heavily congested compared to reserved lanes. In turn, the RL algorithm may erroneously smooth the ambulance speeds rather than let them be discontinuous. Shoulders, on the other hand, are not necessary because we are currently focused on redeployment rather than deployment. During redeployment, we assume that ambulance do not drive on the shoulder of the road.

7.5 Define Super Nodes

We have explained nodes and their place in the directed graph. Another concept is that of a super node which is an aggregate of nodes corresponding to portions of streets or neighbourhoods. Its main advantage lies in being more robust to noise since each super node contains a greater number of historical samples.

7.6 Determine an Optimal Compatibility Function

In this work, we proposed two possible compatibility functions. Although both led to satisfactory results after 8-10 iterations, it would be interesting to study and determine whether or not an optimal compatibility function exists that would lead to a more accurate or more locally consistent result requiring fewer iterations.

7.7 Segment the Data

Currently, we segment the data into weekdays and weekends. Other data classes undoubtedly exist that can potentially lead to more reliable speed profiles such as weather conditions, holidays and special events. Otherwise, more sophisticated data clustering techniques such as k-means can be employed to help segment the data. Note however that doing so would only be useful if there is sufficient historical data from which to construct the speed profiles.

7.8 Improve Ambulance Speed Estimation

Another improvement would be to use a road following algorithm to estimate speeds from ambulance positions. Currently, we approximate the ambulance's path using straight lines. Since the resolution of the GPS data is 10-11 second intervals, this approximation works fairly well most of the time. However, errors start accumulating every time the ambulance makes a turn. Other factors that are related to the ambulance's speed are the ambulance's lights and sirens. It would be wise to include this information in future implementations. As we are only interested in the redeployment problem, any data with lights or sirens on should be thrown out because ambulances generally drive with their lights and sirens off during redeployment. Being able to discern deployment ambulance speeds with lights or sirens on from redeployment speeds with lights and sirens off would allow us to greatly reduce the variance of the speed estimates obtained.

7.9 Understand Data Sparseness

As a practical system, more experiments are needed to verify that the traffic server can be ported to other cities successfully. Accordingly, it is necessary to study the relationship between the sparseness of the data and the accuracy of the estimates we obtained in order to answer the fundamental question: how sparse is too sparse?

7.10 Include Other Data Sources

Although we only used AVL data to estimate the ambulances' speeds, there exists a plethora other data sources such as traffic cameras or inductive loops that can provide useful information about the state of the road network.

Then the focus would be to fuse the various sources of data together in order to update the state of the road network. The main challenge is to be able to geo-reference the data to a particular road segment and to incorporate the sensor data into the existing framework, in this case, using relative congestion. Perhaps the cheapest improvement would be to place additional GPS devices on other fleets of vehicles such as taxis, delivery trucks or police cruisers.

7.11 Parallelize the RL Algorithm

We can take advantage of the parallel nature of RL algorithm and split the required computations over multiple cores using readily available multi-core processors in order to minimize processing time. The speedup would be significant and cities significantly larger than Ottawa could easily be supported.

CHAPTER 8
Conclusion

We represented the road network as a directed graph and used a belief propagation algorithm implemented as Relaxation Labelling to obtain dense road speed estimates from sparse sensor data. We found that the algorithm is surprisingly good at compensating for the sparse and noisy sensor data by using context to resolve ambiguities. Indeed, we obtained a mean absolute error of 12-13 km/h and a standard deviation of 10-12 km/h using real automatic vehicle location data. Furthermore, the validation results obtained using K-folds cross-validation were remarkably consistent and appeared to be very stable despite variations in the input sensor data.

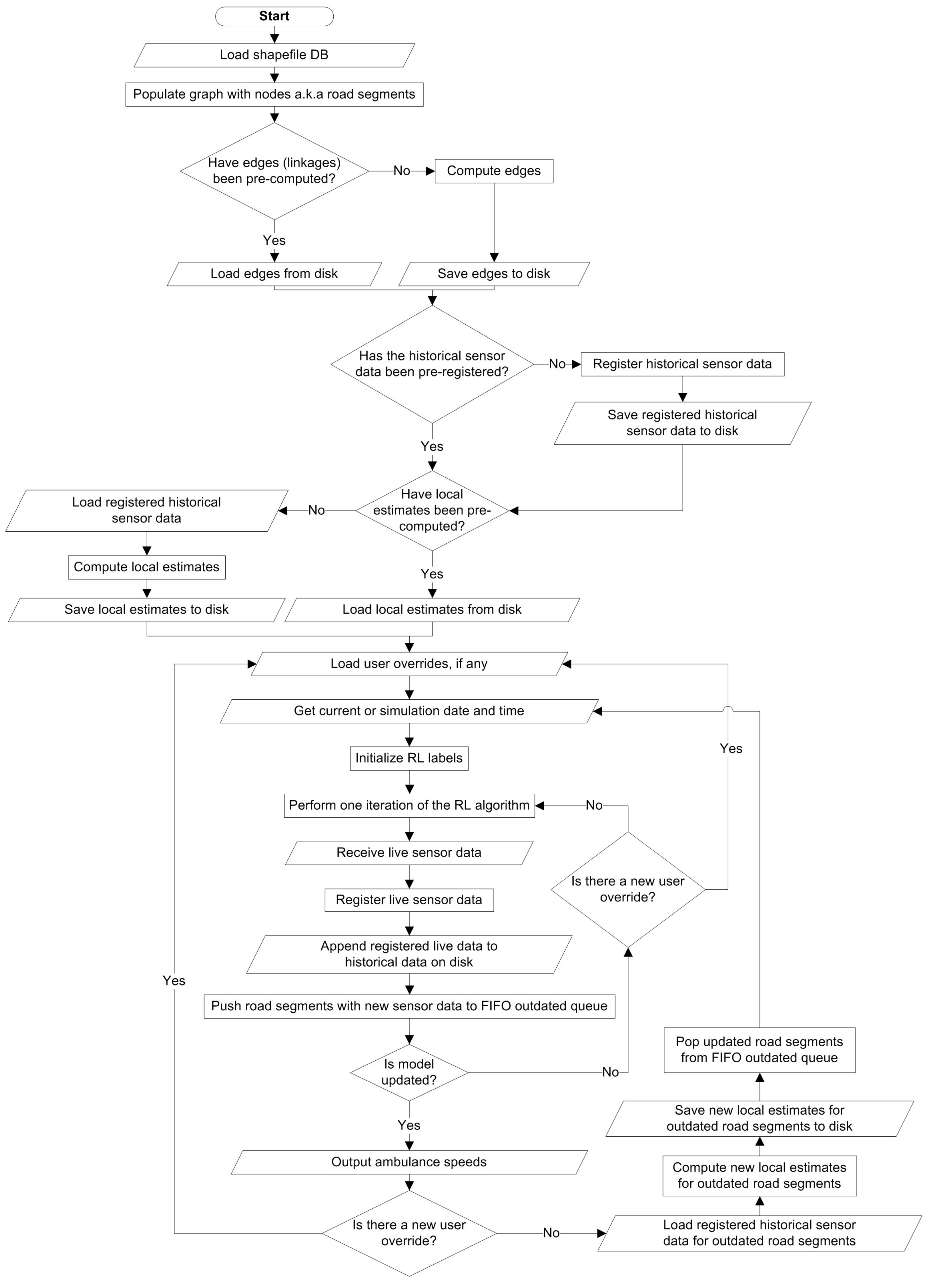
There are a number of contributions that, when put together, make the work accomplished in this thesis unique. The traffic server we propose is able to interpolate sparse and noisy sensor data using a belief propagation network. It can accommodate arbitrary data sources as long as they can be geo-referenced to a particular location in the road network and be represented in terms of relative congestion. It incorporates measures of confidence to represent uncertainties in the input data and in the output speed estimates. In terms of scale, it is able to represent the entire city of Ottawa, Canada, including both freeways and surface streets. It is currently able to update the entire road network in intervals of 5-6 minutes running on a single processing core, but can run in much less time if the processing is distributed across multiple processing cores. It has been extensively validated and tested with real AVL data using K-folds cross-validation. Finally, it can easily be ported to other cities and is locally adaptable making it quite robust to user interventions.

We believe that the work presented in this thesis is valuable because it addresses the sparse and noisy sensor data problem and offers a baseline to which we can compare eventual more complex strategies. There is much room for improvement and we offer many suggestions. Namely, the traffic server should model sources and sinks explicitly. It should incorporate statistics at intersection to compute dynamic edge weights. It should replace the linear approximation between road congestion and vehicle counts by a non-linear one. It should incorporate reserved lanes information to account for discontinuities in ambulance speeds. It should make use of super nodes that are more robust to noisy and sparse data. It needs more experimentation in order to determine an optimal compatibility function that would result in more accurate estimates and require fewer iterations. It needs to segment the data further to account for holidays, special events or particular weather conditions. It needs to use a road following algorithm that can better estimate ambulance speeds. It needs to understand the question of "‘how sparse is too sparse?"’. It needs to include other data sources to make use of all the data that are available. Finally, it needs to be parallelized to make use of the multiple core processors that have recently become widely available.

Currently, the traffic server is being prototyped in Ottawa, Canada. CAE aims to have the dispatching system serve other departments such as Police or Fire as well as other cities across the world. Such a system is useful anywhere there are dispatchers that must effectively manage a limited set of resources given a number of operating constraints.

Appendix A

Below is the flow chart of the traffic server.



Start
Load shapefile DB
Populate graph with nodes a.k.a road segments
Have edges (linkages) been pre-computed?
No
Compute edges
Yes
Load edges from disk
Save edges to disk
Has the historical sensor data been pre-registered?
No
Register historical sensor data
Save registered historical sensor data to disk
Yes
Have local estimates been pre-computed?
No
Load registered historical sensor data
Compute local estimates
Save local estimates to disk
Yes
Load local estimates from disk
Load user overrides, if any
Get current or simulation date and time
Initialize RL labels
Perform one iteration of the RL algorithm
No
Yes
Receive live sensor data
Is there a new user override?
Register live sensor data
Append registered live data to historical data on disk
Yes
Push road segments with new sensor data to FIFO outdated queue
Pop updated road segments from FIFO outdated queue
Is model updated?
No
Save new local estimates for outdated road segments to disk
Yes
Output ambulance speeds
Compute new local estimates for outdated road segments
Is there a new user override?
No
Load registered historical sensor data for outdated road segments

Figure 8–1: Flowchart of the Traffic Server

References

[1] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433, 1976.

[2] W.J. Christmas, J. Kittler, and M. Petrou. Structural Matching in Computer Vision Using Probabilistic Relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 749–764, 1995.

[3] P.W.H. Kwan, K. Kameyama, and K. Toraichi. On a relaxation-labeling algorithm for real-time contour-based image similarity retrieval. *Image and Vision Computing*, 21(3):285–294, 2003.

[4] A. Phan and F.P. Ferrie. Obtaining Dense Road Speed Estimates from Sparse GPS Measurements. *IEEE Intelligent Transportation Systems Conference*, pages 157–162, 2008.

[5] J. Miller and E. Horowitz. Freesim - a free real-time freeway traffic simulator. *IEEE Intelligent Transportation Systems Conference*, pages 18–23, 2007.

[6] Freesim http://www.freewaysimulator.com/.

[7] LE Owen, Y. Zhang, L. Rao, and G. McHale. Traffic flow simulation using Corsim. *Simulation Conference Proceedings. Winter*, 2, 2000.

[8] Corsim http://ops.fhwa.dot.gov/trafficanalysistools/corsim.htm.

[9] L. Li, W.H. Lin, and H. Liu. Type-2 fuzzy logic approach for short-term traffic forecasting. *Intelligent Transport Systems, IEEE Proceedings*, 153(1):33–40, 2006.

[10] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I France*, 2(2):2221–2229, 1992.

[11] R. Chrobok, J. Wahle, and M. Schreckenberg. Traffic forecast using simulations of large scale networks. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 434–439, 2001.

[12] J. Fawcett and P. Robinson. Adaptive routing for road traffic. *Computer Graphics and Applications, IEEE*, 20(3):46–53, 2000.

[13] Trafficmaster http://www.trafficmaster.co.uk/.

[14] N. Utamaphethai and S. Ghosh. Dicaf: A distributed architecture for intelligent transportation. *Computer*, 31(3):78–84, 1998.

[15] K.A. Redmill and U. Ozguner. Vatsim: a vehicle and traffic simulator. *Intelligent Transportation Systems, 1999. Proceedings.*, pages 656–661, 1999.

[16] J. Lei, K. Redmill, and U. Ozguner. Vatsim: a simulator for vehicles and traffic. *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 686–691, 2001.

[17] QI Yang and H.N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C*, 4(3):113–129, 1996.

[18] L. Smith, R. Beckman, and K. Baggerly. Transims: transportation analysis and simulation system. Technical report, LA-UR–95-1641, Los Alamos National Lab., NM (United States), 1995.

[19] G. Cameron, B.J.N. Wylie, and D. McArthur. Paramics: moving vehicles on the connection machine. *Proceedings of the 1994 ACM/IEEE conference on Supercomputing*, pages 291–300, 1994.

[20] M. Fellendorf. Vissim: a microscopic simulation tool to evaluate actuated signal control including bus priority. *the 64th ITE annual meeting, session*, 32, 1994.

[21] A. Barisone, D. Giglio, R. Minciardi, and R. Poggi. A macroscopic traffic model for real-time optimization of signalized urban areas. *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, pages 900–903, 2002.

[22] Shapefile c library http://shapelib.maptools.org/.

[23] Boost c++ libraries http://www.boost.org/.

[24] R. Hummel and S. Zucker. On the foundations of relaxation labelling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):267–287, 1983.

Index

LIST OF ABBREVIATIONS

AVL: Automatic Vehicle Location

CM: Congestion Measure

DMTC: Distributed Traffic Management Centres

ERSI: Environmental Systems Research Institute

ETA: Estimated time of arrival

EVC-in$_i$: Expected vehicle counts due to a node i's in-neighbours

EVC-out$_i$: Expected vehicle counts due to a node i's out-neighbours

KCL: Kirchhoff's Current Law (KCL)

LUT: Lookup Table

MAE: Mean Absolute Error

OPS: Ottawa Paramedic Service

RCap$_i$: Road capacity of a node i

RCon$_i$: Relative congestion of a node i

RL: Relaxation Labelling

VC$_i$: Vehicle counts of a node i

VFM: Vehicle Flow Model