DISTRIBUTED AND ADAPTIVE TRAFFIC SIGNAL CONTROL WITHIN A REALISTIC TRAFFIC SIMULATION

by
Dave McKenney

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

MASTER OF COMPUTER SCIENCE

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
September, 2011

Your file Votre référence
ISBN: 978-0-494-83183-0
Our file Notre référence
ISBN: 978-0-494-83183-0

Table of Contents

List of Tables

List of Figures

Abstract

Left unmanaged, high vehicle volumes within urban traffic networks results in congested and slow moving traffic. This, in turn, leads to many negative economical and environmental consequences. It is important, then, to efficiently control the vehicles within these networks. Many researchers have made an effort to improve the efficiency of traffic signals using intelligent systems. However, many systems developed have been evaluated using unrealistic traffic models which often include single intersections and static traffic volumes. This thesis introduces a distributed/adaptive algorithm which can operate within a realistic environment. A realistic model of a section of downtown Ottawa is also developed and used to test the performance of the algorithm. Through simulation, acceptable algorithm parameters are identified and the algorithm's performance is compared to that of a fixed controller. It is found that an adaptive approach to signal control results in higher vehicle speeds than those found using a fixed controller.

Acknowledgements

I would like to begin by thanking my thesis supervisor, Professor Tony White. Without his generous guidance and support, this research would never have been possible. I greatly enjoyed our weekly meetings, despite (or because of) the fact that the discussion would often move quickly from work to the newest Apple products. I should also thank Professor White for the financial support he provided over the past two years, as well as the help he provided me in finding external sources of funding as well.

I should also thank my family for their support over the last two years (financial and otherwise). Thanks to my Dad for constantly telling me to get a job and thank you to my Mom for trying to explain to him what it is I actually do with my time. Also, thank you to my brother who has entertained me, as always, with his constant desire to pursue a new and expensive hobby (e.g., go-karting, kayak fishing).

I would also like to thank my partner, as she has been there for me whenever I needed her. She has also been happy to distract me from my work, whether I needed to be or not. For that reason, I would also like to wish her good luck as she heads to graduate school, as I will be repaying the favour.

Finally, a sincere thank you must be given to the numerous people at the City of Ottawa. Without their interest and help, this work would not be the same. I look forward to working with them further as our research continues in the future.

Chapter 1

Introduction

1.1 Introduction

As populations within cities around the world climb, the number of vehicles present on the roadways of these cities also increases, resulting in slow moving, congested traffic. In fact, it has been shown that traffic jams form within a road network when volumes become too high, even if there are no incidents or bottlenecks (Sugiyama et al., 2008). One method of improving the traffic flow within urban environments has been to build more roadway or to increase the size of the existing road infrastructure (e.g., increasing the number of lanes on a busy roadway). This strategy, however, carries a high economical penalty and also consumes a large amount of the limited space available within an urban setting. Another method of increasing the efficiency of a traffic network is to improve the ability of traffic control devices to efficiently control the traffic flows within the network. One of the principal targets of this type of optimization are the traffic signals, which control opposing traffic flows at junctions within the network. In-car intelligence is also being increasingly investigated (e.g., AudiWorld, 2011) in an attempt to increase network efficiency, but this falls outside the scope of this thesis. Pedestrian traffic can also affect the flow of traffic within a network, but this interaction is also outside of the scope of this thesis.

A simple approach to improving traffic signal performance is to develop signal plans based on historically measured traffic data. There are, however, several problems related to this approach. One problem with this type of solution is that the historical data becomes inaccurate over time, leading to inefficiency within the controller (Bell and Bretherton (1986) proposed a decay in performance of 3% a year). Another problem with this type of approach is that it implements a fixed solution for specific times within the day. Even if the historical data accurately describes the overall traffic volumes within the network, it is unlikely that the current traffic volumes

are exactly the same as historically predicted. This again causes inefficiency within the signal controller, as time devoted to certain flows could be used to serve others which currently have higher demand. Recent traffic systems research, however, aims to develop intelligent signal control systems which are capable of monitoring traffic state through available sensors and making control decisions based on the available data. These intelligent systems attempt to build traffic models in real-time, creating traffic signal plans which effectively control the volumes present within the models. These approaches avoid the aging problem inherent in the use of historical data and, to varying degrees, address the inefficiencies present using fixed signal plans.

The intelligent systems which have been developed for real-time signal control can be further divided into those that are centralized in nature and those which rely on distributed computing. Distributed systems, in the domain of traffic control, offer a number of advantages. First, distributed computation generally allows a system to scale much better than a centralized counterpart. This can be very important when dealing with city-wide traffic control, where there are thousands of intersections which require real-time observations and decisions to be made. Distributed systems are also much more robust than centralized systems. Assuming all intersections within a city traffic network are connected to a centralized control centre via a wired communication network, a single break in the network could eliminate the intelligent control of a large number of intersections. If the break in communication comes extremely close to the control centre (or the centre loses all communication abilities), the performance of the network can significantly degrade. Distributed systems which rely strictly on local communication, on the other hand, can deal with these communication errors much more effectively. If a communication line links only two neighbouring intersections, a break in the line can only effect those two intersections. This increased system robustness can be extremely important within the traffic control domain, as even a small malfunction at a single intersection can quickly compound into network-wide problems.

While many intelligent control systems have been developed, much of the research tests these systems on simple networks with theoretical (and often static) traffic volumes. While these types of investigations are useful in the initial development stage,

solving these simple problems fail to demonstrate any real-world applicability. To improve intelligent traffic system research to a point where the developed systems can address real problems, models must be developed which reflect real-world networks and traffic scenarios.

1.2 Goals

As mentioned above, to demonstrate true performance abilities, intelligent traffic control systems should be tested on realistic traffic scenarios. The first goal of this thesis, then, is to outline a methodology which can be used to develop realistic traffic models for the testing of intelligent control systems. Second, the thesis aims to create a multi-agent control algorithm capable of adapting signal plans in real-time based on local traffic state observations. Finally, the performance of this algorithm will be evaluated within a simulated environment using realistic traffic models.

1.3 Problem Statement

Many traffic control approaches rely on fixed-length signals, which fail to address real-time traffic volumes. While many intelligent control approaches have been developed to solve this problem, they are typically implemented and tested within small and simple traffic simulations. These simple models fail to capture the many interactions that may occur within a larger traffic system (e.g., interactions between intersections). Also, when using some approaches that involve centralized systems, testing on small networks fails to demonstrate if the system is capable of scaling to control a realistically-sized network.

This thesis creates a distributed, and thus scalable, algorithm which can adapt traffic signal plans to fit observed traffic volumes. This thesis also develops a realistic traffic model on a moderately-sized (63 intersection) network to evaluate the performance of the algorithms created.

1.4 Contributions

The following list outlines the main contributions made by this thesis.

- A review of previous intelligent traffic signal control methods (presented in Chapter 3), along with the identification of the advantageous characteristics which should be possessed by an intelligent signal control system.

- Chapter 5 describes the development of a realistic traffic model based on a section of the downtown area of the City of Ottawa. The traffic network within the model closely resembles the real-world network, while the vehicle routes used within the model are generated using data provided by the City of Ottawa.

- A detailed description of the methodology used during creation of the realistic model, which proves feasibility and provides a starting point for others who wish to develop further models is also included within Chapter 5.

- Development of a parameterized multi-agent algorithm (see Chapter 6) which generates signal plans based on real-time local state observations. The algorithm is also capable of operation within complex/realistic traffic scenarios.

- Proposal (Sections 6.7.3-6.7.7) and comparison (Section 7.3.6) of a number of volume calculation methods to be used within the main control algorithm, including purely reactive and predictive approaches.

- Optimization of a number of algorithm parameters, along with discussion of the effects the various parameter values have on overall system performance (presented in Section 7.3).

- Comparison of the proposed algorithm's performance to that of a fixed control plan based on actual signal plans provided by the City of Ottawa (included in Section 7.5).

1.5 Organization

The structure of the remainder of this thesis is as follows. Chapter 2 presents background material necessary to understand the work presented within the thesis. Chapter 3 reviews the existing intelligent signal control research literature, with a summary outlining preferential characteristics within an intelligent signal control system. An

initial algorithm is developed and evaluated within Chapter 4, showing the promise of adaptive signal control systems. Chapter 5 details the process used in the creation of a realistic traffic model using both openly available data and information provided by the City of Ottawa. An adaptive multi-agent algorithm which is capable of operation within a real-world traffic environment is presented in Chapter 6. The properties of the proposed algorithm are investigated in detail within Chapter 7, including a performance comparison between the adaptive algorithm and fixed plans created using information supplied by the City of Ottawa. Chapter 8 concludes the thesis with a summary of the main contributions and completed work, as well as discussion of important future research directions in the area of intelligent traffic systems.

Chapter 2

Background

2.1 Introduction

This chapter provides a brief review of material necessary to understand the remainder of this thesis. This includes discussion of traffic optimization (with a focus on traffic lights), general architectures which can be used in traffic light optimization and background information on several computational intelligence approaches which have been applied in the traffic optimization domain. Also included, is a discussion of traffic simulation, including a summary of various traffic simulators currently available (see Table 2.1).

The remainder of this chapter is organized as follows. Section 2.2 briefly explains the problem of traffic optimization, while Section 2.3 focuses on traffic light optimization specifically (including Sections 2.3.1-2.3.3 which detail the most often optimized parameters of traffic signals). Section 2.4 discusses several of the broad architectures which have been used for traffic light optimization, including fixed-time and traffic responsive strategies. A discussion of a number of approaches which have been used to develop signal controllers is included in Section 2.5. Section 2.6 introduces traffic simulation, with a focus on microscopic traffic simulation (Section 2.6.1). Finally, Section 2.7 summarizes the contents of the chapter and introduces the work presented in the following chapter.

2.2 Traffic Optimization

The problem of traffic optimization involves any number of methods which aim to improve the flow of vehicle traffic within a road network. These methods typically involve influencing driver behaviour (e.g., traffic lights and signage) and making network modifications (lane additions, turning lanes). While network modifications, such

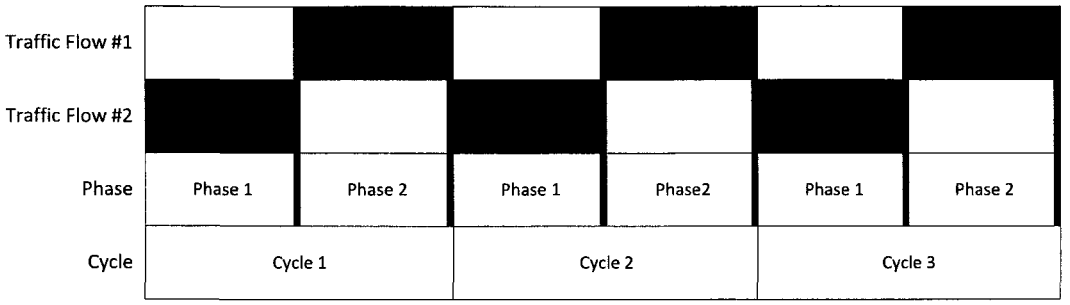


Figure 2.1: Example of cycle and phase relationship with traffic flows which can (white) and cannot (black) proceed

as lane addition, can result in drastic improvements in traffic flow, these modifications require space which is limited or non-existent in many cases. For this reason, more effort has been placed on controlling the vehicle flows within the network more efficiently using available traffic control devices.

2.3 Traffic Light Optimization

Traffic light optimization is one of the most effective, and thus most researched, methods of improving vehicle flow within a traffic network. For obvious safety reasons, conflicting flows present at intersections within a traffic network require regulation and control. The effectiveness of the control method applied to the intersections largely determines the overall performance of the network. The three most important parameters which determine the behaviour of a traffic signal plan are the phase lengths, signal cycles and offset values. These parameters, as well as the effects of their optimization, are explained below and a graphical representation is included in Figure 2.1.

2.3.1 Phase Lengths

A phase is a time period in which certain vehicles may travel through the controlled intersection, while others may not. Each phase has a specific set of lights which are green and another set of lights which are red. These sets, in turn, define which vehicles can proceed through the intersection and which vehicles must wait. Each

phase within a signal plan has a specified length, which determines how long that phase will last during each light cycle. Choosing effective phase lengths allows for improved flow of vehicles through the intersection, as more congested lanes are allowed to proceed for a longer time than less congested lanes. Optimizing phase lengths for a single intersection, however, can have drastic consequences at other locations within the network as it can change the vehicle volumes at downstream intersections.

2.3.2 Cycles

A cycle is composed of a number of phases. Generally, a cycle has a fixed time length which equals the sum of all of its phase lengths. Each traffic light typically implements a single cycle at any given time. This cycle will run through completely, before repeating. There are several different methods of optimizing a signal plan's cycle. First, the length of the cycle can be increased or decreased, allowing phases to repeat more/less often. It is generally thought that shorter cycle lengths can be much more effective with low traffic volumes, as the phases change more quickly to allow sporadically arriving vehicles to proceed (Findler and Stapp, 1992). Higher traffic volumes though, can benefit from increased cycle lengths, as more vehicles can proceed through the intersection during a single phase. Also, a longer cycle length decreases the percentage of time that all traffic flows must be stopped due to safety requirements while switching phases. The order in which phases occur during a cycle can also affect the utility of a signal plan. While the order matters much less at simple intersections, it can become important in more complex control scenarios, where there may be many different phases allowing traffic to travel in various directions (e.g., turning lanes with advanced green lights). Finally, with more complex intersections, it may be beneficial to add/remove certain phases depending on the traffic state (observed or predicted) at the current time. As an example, it may be beneficial to have an advanced green light for a turning lane at one point during the day, while it would hinder traffic flow at another point in the day.

2.3.3 Offsets

An offset value specifies at which point in the cycle the first phase will begin, allowing different intersections to begin their cycles at differing times. Improving offset values results in coordination between intersections, which can allow vehicles to proceed through multiple intersections without having to stop. This phenomena is known in traffic research as a 'green wave' and can be an important factor in improving overall network performance (for more on green waves, see Robertson and Bretherton, 1991).

2.3.4 Safety Requirements

Since conflicting traffic flows are present at intersections, there are several safety requirements that must be realized by a traffic light plan. First, each signal plan must ensure that conflicting traffic flows cannot proceed at the same time. Also, there must be a specified amount of time where all traffic flows are stopped before switching from one signal plan to another. This time period allows vehicles that may have just entered the intersection to exit before conflicting traffic flows move into the intersection.

Further to these vehicular safety constraints, many jurisdictions implement protocols which ensure the safety of pedestrians as well. As previously mentioned, however, the consideration of pedestrians falls outside of the scope of this thesis.

2.4 Traffic Light Optimization Architecture

Papageorgiou et al. (2003) outlined a dichotomy of traffic light optimization techniques: fixed-time strategies and traffic responsive strategies. Each of these strategies can be further subdivided into those that are isolated (controlling only a single intersection with no consideration of other intersections) or coordinated (considering more than one intersection). The details of the two broad approaches are explained further in the following two sub-sections.

2.4.1 Fixed-time strategies

Fixed time strategies rely on off-line optimization algorithms, which attempt to select parameters such that an overall goal is reached (e.g., minimizing travel time, maximizing network capacity). These optimizations are performed using historically observed traffic data, as opposed to real-time observations of traffic state. This, of course, can result in poor overall performance within the traffic network for three reasons. First, there is no guarantee that traffic volumes on a given day will match those that were used to optimize the intersections' signal plans. The larger the difference between current traffic state and historical traffic state, the less effective the signal plans will be. Also, as the historical data ages, it becomes more likely that the underlying traffic volumes will change. This can become especially crucial in an urban environment, where new residential, commercial or industrial developments can result in an overall traffic volume change for a given area. Once again, these fixed plans fail to recognize these changes until new measurements are taken and new plans developed. Finally, traffic volumes can change at anytime due to disturbances caused by traffic accidents, construction or other incidents. Fixed controllers fail to realize and adapt to these disturbances, resulting in inefficient signal control.

The fact that isolated fixed-time strategies do not take into account other intersections can result in further problems. For example, one intersection may be optimized in such a way that it allows vehicles to constantly flow in one direction. An intersection downstream from this area, however, will not be anticipating the increase in traffic volume because it considers only historical observations when deciding on its own signal plan. The downstream intersection then, will not be prepared for this new situation and will operate inefficiently. If the efficiency of the downstream intersection's plan becomes poor enough, major problems can occur as traffic jams form and propagate throughout the network. Coordinated fixed-time strategies address this problem, by analyzing the overall performance of the signal plans at all intersections within the network. While the computational requirements for this analysis can be extremely high, the off-line optimization allows for the time necessary to find a solution.

2.4.2 Traffic Responsive Strategies

Traffic responsive strategies aim to optimize signal plans using real-time traffic state observations. This real-time measurement of traffic is generally achieved through the use of sensors placed within the road network, which are capable of detecting vehicles as they pass. The problem of signal inefficiencies due to measurement ageing is eliminated when using real-time observations to determine signal plans. Another issue arises though, as traffic responsive strategies must generate signal plans in real-time and therefore cannot perform as much analysis as the off-line optimization methods used with fixed-time strategies. For this reason, many classic optimization algorithms cannot be used for large traffic networks.

Traffic responsive strategies typically implement an architecture such as that shown in Figure 2.2. First, an observation period occurs, in which sensors within the network generate information about the current traffic state. This information is then passed to the optimization algorithm, which performs the steps necessary to generate new signal plans for the network being considered. The traffic lights then implement these new plans until another observation period is completed and the plans change once again. Essentially, traffic responsive signal controllers create a model of traffic flows in real-time and optimize the allocation of resources (green time) based on the predicted traffic volumes. Using this strategy, the signal plans adapt throughout the day to meet the current traffic state, as opposed to a historically observed traffic state. As with isolated fixed-time control strategies, isolated traffic responsive control at an intersection can cause problems at other intersections within the network. This can happen when one intersection implements a plan which results in a traffic volume far from what another intersection has anticipated. This problem, however, is not as costly when using a traffic responsive strategy because the failing intersection will modify its plan much sooner than a fixed-time control strategy. Still, the level of coordination between intersections, as well as the speed at which intersections can adapt to unexpected traffic volumes, can greatly affect the overall performance of the network.

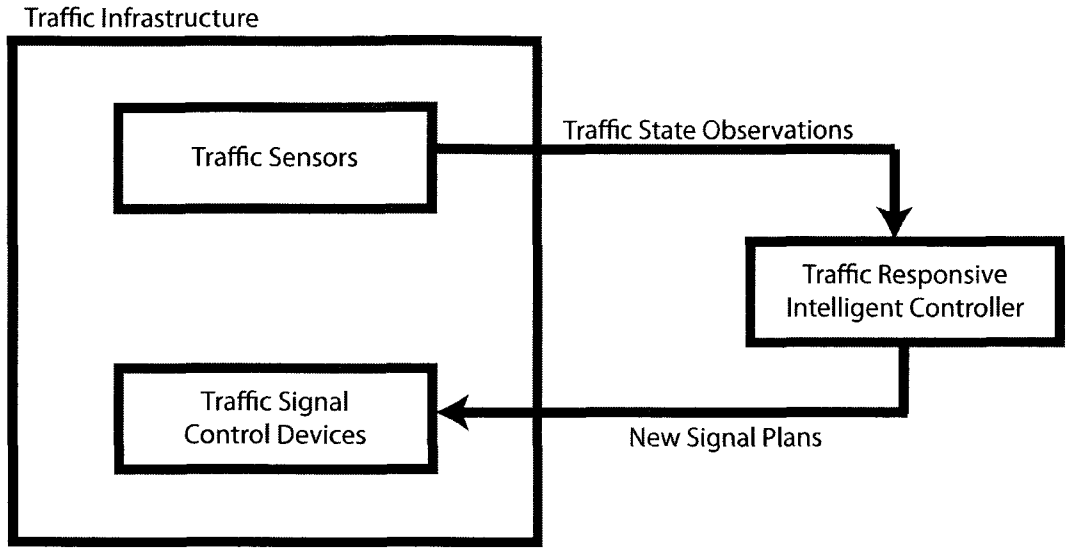


Figure 2.2: A simple responsive traffic system architecture

2.5 Computational Approaches to Traffic Light Optimization

This section includes background information regarding a number of the approaches which have been used in the traffic light optimization domain. This approaches include multi-agent systems (Section 2.5.1), evolutionary computing (Section 2.5.2), fuzzy logic (Section 2.5.3) and swarm intelligence (Section 2.5.4).

2.5.1 Multi-agent Systems

A multi-agent system consists of an arbitrary number of intelligent agents acting/ interacting within a common environment. Wooldridge (2002) also outlined the following three characteristics which the agents within a multi-agent system must possess:

- Autonomy: The agents must be able to act on their own accord, making informed decisions based on the information available to them.
- Decentralized: There is no centralized controlling agent that makes decisions for all other agents.
- Local views: No agent has a global view of the system. Actions must be made

based on information available locally, although locally available information may have been communicated to the agent from further away.

Multi-agent systems are particularly well suited to solving problems which cannot easily (or possibly) be solved by a single agent. Some areas where multi-agent systems have been applied include financial systems (Decker et al., 1997), disaster response (Schurr et al., 2005), social network analysis (Sabater and Sierra, 2002) and traffic control (Balaji and Srinivasan, 2010).

2.5.2 Evolutionary Computing

Evolutionary Computing is a research area within Computer Science, which aims to develop solutions for problems by exploiting the power of natural evolution (Eiben and Smith, 2003). The natural ability of evolution in problem solving can be easily seen by looking at the many diverse and successful species on Earth. Two of the most common evolutionary computation approaches include genetic algorithms (Mitchell, 1998) and genetic programming (Koza, 1992). Typically, evolutionary approaches repeat a process similar to the three steps below until a specified ending condition is met:

1. Create a population of individual solutions: This can be generated at random initially, but uses selected individuals from previous generations, modified by genetic operators (mutation, crossover), in later iterations.

2. Evaluate the fitness of the individual solutions: This process allows each individual to be assigned a fitness value which generally represents the effectiveness of that individual's solution.

3. Select individual solutions based on fitness: As with natural evolution, some individuals will be more likely to pass genetic material to the next generation. Using the fitness values of the individuals, a number are selected and used to create the next generation.

These evolutionary approaches usually involve centralized computing, with a large level of computing power being required to simulate traffic for each individual within

the population for a number of generations. The computing power required also continually increases as the size of the network under consideration increases. For this reason, it seems infeasible to use an evolutionary approach to evolve control of large traffic networks in real time.

2.5.3 Fuzzy Logic

The idea of fuzzy logic was first proposed by Zadeh (1996). When using fuzzy logic, truth values of variables can take on a continuous value in the range of $]0, 1[$ as opposed to the traditional binary truth values of 0 or 1. Another difference between fuzzy and binary logic is that fuzzy logic tends to use linguistic variables, as compared to the numerical variables of binary logic. This use of linguistic variables, combined with the idea of degrees of truth, is useful in domains with variables that are not easily attributed to one group or another. For example, is the traffic volume in a traffic network currently low or high? Fuzzy logic allows truth values such as 0.7 low and 0.3 high (for a medium-low volume), or 0.95 high and 0.05 low (for a very high volume). An example of how these values are distributed is shown in Figure 2.3, which contains an example fuzzy membership function. Also, as opposed to other intelligent methods such as neural networks (the power of which is most often stored in the connection weights), the use of linguistic variables allows fuzzy logic algorithms to work in terms that humans can easily understand. Decisions in fuzzy logic are usually made using a rulebase which can be developed by expert knowledge, trial-and-error or an automatic method such as a genetic algorithm. Rules within the rulebase typically take the form of `if VARIABLE1 is VALUE1 then OUTPUT`, or as a more concrete example: `if traffic_volume is high then cycle_length is high`. While the use of a rulebase carries the previously mentioned advantages, the performance of the controller greatly relies on the effectiveness of the rules developed and it can be difficult to determine if the rules being used are helpful.

2.5.4 Swarm Intelligence

Swarm intelligence deals with the emergent collective intelligence of a group of simple agents (Bonabeau et al., 1999). There are a number of important swarm intelligence

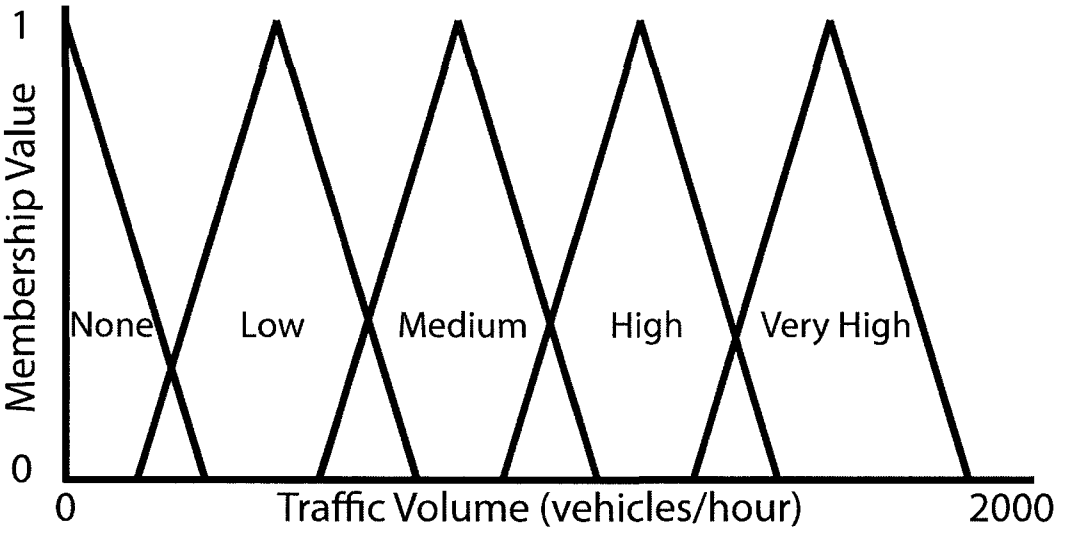


Figure 2.3: An example fuzzy membership function

ideas which may be applied in the traffic domain. The first involves finding routes within networks using pheromones, as is done by ants when searching for food. In this case, pheromone levels are increased on shorter, more desirable paths, making those paths more likely to be traversed. Eventually, this type of system converges to a path which is hopefully optimal (although locally optimum path discovery is possible, as with many optimization techniques).

Another common swarm intelligence technique is that of task allocation (Theraulaz et al., 1998), which aims to assign different jobs to agents to accomplish some goal. In the domain of traffic signal control, each phase within the signal plan can correspond to a job which must be selected by the controlling agents. Each agent within the system has a specific propensity for each possible job (referred to as the response threshold of the agent for the job), as well as an amount of motivation for each job. This motivation, once again, is provided by virtual pheromones which indicate traffic volume levels. Considering both the motivation for a job to be selected, as well as the specific agent's propensity for each job, a single job can be selected at any time by each agent. The propensity of the agent to select the chosen job is then modified after a period of time based on the effect the job selection had on the system (i.e. increasing the likelihood of choosing jobs which had a positive effect on

the system). Details on the application of this technique to traffic signal control is included in Section 3.6.

A third swarm intelligence technique which has been used for traffic light optimization (and many other search problems) is particle swarm optimization. In particle swarm optimization, a group of particles, each of which represents a possible solution, is moved through a search space. The movement of each particle is based around a position and velocity which are modified throughout execution in an attempt to find the optimal solution. In this case, signal plans of the network are encoded into vectors, with the fitness of these vectors being evaluated through simulation. The velocity of all particles within the system is affected by the best solution found so far, drawing more particles near to this solution in hopes that an optimum lies nearby.

2.6 Traffic Simulation

Traffic simulation is the modelling of vehicle traffic systems for the purpose of investigating/planning transportation systems. These simulations offer a safe and convenient environment to investigate possible modifications to transportation systems. Traffic simulation as a whole can be largely divided into two broad approaches (microscopic and macroscopic), with another approach (mesoscopic) being a hybrid of the two. All three approaches are explained in more detail below, with a focus on microscopic simulation and the various components involved in creating a microscopic simulation.

2.6.1 Microscopic Simulation

Microscopic traffic simulation relies on individual driver behaviour. Each vehicle within the simulation environment is updated discretely using a specified driver model. Driver decisions, then, are based on both properties of the traffic network (i.e., speed limit, lanes) and nearby vehicles. Using a driver model for decision making allows for a heterogeneous distribution of vehicle behaviour, with vehicles acting differently based on their driver model parameters. A summary of the important modules within a microscopic traffic simulation environment are provided below, while Table 2.1 provides a summary of various available microscopic simulators.

	AIMSUM	CORSIM	FOSIM	MITSIM	PARAMICS	SimTraffic	SMARTAHS	FREESIM	SUMO	MATSIM
Free	No	No	No	Yes	No	No	Yes	Yes	Yes	Yes
Open Source	No	No	No	Yes	No	No	Yes	Yes	Yes	Yes
OS	Windows Linux OS X	Windows	Windows	Linux	Windows	Windows	Linux	Linux Windows OS X	Linux Windows OS X	Linux Windows OS X
Documentation/Support	Yes	Yes	Yes	Limited	Purchasable	Yes	Very Limited	Very Limited	Documentation Active Community	Yes
Languages/Tools	N/A	N/A	N/A	C PVM	N/A	N/A	C	Java mySQL Perl Apache	C++ Python	Java
Network Gen	GUI	GUI	GUI	TextFiles GUI	GUI	GUI	Text Files	Text Files	XML OSM Import	OSM Import
Route Gen	OD GUI	GUI	GUI	OD Text File	OD GUI	GUI Turning Ratios	OD Text File	Text Files	XML Generators	Text Files
Sim GUI	3D	Yes	Yes	Yes	3D	3D	Graphs	Yes	Yes	Yes
Output	Graphs	XML CSV	Graphs	Text	XML CSV HTML	Graphs	Graphs	N/A	XML	Text
Network Types	Any	Any	Freeways	Any	Any	Any	Freeways	Freeways	Any	Any

Table 2.1: Summary of attributes for various microscopic traffic simulators

Network Creation

Network creation is the process of designing a road network for use within a simulation environment. This generally includes specifying properties of road segments (e.g., number of lanes, speed limit, start/end location) and the connections between lanes at intersections. While some environments require these specifications to be maintained in text files, which can be extremely time consuming, others can directly import traffic networks from other sources such as OpenStreetMap (OpenStreetMap, 2011) or geographic information systems (GIS). This automatic importation of traffic networks, along with other beneficial simulator attributes, is included in Section 5.2, which discusses the reasons for choosing the traffic simulator used within this thesis.

Route Generation

Route generation involves specifying routes for each vehicle that will be included in a simulation. The requirements of the route generation phase vary widely between simulators, as each one accepts different levels of information and expects routes in different formats. Generally, the route of each vehicle is specified either explicitly with a set of road segments that must be traversed, or through an origin/destination pair, where the route is generated dynamically when the vehicle enters the simulation. Using either of these methods, it would be unrealistic to generate a set of routes for a meaningful simulation by hand. For this reason, most simulation environments provide tools which automatically generate vehicle routes based on specified parameters.

Two common methods of generating routes are explained below.

Origin/Destination (OD)

Using an origin/destination approach typically involves specifying a number of subnetworks/areas, along with a number/percentage of vehicles which will begin their trip in one area (the row) and end in another area (the column). These numbers can be used to generate trips probabilistically throughout a time period, with routes between the two points being computed using a routing algorithm (often shortest path). An example OD matrix is shown in Table 2.2.

Turning Ratios

When defining routes using turning ratios, probabilities for each option at a turning point in the network must be specified (or calculated from vehicle counts). These probabilities must total 1 for each turning point, as each vehicle must take one of the possible directions. Vehicle introduction rates, which determine how many vehicles enter the simulation and where they originate from, along with 'sink edges', must also be specified for the network. When a vehicle's route reaches a 'sink edge', the route is completed and the vehicle is removed from the simulation. The route creation program then determines vehicle's routes probabilistically (based on the turning ratios) from the origin of the vehicle until it reaches a sink edge. Table 2.3 presents an example table used for turning ratio generation. This table includes, for each interval, the number of vehicles which exited left (LT), right (RT) or straight (ST) for each incoming street. The total vehicles exiting each incoming street is also provided (SUB).

Output

The output abilities of a simulation environment play a large role in determining that environment's utility. This is because it would be impossible to make any conclusions without the ability to measure relevant indicators. The most basic information (which is included in almost every simulator) generated from a microscopic simulation is valuable information about the vehicles within the network, such as speed and trip

	Area 1	Area 2	Area 3	Area 4	Area 5
Area 1	0	15	3	9	9
Area 2	18	0	27	5	8
Area 3	1	21	0	6	10
Area 4	9	7	7	0	13
Area 5	7	6	11	15	0

Table 2.2: An example origin-destination matrix for 5 areas within a network

Time Interval		Eastbound				Westbound				
Start	End	LT	ST	RT	SUB	LT	ST	RT	SUB	Total
7:00	8:00	83	81	0	164	0	39	67	106	270
8:00	9:00	112	106	0	218	0	73	52	125	343
9:00	10:00	91	67	0	158	0	61	66	127	285
11:30	12:30	52	56	0	108	0	54	74	128	236
12:30	13:30	99	25	0	124	0	75	63	138	262
15:00	16:00	57	39	0	96	0	53	79	132	228
16:00	17:00	93	53	0	146	0	46	50	96	242
17:00	18:00	83	31	0	114	0	76	60	136	250
8 Hour Total		670	458	0	1128	0	477	511	988	2116

Table 2.3: An example table used for turning ratio calculation and route generation

times. Some simulators also provide aggregated information about road segments within the network (or the network as a whole), often including measurements such as average vehicle speed, vehicle density and total number of vehicles. More advanced simulators can generate calculated measurements involving noise and emission levels, which of course, are of high value to those interested in designing systems which minimize pollution of various kinds. This type of information is not considered in this thesis, however, as the work is only concerned with increasing average vehicle speed within the network.

Along with the ability to produce these observation outputs, the immediate usability of this information may be of some importance to many people. For example, many of the simulation environments available write these observations directly to text files. This data then requires some level of treatment to generate usable data such as graphs. Some of the more advanced simulators, however, allow for a large number of graphs to be generated automatically from the output.

Car Following Models

Car following models are a common form of driver behaviour model applied within microscopic traffic simulators. Car following models attempt to replicate the actions of real-world drivers by making behavioural decisions based on nearby vehicles. The use of a car following model for driver behaviour within a traffic simulator allows for the observation of interactions between vehicles within the network. Parameterized car following models also allow for heterogeneous driver behaviour, with behaviour differing based on the various parameters used (e.g., comfortable following position, acceleration).

2.6.2 Macroscopic Simulation

Instead of modelling individual vehicle behaviour, macroscopic traffic simulators rely on traffic flows, densities, and speed to model transportation systems. This approach draws comparisons to fluid flows and was actually compared to the movement of a river in the original work by Lighthill and Whitham (1955). The entire network within a macroscopic simulator is divided into many small sections, with the state of

each of these sections being modelled based on the key parameters (flow, density and speed).

2.6.3 Mesoscopic Simulation

Mesoscopic traffic simulators combine elements of both macroscopic and microscopic traffic simulation. Mesoscopic simulators typically model individual vehicles (a microscopic approach), however, the actions of these vehicles are based on overall averages (a macroscopic approach).

2.7 Summary

This chapter provided a brief introduction to a number of the central concepts of this thesis. Section 2.2 introduced the general problem of traffic optimization, identifying the optimization of traffic flows (as opposed to the expansion of traffic infrastructure) as a key goal. Section 2.2 discussed traffic light optimization, including a description of the main traffic light parameters addresses by optimization algorithms (Sections 2.3.1-2.3.3). Sections 2.4 and 2.5 present common architectures used in traffic light optimization and a number of computational approaches which have been applied within previous traffic research. Traffic light optimization is addressed further in Chapter 3, which details previous optimization research from the traffic domain, as well as Chapters 4 and 6 which introduce a distributed/adaptive traffic light optimization approach. Section 2.6 introduces traffic simulation, with a focus on microscopic simulation (Section 2.6.1) which is used within this thesis. This included discussion of the main steps involved when creating scenarios for use within a microscopic traffic simulator. Within Section 2.6.1, Table 2.1 presents a summary of the key attributes of a number of available microscopic traffic simulators. Chapter 5 of this thesis deals with the creation of a realistic traffic model and will further detail the steps taken in developing scenarios for microscopic traffic simulation.

Chapter 3

Related Work

3.1 Introduction

A number of different intelligent computing techniques have been applied in the domain of traffic control. Some of these (see Sections 3.2 and 3.8) rely on powerful computing systems or large amounts of pre-calculated data. Others (Sections 3.5) develop control systems which operate using simple, locally-aware agents. While most of these works rely on technology that is currently available, others (see Section 3.3) plan for future technological advances by including theoretical requirements such as automated vehicle control. This chapter outlines the state of the art in intelligent traffic control systems, detailing the various different approaches that have been used. Each section outlines work completed using a specific approach, with discussion of important factors such as adaptability, scalability and ability for real-time control. For easy reference, a table summarizing the overall advantages/disadvantages of each approach is presented at the end of each section. Section 3.11 concludes the chapter by identifying, from analysis of the works presented, a number of requirements for the algorithms developed in the remainder of the thesis.

3.2 Evolutionary Computation

Montana and Czerwinski (1996) presented one of the first works using an evolutionary approach for intelligent traffic signal control. Two strategies were developed and compared in this work: a strongly-typed genetic programming (Montana, 1995) approach to generating a traffic controller and a genetic algorithm (Holland, 1975) approach to optimizing fixed time signal plans. A description of each approach is provided below:

Genetic Programming A genetic programming parse tree is evaluated at every second, with the Boolean value of this tree resulting in either phase change

(true) or no change (false). The parse tree uses typical Boolean functions (AND, OR, NOT, >) as well as a number of terminals such as number of vehicles approaching a light, whether vehicles are backed up to a sensor downstream and how long the current light has been in operation.

Genetic Algorithm Each individual consists of three real-value entries for each intersection in the network, with the first 2 numbers representing the lengths of each phase and the third number representing the signal offset value.

For both approaches, the fitness function was based on total delay within a simulation, with lower delay resulting in higher fitness. The performance of both the genetic programming and genetic algorithm strategy were compared for three different small networks with constant traffic flow rates. The performance of the genetic programming approach was better than that of the genetic algorithm for all three cases that were investigated. The individuals which performed best in the training scenarios for both approaches were also compared with different random number seeds to investigate the generalizability of the solutions. Once again, the genetic programming individuals performed the best, with results closely comparable to those found in training. The genetic algorithm had much wider variability, showing that the fixed signal time approach did not generalize as well.

Vogel et al. (2000) applied a different evolutionary approach, that of evolution strategy (Beyer and Schwefel, 2002), to evolve both phase order and phase length for a single intersection. Using this approach, a set of chromosomes is used to encode the various parameters necessary to define the traffic signal plan. The chromosomes used are explained below:

Phase Chromosomes This set of chromosomes encodes which flow directions belong to each phase. The number of chromosomes used depends on the number of phases, which can range from 2 to the number of flow directions.

Phase Order Chromosome This chromosome encodes the order in which the phases of the signal plan occur.

Green Time Chromosome This chromosome contains the amount of green time that should be allocated for each phase in operation.

Traffic Information Chromosomes This set of chromosomes was used to encode things such as parameters for traffic detectors (e.g., sampling rate).

The specific evolution strategy (ES) used in this work was the (1+1) ES, which is synonymous with a hill-climbing approach. In each generation, the initial individual (the population is of size 1) will be mutated slightly to form a second individual. Of these two individuals, the one with the highest fitness value continues to the next generation. The ability of this approach to find effective parameter sets was investigated on a single intersection network with 4 incoming edges. Through simulation, it was shown that the proposed strategy was capable of improving the parameter set over time. In fact, it was even shown that a random initialization of parameters could be evolved just as well (and even better) as a parameter set initialized with heuristically determined parameters. The approach was not compared to any other control mechanisms though, so the overall effectiveness of this strategy is still questionable.

A genetic algorithm (GA) was used once again by Sánchez et al. (2004) to optimize the signal timing of a set of intersections. In this work, each individual encodes the length of each phase within the cycle for each intersection. An example of what each chromosome may look like is shown in Figure 3.1. Within the figure, it can be seen that the chromosome has an integer value for each phase within each intersection. Although the figure has 5 phases for each intersection, this number can vary based on the control logic required for each intersection. A traffic scenario is simulated for each individual and the population evolves in an attempt to minimize overall travel times. Using a small traffic network, the GA approach was compared to results found using a random search technique, a fixed control scheme in which every traffic light maintains the same green period as the others and a scheme in which the traffic lights switch to allow the flow with most cars present to proceed. The GA approach presented performed well in comparison to the random control (83.82% decrease in travel time) and the fixed control scheme (56.74% decrease in travel time). It failed, however, in generating plans which performed better than the third approach which maintained travel times of approximately 60% of the times found using the GA. This work was expanded further by Sánchez et al. (2008), with the GA optimization being applied to a real world traffic network and compared to signal plans supplied by the traffic

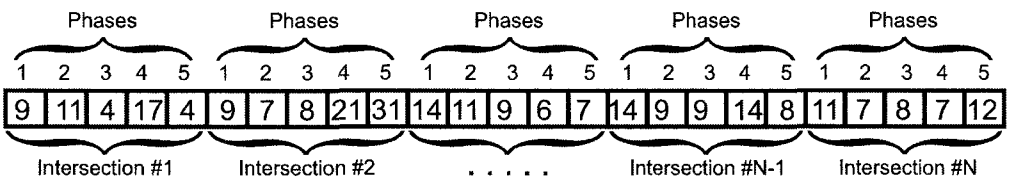


Figure 3.1: Example of a chromosome encoding phase lengths for a traffic network

authority of that area. The modelled network consisted of 20 intersections located in the city of Santa Cruz de Tenerife and the traffic flows were probabilistically created using an origin-destination matrix inferred from the information available from the traffic department. After evolving a population of 200 individuals for 250 generations, average improvements ranged from approximately 7% to over 20% when compared to 9 signal plans supplied by the traffic department. This same GA optimization approach was further investigated by Sánchez et al. (2010), this time modelling the 'La Almozara' area in Saragossa, Spain (it should be noted that the traffic volumes used within these evaluations did not fluctuate over time). While analyzing the performance of the system, the authors noted that the traffic volumes within the modelled area were simply too low, resulting in little opportunity for optimization and thus unconvincing and inconclusive results. While this approach improved over other plans, which is certainly promising, it has still been evaluated using rather small networks. As the networks increase in size (as they would when controlling an entire city network), the search space involved in finding effective signal plans will increase significantly. This may decrease the utility of this approach if there is no further work completed to divide the problem into manageable sub-problems.

3.3 Reservation Based Systems

Dresner and Stone (2004) moved away from a traditional view of traffic light operation, proposing an architecture in which intersections reserve time and space for vehicles to cross the junction. Using this approach, the space inside an intersection is divided into an NxN grid of 'reservation tiles', each of which may contain at most one vehicle at any time (to prevent collisions). Driver agents then communicate with

Table 3.1: Advantages and disadvantages of using an evolutionary computing approach for intelligent traffic signal control

Advantages	Disadvantages
Can easily solve simple networks. Can deal with static traffic volumes easily. Have been shown to work with real networks.	Large amount of computing power required. Centralized computing power required increases as problem size increases. Large problems may be difficult to solve in real-time.

the intersection to reserve the space they will need to cross the intersection, passing several important pieces of information such as vehicle attributes, arrival velocity and arrival time. Using this information, the intersection agent will check to ensure that the space required for the requesting driver agent is available at the time requested before approving the reservation. If any of the necessary space is already reserved at the required time, the reservation will be rejected. In the case of a reservation rejection, the driver agent must attempt to make a new reservation until one is successfully granted. This method of intersection control can perform extraordinarily well because it eliminates the lost time from phase switching found in typical intersection control and can also allow vehicles to cross the intersection at their arrival velocity without having to stop. This approach, however, also relies on advanced and highly reliable communication networks, as well as sophisticated automated vehicle control. Reliance on these modules, however, can cause safety concerns, as will be discussed below. In this initial work, the reservation system was evaluated on a very small network, consisting of a single intersection with East/West and North/South traffic flow and a number of lanes ranging from 1 to 6. In each case, the system resulted in an average delay of nearly 0 with the best case being found using 1 lane per road (0.016 step average delay) and the worst being found with 5 lanes per road (0.031 step average delay). There are, however, several key requirements that make this approach infeasible. First, vehicles must be in very specific places at specific times to avoid any collisions. While this may be possible, it requires a very precise and automated vehicle controller. Furthermore, it this type of control were possible,

it would be extremely difficult when considering all of the factors found in the real environment, such as differing road conditions. Also, traffic dynamics may result in missed reservations. For example, a vehicle may have an earlier reservation than that of the vehicle it is following, meaning it will undoubtedly miss its own reservation.

A key improvement to the original work is presented by the same authors in Dresner and Stone (2005). The original intersection reservation agent assigned reservations to vehicles based on a constant velocity. The reservation agents in the improved system, however, consider the possibility of vehicle acceleration within the intersection boundaries, which allows for a larger chance that a driver agent's reservation request will be approved. While including possible acceleration within the model allows for an increase in reservation approval, it also adds yet another variable which must be precisely controlled to avoid infringing on other vehicle's space within the intersection. This precise control of vehicle's can become even more difficult if sensor noise is included (it was not included within this work, however). This work also formalizes the communication protocol used within the system, something which was not specified in the original work.

Dresner (2006) and Dresner and Stone (2006) propose several further improvements to the prior work. First, it is suggested that the reservation agent be modified such that it can delay a response to a vehicle's reservation request. Previously, with immediate responses, a single vehicle may have a reservation approved shortly before two other vehicles request reservations that conflict with the first vehicle (though not conflicting with each other), resulting in two rejected reservations and one accepted. With the ability to delay the confirmation/rejection of a request, the reservation agent instead could approve the two non-conflicting reservation requests, while only denying the single conflicting request. Also, within the improved system, different types of vehicles can be assigned priorities (e.g., emergency vehicles are a high priority) which determine which reservations are rejected/accepted. The third improvement proposed is to treat the intersection as a market where vehicle agents place bids for time/space, with the highest bidder receiving the reservation. While no implementation details or results are presented regarding market based control (Wellman, 1993), this approach is discussed further in Section 3.4.

A much more detailed version of these works, as well as new simulation results, is presented in Dresner and Stone (2008). One of the main contributions of this new work is the consideration of human drivers. The authors acknowledge that the system would need to be implemented in stages and that it may never be fully adopted. For this reason, they consider changes that must be made to accommodate human drivers, such as ensuring that lights are green/red according to the current use of the intersection. While using a reservation-based system for intersection control has seemed very promising so far, the results from the experiments including human driver within the system show a dramatic decay in performance. With 1% of drivers being human, the average delay of vehicles is much higher than with no human drivers. In fact, around the median range of vehicle rates tested, 1% of drivers being human caused average delay to increase nearly 500%. These results signify a reliance on an extremely advanced intelligent driver model which is used to control all vehicles within the network, which make this approach infeasible assuming the system is not adopted by 100% of vehicles within the network (which is almost guaranteed, as noted by the authors). At the highest vehicle volumes, the average delay was found to be nearly 10 times as high; these numbers were even worse when considering higher percentages of human drivers. This work also investigates another ability that an intersection controlling agent must have in a real world scenario - accident detection. It is found that if the controlling agent has no means of detecting incidents within the intersection, then a large number of incidents will occur in the time following the original incident, as vehicles who assume the intersection will be clear collide with the stopped vehicles. After enabling the intersection to instantaneously detect an incident and assuming that all vehicles can be made aware of this incident, the number of resulting crashes in the 60 seconds after the initial crash was reduced from approximately 90 to 1.77 (using 6 lane roads). It is interesting to note, that with 5% human drivers included in the simulation, the incident rate was further lowered to 1.50 crashes over the 60 second interval as the human drivers do not assume the intersection will be clear of other vehicles. It is also shown that a delay in detecting the incident of only 5 seconds can increase the number of crashed vehicles during the interval to over 3.5. Furthermore, this approach relies on information such as

Table 3.2: Advantages and disadvantages of using a reservation-based approach for intelligent traffic signal control

Advantages	Disadvantages
Control decisions made in real-time. Distributed control allows easy scalability. Shown to be very effective, assuming everything goes as planned.	The model itself is somewhat infeasible. Requires agent-controlled vehicles and an extremely reliable communication network. Reservations may be missed frequently due to traffic dynamics. Shown to be ineffective without extremely high adoption rate. Has not been tested in realistic network settings.

maximum acceleration/deceleration of vehicles to calculate space/time requirements of vehicles. While this is easily done in a simulation environment, in the real world these values can vary based on a large number of difficult to determine factors such as weather conditions, tire conditions/choice and brake quality. Thus, it would seem that while this reservation approach may be extremely effective at controlling traffic flows in real time, there needs to be much more work done in the area of intelligent vehicle control before a real-world implementation could be considered safe or effective.

3.4 Market-based Control

Market-based control (Wellman, 1993) is an approach which views a system as a virtual marketplace, with economic agents interacting amongst each other. While Dresner (2006) and Dresner and Stone (2006) proposed a market-based extension to the reservation controlled intersection architecture, the first real work on market-based intersection control was completed by Balan and Luke (2006). Intersection control agents within this initial work make decisions based on the number of 'credits' that the waiting drivers hold. Each driver agent begins with a fixed number of credits and also receives credits while waiting at a red light. When a vehicle crosses through the intersection, it must pay a fixed number of credits. The combination of this

credit gaining/losing system, coupled with the fact that intersection control agents make decisions based on the number of credits each waiting vehicle possesses, has several beneficial properties:

- The intersection will generally favour incoming edges with more vehicles, as more vehicles results in larger credit totals.
- There is a level of fairness involved, as vehicles who have waited a long time previously (gaining credits) will be more likely to travel through the intersection.
- Emergency and other special vehicles can be given priority by augmenting the number of credits they have.
- It allows for a further extension in which drivers who are in a hurry can sacrifice credits they have stored to get through the network faster.

Experiments were carried out on a small traffic network (4x4 grid) where the method of assigning values to vehicles based on credits was compared to the following methods:

Counting Cars One point per vehicle in the queue.

In-Range Time One point per vehicle for each second that vehicle is within the sensor range of the intersection.

Mean Waiting Time A vehicle's points are equal to the average time spent waiting at all intersections during the current trip (including the current intersection).

Previous Mean Waiting Time A vehicle's points are equal to the average time spent waiting at all intersections during the current trip (not including the current intersection).

Initially, the main goal of the research was to develop a control scheme which was fair, by decreasing the variance of delay experienced by vehicles. While this goal was realized (the credit system resulted in lower variance than the other methods), the proposed method was also capable of controlling intersections more efficiently than the other methods. The credit system, however, was not compared to any other traffic control methods, thus it is difficult to comment on the method's overall effectiveness.

Vasirani and Ossowski (2009) combined both reservation and market-based control within the traffic domain. Unlike the work by Balan and Luke (2006), the intersection agents attempt to maximize profit by setting the cost of a reservation based on the level of demand. It is assumed that there is communication enabled between intersections and that all driver agents are aware of the current price of a reservation at all intersections. This work also assumes that real currency is used to pay for intersection use (an idea many may find undesirable) and that a secure payment method is available for the driver agents. The proposed system could, however, easily be applied using a virtual currency as is done in Balan and Luke (2006). A driver agent can request a reservation from an intersection and in the event it is approved, must transfer part of the cost in advance to secure the reservation. The driver agent then pays the remainder of the cost when it meets the reservation, or loses the initial fee if it is late or cancels the reservation. If a vehicle happens to arrive at an intersection without a reservation (either because it failed to make one, or was late), it must wait for a free reservation. The intersection manager will provide priority to any paying driver agents, but must grant a reservation to a driver who has been waiting a specified amount of time. Driver decision making, as far as routing is concerned, is based on a weighted sum of travel time and cost. This allows each driver to find a balance between how long it will take them to complete their trip and how much that trip will cost. An emergent feature of this strategy is dynamic traffic volume balancing caused by variable intersection costs. Figure 3.2 provides an example network with two available routes between points A and B. Intersections along the most direct route between the two points have, due to high demand, been assigned a cost of $10 each (for a total trip cost of $50). The slightly longer route, which has experienced less demand, has a cost of $5 per intersection for a total cost of $35. Drivers will be motivated by these price differences to select the slightly longer route, as the time lost begins to be outweighed by the money gained. One aspect of these systems that remains unclear is that of vehicle queueing. There is no discussion included regarding the obvious problem of a vehicle wishing to pay and proceed through the intersection being physically behind a vehicle that does not wish to pay at the moment. The paying vehicle must (in theory) travel through the intersection during its reservation, but may not be able

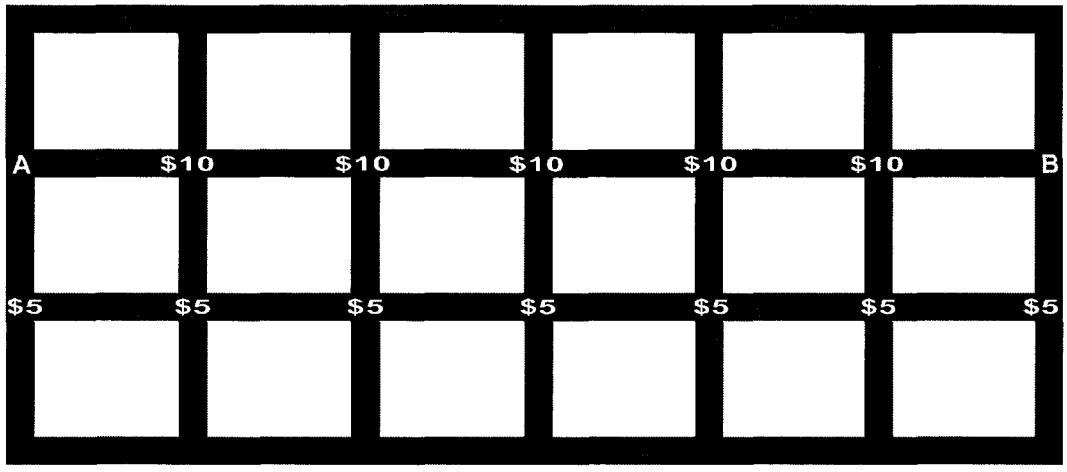


Figure 3.2: An example network with different reservation costs at different intersections

Table 3.3: Advantages and disadvantages of using a market-based approach for intelligent traffic signal control

Advantages	Disadvantages
Real-time and distributed decision making. Can create a fair control system. Can allow signal decisions based on driver desires (e.g., relaxed, hurrying).	Conflicting agent opinions (e.g., a hurrying driving behind a relaxed driver). Requires secure and reliable communication for transferring currency (real or fake). Has not been tested using realistic scenarios.

to do so due to the vehicle preceding it. This situation also violates the properties of a market, in which the highest bidder should travel through the intersection first. To investigate the properties of this model, the authors used a mesoscopic simulation of an area of Madrid, although the traffic volumes were theoretical. It was found that varying the cost of reservations at the different intersections can help balance traffic flow and decrease overall travel time. As previously mentioned, this balancing of traffic volumes is realized because intersections which are not busy will decrease the cost of a reservation, making it more likely a vehicle will travel that route to save money.

3.5 Self-organizing Systems

Gershenson (2005) developed a simple, reactive self-organizing system which was capable of controlling traffic signals within a network. The developed control system was evaluated in the NetLogo (Wilensky, 1999) modelling environment using the included Traffic Grid (Wilensky, 2003) model. The algorithm and model presented by Gershenson inspired the initial work of this thesis, which will be presented in Chapter 4. This simple model of a grid network allows the user to define the volume of cars (the total number of vehicles in the network) and uses simple rules to control the behaviour of the vehicles within the model (accelerating if they can, stopping if they must). The initial method outlined in this work, named Sotl-Request, involved a single agent maintaining a counter at each intersection. At every discrete timestep, this counter is incremented by the number of cars approaching the red light of the intersection. Once the counter reaches a specific threshold value (determined by trial-and-error in this case, as were all system parameters), the light at the intersection is switched. Using this method, the signals at the intersection are completely self-controlled and manage to switch to allow incoming vehicles (or those waiting) to pass through. The counter at each intersection increases as vehicles drive toward the intersection, which allows this approach to automatically coordinate and offset lights throughout the network. The main problem with the counter-based approach is that a large number of vehicles in the network can result in rapid switching of the lights, which leads to high levels of inefficiency. For this reason, the second approach outlined (Sotl-Phase) added a minimum amount of time that must pass before switching lights at an intersection. This allows the system to be controlled effectively, even at higher volumes. While the Sotl-Phase control method was capable of controlling the signals well, the third method (Sotl-Platoon) added further restrictions to the system in an attempt to keep groups of vehicles travelling together. In Sotl-Platoon, when an agent wishes to switch the lights at its intersection, it will not do so if there is an approaching vehicle within a certain distance of the intersection (this rule is ignored if there is a number of cars greater than another parameter value, as these vehicles will still remain as a platoon). Cools et al. (2008) extended this work in three ways. First, the authors implemented the algorithm in a more realistic traffic simulator

(MoreVTS, 2011), as the NetLogo Traffic Grid model provides only very basic traffic simulation. Second, a 12 intersection stretch of main road with incoming side roads was modelled to duplicate an area of Brussels. Finally, the algorithm proposed was evaluated against the plans used by the traffic authority for the area. These results show the proposed algorithm, on average, resulted in 50% less trip waiting time than the plans currently being used to control the network.

A much more analytical approach to self-adaptive traffic control was taken by Lämmer and Helbing (2008). Agents at each intersection in this approach use either of two possible strategies (depending on the local traffic situation). The first, which is used when traffic flow volumes are below saturation, determines optimal times to switch the lights at an intersection. This is achieved by taking into account the arrival and departure rates at the intersection, which allows waiting times to be predicted for each flow in the event they are stopped. Using these waiting time estimates and taking the time to switch lights into consideration, the decision to either switch lights or leave the lights can be made at any point. If the length of an incoming queue becomes longer than a specified value, the agent switches to a stabilization strategy. While using the stabilization strategy, an intersection maintains a list of incoming queues which are longer than should be allowed, changing the lights to clear these queues one after another. Simulations were performed, on a single intersection and an irregular 9x9 grid network, to compare the combined strategy approach to a fixed cycle approach, as well as both the optimization and stabilization strategies by themselves. The simulation results showed that the combined strategy was much more successful at keeping small queue lengths. Also, it was shown that neither of the two strategies worked effectively on their own.

3.6 Swarm Intelligence

An interesting area of artificial intelligence which authors have begun to apply to traffic light control is swarm intelligence, which relies on the collective intelligence of a group of simple agents (Bonabeau et al., 1999). de Oliveira et al. (2004) and de Oliveira and Bazzan (2007) applied the swarm intelligence technique of task allocation (Theraulaz et al., 1998) to the traffic control problem, with agents capable

Table 3.4: Advantages and disadvantages of using a self-organizing system for intelligent traffic signal control

Advantages	Disadvantages
Distributed. Adaptive. Typically rely on simple mechanisms. Local communication allows for co-operation and coordination between agents. Generally produce robust solutions. Effectiveness has been demonstrated on realistic traffic scenarios.	Generally require a communications network (although it can be simple due to local communication). Relatively little work completed in the area.

of selecting effective plans and coordinating with each other. Using this approach, a single agent at each intersection is used to select from 2 possible signal plans (one consisting of 40 seconds of green time in the N/S direction and 20 seconds in the E/W direction, the other with 20 seconds N/S and 40 seconds E/W). Each agent has a specific propensity (which varies over time) to implement each of the two plans. As is common in many swarm intelligence approaches, agents are also motivated to select a plan by pheromones (these pheromones act as an environmental stimulus for real-world social insects). Vehicles which are waiting at a red light in the traffic network release pheromones, which increases the motivation of the agent to select a plan which favours that edge (more green time dedicated to the edge's flow). As in the real-world, pheromone levels also dissipate over time which decreases the motivation if vehicles are no longer present. Unlike the real-world, however, a simulation environment provides an easy way of representing pheromones (using state variables within a computer program). Within this work, there is no discussion on how these pheromones may be represented in a real-world system and thus the feasibility of such a system is questionable. The overall motivation of an agent within this system to chose a signal plan is based on three criteria:

1. The concentration of pheromone on incoming lanes

2. The amount of green time allocated to each of the incoming lanes by the signal plan

3. The plans that neighbouring intersections are implementing

Coordination between agents is achieved through the consideration of neighbours plans (which are available through the use of local communication) when calculating motivation to select a plan. The motivation for a plan is increased by the percentage of neighbours also implementing that plan, which makes an agent more likely to select a plan that its neighbours are also using (the level of this effect is controlled by a system parameter). The actual method of plan selection is based on typical task allocation algorithms and takes into account both level of motivation for each plan as well as the response threshold of the particular agent to each plan. After a plan is implemented for a period of time, the agent's response threshold for that plan is either decreased (more likely to be selected) or increased (less likely to be selected) based on the performance of that plan over the time period. Using this method, it is hoped that agents will lower their response threshold for effective signal plans, making it more likely those plans will be implemented in the future. This approach was evaluated on a small, 5x5 grid network, with incoming volumes being either 20 per minute in the N/S direction and 10 per minute in the E/W direction, or vice versa. These insertion rates were held for periods between 60-180 simulation minutes before switching to the opposite volume description. The results demonstrated that the agents did indeed cooperate, with groups of agents implementing the same plan based on the traffic volume. While the ability of agents to cooperate is shown, the actual ability of this system to effectively control traffic is not shown, as there is no comparison to other control approaches included.

A second approach using particle swarm optimization (Kennedy and Eberhart, 1995) is presented by García-Nieto et al. (2011). To investigate the ability of particle swarm optimization to find effective traffic signal plans, real-world sections of both Sevilla and Malaga, Spain were constructed in the SUMO (SUMO Traffic Simulator, 2011) traffic simulator using data available from OpenStreetMap (OpenStreetMap,

Table 3.5: Advantages and disadvantages of using swarm intelligence for intelligent traffic signal control

Advantages	Disadvantages
Distributed. Adaptive. Rely only on local information.	No real-world pheromone implementation. Used only on simple problems so far (no realistic traffic scenarios/networks). Certain swarm approaches can be slow to adapt due to the learning nature.

2011). The plans developed by the particle swarm optimization approach were compared to those found using random search, as well as plans developed using a deterministic algorithm included in the SUMO traffic simulator. It is found that the proposed approach achieved the best (lowest score) fitness on both of the test networks, with a fitness value less than half of the others for the Malaga network and less than 65% of the others for the Sevilla road network. While this approach has performed well in these simulations, an optimization-based approach such as this may be hindered by real-world traffic dynamics and/or the time limits imposed in a real-time control scenario.

3.7 Fuzzy Logic

The first application of fuzzy logic in the traffic signal control domain was proposed by Pappis and Mamdani (1977). This approach consisted of 3 input variables: time (e.g., very short, medium), recent arrivals at the green phase (e.g., few, many) and queue length at the red phase (e.g., none, small, larger). The rulebase, which was developed by trial-and-error in this case, produces a single output from these inputs: the extension time for the current phase. As is seen with the decision support systems described in Section 3.8, this rulebase remains static and does not adapt along with changing traffic parameters, which can lead to degraded performance over time as the system does not generate a predictive model of traffic. After running for a specific amount of time, the fuzzy logic controller takes the input values and checks the

rulebase to see how long the current phase should be extended. This process is repeated until it is decided that no extension should be given. The fuzzy controller approach presented was compared to a vehicle-actuated controller on a simple single intersection network for various fixed traffic volumes. Using the fuzzy logic controller resulted in a 10-21% decrease in overall delay. Chou and Teng (2002) also proposed an extension-based fuzzy controller, using only the queue lengths of the 4 incoming edges as input variables. The controller was shown, through simulation, to perform much better than a fixed-timed approach to signal control.

As opposed to using the extension principle, Chiu and Chand (1993a) and Chiu and Chand (1993b) designed a controller which adjusts phase split, offset and cycle time (using a different rulebase for each of these calculations) based on the saturation levels of the network's edges. After each cycle, an adjustment is made to both the phase splits and cycle length based on the saturation levels of the E/W traffic flow and the N/S traffic flow. This is done by calculating a fraction of the current cycle time for each flow based on the inputs. The rulebase for modifying offset values relies on the volume difference between nodes, as well as the estimated change required to produce a 'green wave' on the edge with the highest saturation. The effects of this controller were investigated in a simulation on a 3x3 grid network with vehicle insertion rates varying for each incoming edge. Initially, the signals in the network would be controlled using a fixed signal plan with equal time in each direction. After 30 simulated minutes, 3 intersections would be selected for adaptation, resulting in approximately 66% as much delay as the fixed plan. After another 30 simulated minutes, all intersections would be allowed to use the fuzzy controller, resulting in around half the delay of the fixed plan.

Hoyer and Jumar (1994) developed a more complex fuzzy controller which can deal with a varying number of phases. The fuzzy controller also included phase selection as an output, allowing dynamic phase ordering based on real-time traffic data. Inputs used in the fuzzy controller included current traffic volumes on the incoming lanes, as well as the elapsed time since the last phase change. The controller was compared to four other controllers (including a fixed time controller, two vehicle actuated approaches, as well as a simple fuzzy logic controller with 2 phases and

a fixed cycle time) on a 3 intersection network with different traffic volumes and turning rates. With low volume of vehicles, all but the fixed controller maintained a similar level of travel time. At higher volumes though, the proposed approach seems to adapt better than the simple fuzzy controller and the vehicle actuated approaches. Similar to this, Lee and Lee-Kwang (1999) created a controller which optimized phase length and order, but also considered the state at neighbouring junctions. This was accomplished by adding input variables to describe the state of congestion at the neighbouring junctions, as well as the congestion between the intersection and its neighbours. Lee and Lee-Kwang (1999) compared the developed controller to a vehicle actuated controller on 3 grid-like (with minor mutations) networks, with both fixed and steadily increasing traffic levels. Improvements of 3.5-8.4% were found with fixed traffic volumes, while 4.3-13.5% was found with the increasing flows.

Similar to several of the previous approaches outlined, the FUSICO project (Niittymaki and Pursula, 2000) developed a fuzzy controller that included both a fuzzy extender for determining the time to switch phases, as well as a fuzzy selector for selecting phases. The controller was shown to outperform a simple extension-based control method for volumes of 400-1500vph (vehicles per hour), while the extension controller performed better on volumes less than 400vph. When speaking about the rulebase of FUSICO, the authors state that rulebase creation became complicated when the number of possible phases increased from 3 to 4. This highlights a key concern of fuzzy control design: how to design an effective set of rules. If 4 phases can cause difficulty, the problem will be further compounded when working with a complex intersection with twice this number of phases. A similar approach was presented in Murat and Gedizlioglu (2005) and compared to those presented in Niittymaki and Pursula (2000) and Pappis and Mamdani (1977). The new approach, however, does not improve upon the two previous solutions.

While the fuzzy control approach was much the same as seen in other proposals, Wei et al. (2001) proposed using a genetic algorithm to optimize parameters of the fuzzy controller to improve functionality; however, little detail was given on the implementation or effectiveness of this idea. A genetic algorithm was effectively used in Heung et al. (2005) for rule generation in a fuzzy traffic controller. A set of fuzzy

Table 3.6: Advantages and disadvantages of using fuzzy logic for intelligent traffic signal control

Advantages	Disadvantages
Simple. Generally rely on easy to obtain observations. Controls in real time. Distributed.	Difficult to extend to complex intersection logics. Static/Non-learning - fail to adapt to changing traffic demands. Tested on small networks only. Commonly requires expert knowledge or trial-and-error to produce rulebases. Have not been tested within a realistic environment.

rules was maintained for offline evaluation, in which the effectiveness of different rule combinations was analyzed. If an appropriate performance increase was found regularly while using a specific rule, that rule would be promoted to the knowledge-base and used within the fuzzy controller. Using the optimized rulebase, the authors were able to realize small improvement gains over a non-optimized set of rules.

3.8 Decision Support Systems

The main goal of a decision support system is to propose possible effective signal plans to human operators, making the human operator more efficient and informed. While this is not exactly intelligent traffic control, simply bypassing the human operator and implementing the best plan would allow these systems to control a traffic network (it is for this reason that these types of systems are included here). Typically, decision support systems rely on a pre-created list of traffic scenarios and possible control actions (along with the estimated effectiveness of these control actions). These lists are created using human traffic experts and historical traffic data, which can be very time consuming. Also, relying on this type of list means the system can be incapable of addressing unexpected problems that are not captured within the database. Performance may degrade significantly then, if the system can not make proper decisions due to the limitations of the scenarios list. Also, these types of systems can suffer

in the same way that fixed traffic plans do as the traffic dynamics change and the original assumptions become invalid.

One of the first researchers to propose a traffic decision support system was Cuena (1995). This work divided the entire network into subnetworks of a manageable size, with a single agent used to control each subnetwork. These agents were each aware of some key information: the network design, possible conflicts and possible control actions.

Similar to the above mentioned approach, Hegyi et al. (2001) and De Schutter et al. (2003) implemented a decision support system which uses fuzzy logic to classify traffic states. Once again, the entire network is broken into smaller subnetworks, each of which has a specific casebase that is generated offline. Each case within these casebases represents a specific traffic situation, control measure and predicted effect. In the event of an unrecognized traffic state, fuzzy logic is used to compare the observed state to the states described in the casebase. Based on the level of similarity the observed state has with various cases within the casebase, the effectiveness of the control actions outlined by the similar cases can be predicted. The action with the highest predicted effectiveness is then implemented by the control devices within the subnetwork. In the later work, prediction of traffic flows in and out of the subnetwork over a period of time is also included. This prediction allows for the effect of other intersections' decisions to be considered automatically, eliminating the need for a coordinating agent. The performance of the system, however, could severely degrade if the actual state is not close to any of the states specified within the casebase, as fuzzy logic will not be able to select a similar case.

A fuzzy neural network based decision support system was described by Almejalli et al. (2007a, 2008). A flowchart representing the operation of this system is included in Figure 3.3. As with the work completed by Hegyi et al. (2001) and De Schutter et al. (2003), the most important piece of information in this system is the set of possible control actions available for each traffic situation. From the diagram, it can be seen that the set of states and control actions are generated from experiential human knowledge coupled with historical traffic data and available traffic control devices, which can be tedious and problematic in the long term. Not only does the

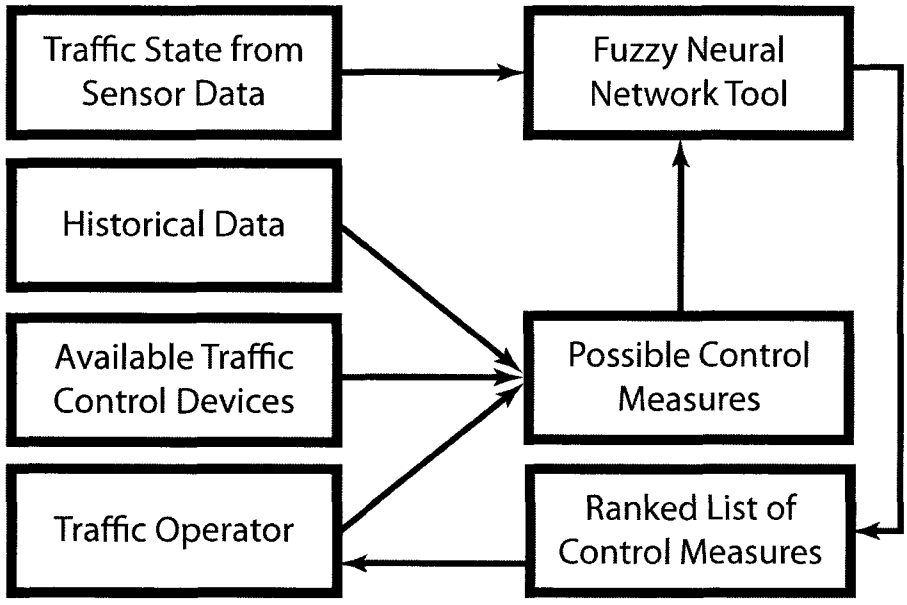


Figure 3.3: Flowchart for the system developed by Almejalli et al. (2007a, 2008)

list of states and control actions need modified as traffic parameters change over time, but the list also would not apply if the topology of the network were to change (e.g., a long term construction project closes several lanes). The fuzzy neural network (which has a structure similar to that shown in Figure 3.5) takes both traffic conditions (available from sensor data) and control actions as inputs. The first step involves converting these inputs into fuzzy membership values (e.g., traffic density is high, medium or low). From these values, there are a number of neurons which each represent a single fuzzy rule (the fuzzy rules themselves are identified using a GA, as proposed in Almejalli et al. (2007b)). Each of these fuzzy rules considers a set of possible traffic conditions and a possible control action, outputting expected traffic measurements (such as waiting time will be low). These fuzzy outputs are then defuzzified into exact fitness numbers, which allow each possible control action to be ranked against the other actions. The effectiveness of this system to predict a traffic situation after applying a control measure is shown by comparing the expected results to those found from a simulation. The results found that the predicted traffic state very closely resembled those found through simulation.

Table 3.7: Advantages and disadvantages of using a decision support system for intelligent traffic signal control

Advantages	Disadvantages
Have been implemented successfully using real-world networks. Assist human traffic operators in making informed decisions. Have been tested using realistic traffic scenarios.	Rely on a centralized architecture. Have not been used to make real signal plan decisions. Use a large amount of data which is difficult to generate and is non-adaptive.

In Almejalli et al. (2009), the aforementioned support system is extended into a hierarchical framework (shown in Figure 3.4), with a coordinator agent (CA) being used to decide globally effective sets of actions based on the suggestions of a number of agents, each of which implements the fuzzy neural network previously mentioned. For each subnetwork, the coordinator agent maintains a table (the CA table) which details the possible effects any control action implemented within that subnetwork may have on all other subnetworks. Agents controlling each subnetwork then use the decision support system developed by Almejalli et al. (2007a, 2008) to propose a ranked list of control actions. Using the CA table, the coordinator can predict the effectiveness of a global set of control actions using the ranked lists of actions provided by the lower-level agents. Through a case study consisting of 3 subnetworks within a realistic model of a section of Riyadh, Saudi Arabia, the authors showed that the coordinator agent was able to consider the effects of the actions on the other subnetworks and determine which action each subnetwork should take.

3.9 Reinforcement Learning

Inspired by behavioural psychology, reinforcement learning (Kaelbling et al., 1996) is a machine learning approach which allows agents to interact with the environment, attempting to learn the optimal behaviour based on the feedback received from interactions. This typically involves breaking the environment into states, from which each agent can select a possible action. The reward gained from taking an action within a state determines the level of reinforcement, which in turn affects the likelihood

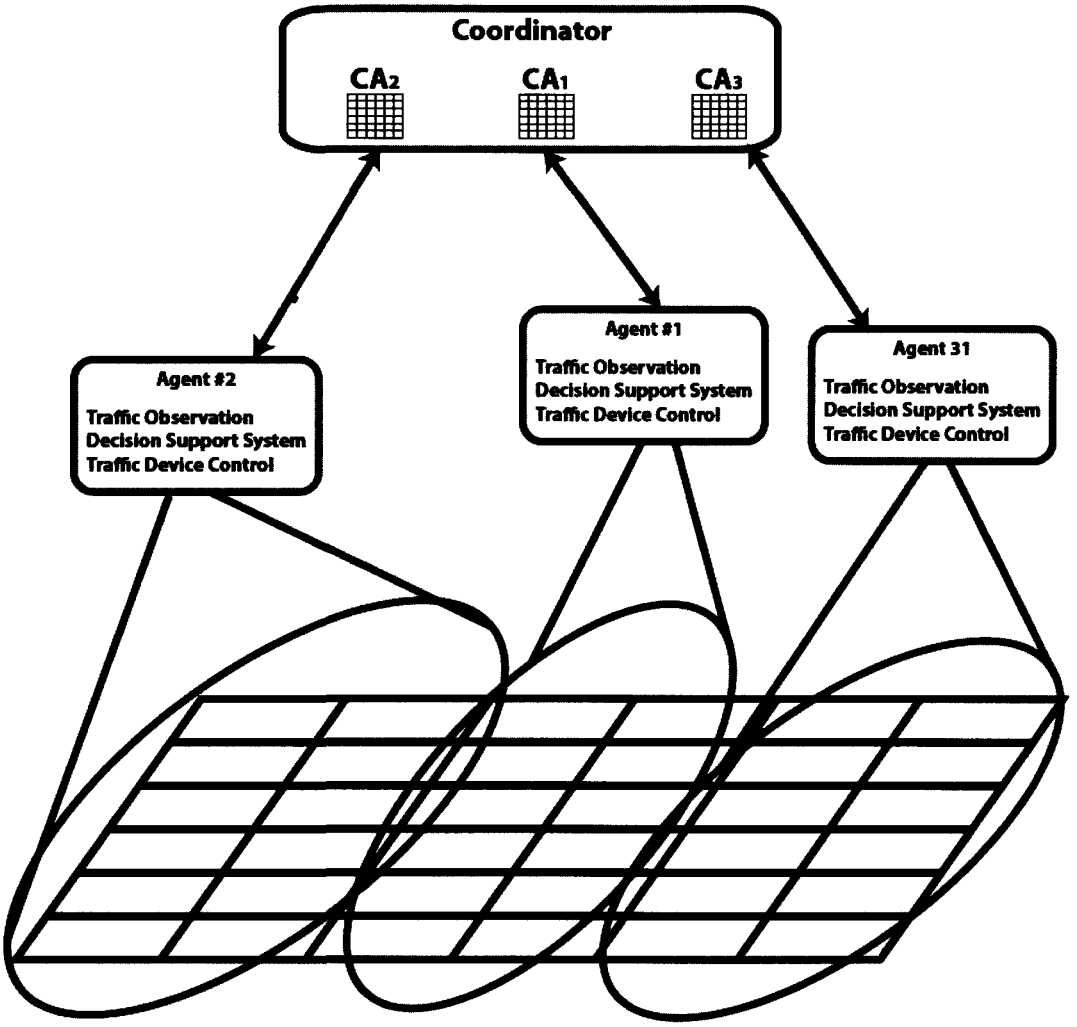


Figure 3.4: Architecture of the hierarchical decision support system implemented by Almejalli et al. (2009)

that the agent will select that action when it is next in that state. As agents in the network monitor the traffic situation, they can identify problems as they occur and select possible control actions to alleviate these problems. Since the actions of each agent can affect the state in other subnetworks, a coordinator agent is included which is capable of determining the compatibility of the differing control actions suggested by the agents. This coordinator also has a notion of agent priority, so it is capable of deciding which control actions (from those proposed) should be implemented by the various agents.

Wiering (2000) and Wiering et al. (2003, 2004) presented one of the first reinforcement learning approaches to traffic signal control. These works assume that the traffic environment can be represented as a Markov Decision Process (Puterman, 1994), which can be defined as $M =< S, A, P, R >$, where S is a set of states, A is a set of actions, P is a transition function which determines the probability of moving to the next state when choosing a specific action from a specific state and R is a reward function which assigns a reinforcement value to each state/action/future state pairing. While the P and R functions are not known a priori, they are inferred from the experience of an agent as it interacts with the environment. It is unclear whether it is valid to assume that a Markov Decision Process models traffic correctly. While it is possible to predict a resulting traffic state from a state/action pair, this can certainly be difficult in a large network with a large amount of interactions. Also, the number of possible states within a large network can be extremely large, which may make implementation of an MDP traffic model difficult. Within an MDP environment, however, each agent interacts with the environment, receiving rewards based on the actions they choose in various states. These rewards are used to induce a model which represents the expected gain when choosing an action in a certain state. Eventually, these values converge (if the assumptions hold true), at which point the maximum expected reward can be chosen by the agent each time. The traffic domain, however, involves constantly changing traffic distributions, which hinder the performance of reinforcement learning approaches designed to work in stationary environments. Two possibilities for learning are identified by the authors here: traffic light learning (which looks at the number of cars in each direction) and car-based learning (which

involves summing the estimated rewards for each waiting vehicle). The authors chose the car-based learning approach (which makes traffic light decisions based on the estimated rewards for each vehicle in the queues) noting that it can be difficult to learn all possible situations a traffic node can experience. Through simulation, the learning approach outlined was compared to several simple traffic control mechanisms on a small 4x4 grid-like network and a small 'city-like' infrastructure. It was shown that the learning algorithms performed consistently better than the others on both network architectures, with an algorithm in which vehicles simultaneously learn better route choices as they progress through the network achieving the highest level of performance.

Steingröver et al. (2005) extend an approach similar to that described above by adding neighbouring junction congestion levels into the intersection state description. This addition can allow the intersections to learn how to cooperate with each other, but also increases the size of the state space. This new method was compared in simulation to the work shown in Wiering (2000) and Wiering et al. (2003, 2004) with both fixed and varying traffic volumes. With fixed volumes, the new method cut average trip waiting time approximately in half. The results were even more drastic when varying traffic volumes were included, with improvements ranging from around 50-75% less.

One problem with the learning processes described so far within this section is that they fail to adapt quickly in a non-stationary environment (where parameter values describing the traffic state change quickly). Since the traffic domain could certainly be considered non-stationary, de Oliveira et al. (2006) attempted to use reinforcement learning with context detection (da Silva et al., 2006) to address this problem (in this case, a context represents a specific traffic distribution). With this approach, multiple models of traffic are maintained, each of which has a learning system associated with it. Each learning system then, is responsible for learning how to optimally control traffic that matches (or at least closely resembles) its corresponding model. An error value can be calculated for each model, allowing the system to decide which model matches the current traffic state, with the best matches being used to control the signals within the network. If no model exists with an error below a specified threshold,

Table 3.8: Advantages and disadvantages of using reinforcement learning for intelligent traffic signal control

Advantages	Disadvantages
Shown to converge to best solution (assuming assumptions hold true). Effective at learning parameters that do not change over time.	Can make invalid assumptions. Fails to adapt quickly. Difficult to handle traffic with varying parameters using the same model. Large amount of states required for a large network. Trial-and-error can be an ineffective approach in a real world situation. Have not been tested on realistic traffic networks/volumes.

a new model will be created and the learning process will begin. The system begins with only a single model, adding new partial models as required by changing traffic parameters. Through simulation, it is shown that the context detection and multiple models allow this approach to perform better than a standard reinforcement learning approach when there is noise in the traffic volumes, as the traditional approach takes a period of time to relearn everything each time the volumes fluctuate.

3.10 Neural Networks

Neural networks (NN) were used in Spall and Chin (1994) to search for optimal signal plans. An entire day is divided into a number of intervals, with a single NN determining signal plans for a specific interval. An assumption is made by the authors that traffic situations in the same interval on varying days will be similar. When looking at long time intervals (e.g., the 3 hour weekday morning rush period), traffic levels are most likely quite similar from day to day, as a similar number of people travel to the same place at the same time. This assumption may not apply as well, however, outside of these typical rush-hour times and also may not apply within an interval itself. On a short-term basis (e.g., 5 minutes), interactions and dynamics can result in traffic volumes which vary from the average for the entire interval. This assumption then, may hold true in some cases, but may also be false in many others.

The NN then, trains on each day over the interval it controls. On a simple 3x2 grid network, using fixed insertion rates for the 2 main roads (high average volume) and 3 crossing roads (low average volume), it was shown that the NN control approach could decrease the total system waiting time by nearly 35 percent over 50 training days. This approach is investigated further in Spall and Chin (1997), using a 3x3 grid network. The volumes within the network are designed (using data from Rathi (1988)) to match those of the Manhattan area the network is based on. The neural network control was used for a 4 hour period over the evening rush hour, with one test consisting of constant volumes throughout the 90 day simulation period and one test increasing the volumes on all roads after 10 days. In the fixed volume case, the NN approach to control was shown to decrease waiting time by approximately 10% over a fixed timing scheme. It is also shown, however, that the NN approach takes many days (approximately 20 to return to the initial value) to adapt to the increased demand from the 10th day. This is because the neural network learns how to handle a single traffic distribution in a highly effective manner. When the traffic volumes change, the neural network must re-learn how to effectively control the traffic lights, which takes time and results in a period of low performance. This is a common problem that can be seen in machine learning approaches, as the system becomes over-specified for controlling traffic with specific parameters.

Wei and Zhang (2002) combined both a fuzzy logic and neural network approach to decide whether to extend a phase or not. An example of this type of control architecture can be found in Figure 3.5. In this approach, measures of traffic volume are initially used as input to a fuzzy neural network (layer #1 within Figure 3.5). This input is then converted into fuzzy values (layer #2) so the fuzzy rule base (layer #3) can be used. The output of the fuzzy rule layer is then defuzzified (layer #4) which determines whether the light should be extended or switched. After a minimum amount of time, the decision process is carried out every second until the light is either switched or a maximum amount of green time has occurred (at which point the lights change automatically). The performance of the fuzzy neural approach was compared to that of a simple extension-based approach for a single intersection and multiple volumes (with volumes remaining fixed for each simulation). Through simulation, it

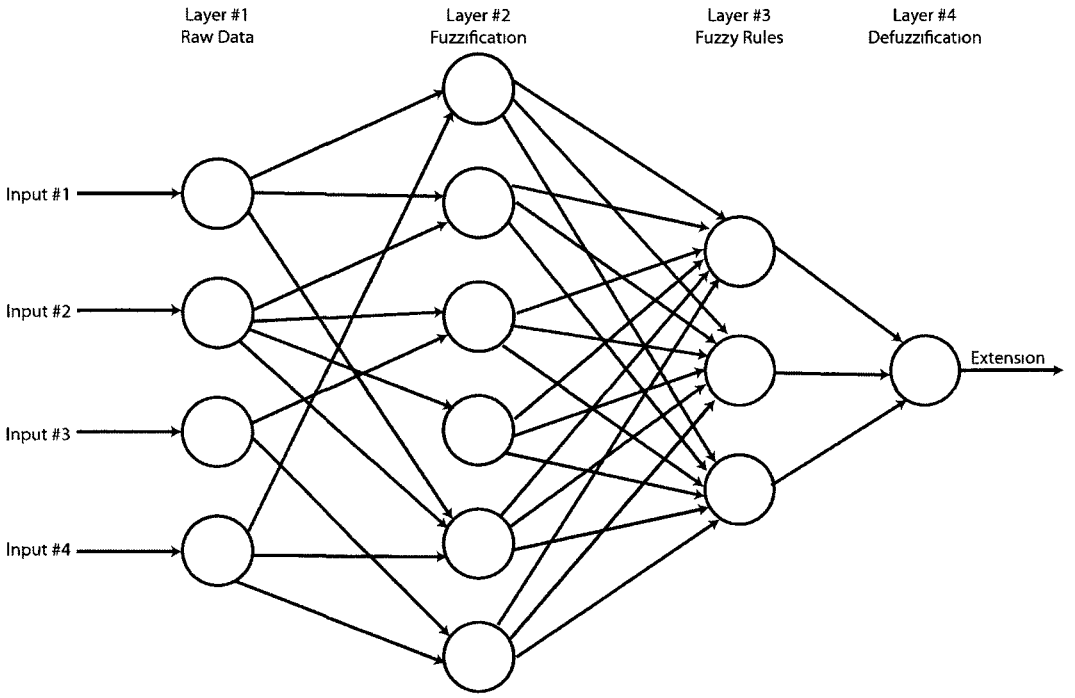


Figure 3.5: Example fuzzy neural network taking 4 inputs and determining phase extension

was shown that the fuzzy neural control results in 15-25% less stops as well as 15-30% less average delay.

The power of fuzzy neural networks were combined with a hierarchical architecture by Choy et al. (2003a,b). The hierarchical architecture divided the entire network into smaller subnetworks, each of which is managed by an intersection controller agent. These intersection controller agents observe traffic state and pass measurements to a higher-level zone controller agent. The zone controller agents use a fuzzy neural network (similar in structure to that used in the decision support system developed by Almejalli et al. (2009, 2008, 2007a)) to calculate a zone signal policy and a zone co-operation factor (which determines the level of cooperation required amongst agents). Once zone policies are proposed, a conflict resolution process takes place to ensure all policies that will be implemented are compatible. Online reinforcement learning is also employed to constantly check the effectiveness of the fuzzy rule base. This learning process is responsible for determining which neurons are connected within

Table 3.9: Advantages and disadvantages of using neural networks for intelligent traffic signal control

Advantages	Disadvantages
Like reinforcement learning, have been shown to converge to acceptable solutions under the right circumstances. Can control in real time. Can be distributed.	Adapt very slowly to changing traffic parameters. Some works require multiple models to be maintained for various times within a day. The inner-workings of neural networks are notoriously hard for humans to understand (e.g., why do we get the result we get). Have been tested on single realistic network only (with hypothetical volumes).

the inner layers of the neural network (thereby determining the possible fuzzy rule base). To evaluate the effectiveness of this system, a 25 intersection network was implemented using data provided from the Land Transport Authority (LTA) of Singapore and the results of the proposed algorithm were compared to those used by the LTA. Two scenarios were simulated using both control methods: one consisting of 3 hours of simulation with a single volume peak and one consisting of 6 hours with 2 volume peaks. From these simulations, it was noted that the overall mean vehicle delay was decreased by approximately 15% for the single peak scenario and nearly 30% for the double peak scenario. A further traffic scenario consisting of 24 hours of simulation with a total of 8 traffic peaks is presented in Srinivasan et al. (2006). The proposed controller maintained lower mean delays over all peak periods and also performed much more effectively on later peaks, where the LTA control scheme began to fail.

3.11 Summary

This chapter outlines a number of computational approaches that have been applied to the domain of traffic signal control, each of which has its own advantages and disadvantages (outlined within Tables 3.1-3.9) Through analysis of the works presented in this chapter, there are a number of requirements that the algorithms developed in the remainder of this work should meet. First, the algorithms should be able to adapt signal plans based on observed traffic state, without using historical data (which tends to be inaccurate, resulting in inefficient signal plans). Second, the control systems developed should be distributed, which will increase both the scalability and robustness of the system. Also, the controlling agents should rely solely on locally available data (whether observed through sensors or communicated by neighbours). As with distribution of computation, reliance on only local information increases the robustness of the system, as the system avoids the problems inherent with complex centralized communication and control networks. The algorithms should also be designed and evaluated on traffic scenarios that closely represent those found in the real-world. This will ensure that the algorithm is not only capable of solving simple traffic problems, but is also applicable to real situations.

In the following chapter, a simple algorithm is developed which is capable of adapting traffic signal plans in real-time using locally communicated information. This algorithm will be evaluated on a simple traffic model/network and its performance will be compared to other possible control methods. In a later chapter, the creation of an algorithm that meets all of the requirements mentioned above will be presented.

Chapter 4

A Simple Adaptive Algorithm for Traffic Control

4.1 Introduction

This chapter develops and evaluates a simple multi-agent traffic control algorithm within the NetLogo (Wilensky, 1999) modelling environment. Through simulation of various traffic distributions, it is shown that the algorithm's adaptive nature allows for successful signal control in a wide range of situations. The algorithm is also compared to several other simple control measures, with the results showing the adaptive approach is more effective overall than any of the other methods.

The chapter begins by describing the development of the traffic model within the NetLogo (Wilensky, 1999) modelling environment (Section 4.2). This includes a description of the original model (provided in the NetLogo modelling package), as well as changes that were made to allow the model to function as required. Section 4.4.1 begins the description of the control system by explaining the various system parameters included within the algorithms. This is followed by Section 4.4.2 which describes the main control algorithm which updates the state of the simulation after every timestep. Section 4.4.3 describes the algorithm that is used to generate signal plans based on locally observed data and information available from neighbouring intersections. The experimental setup used to evaluate the proposed algorithm is included in Section 4.5, with the results found through simulation presented in Section 4.6. The chapter ends with a summary (Section 4.7) of the algorithm and results, along with a short discussion of the work to be presented within the following chapters.

4.2 NetLogo and the Traffic Grid Model

To investigate the effectiveness of a distributed and adaptive traffic light control scheme, the NetLogo (Wilensky, 1999) modelling environment was used. NetLogo is

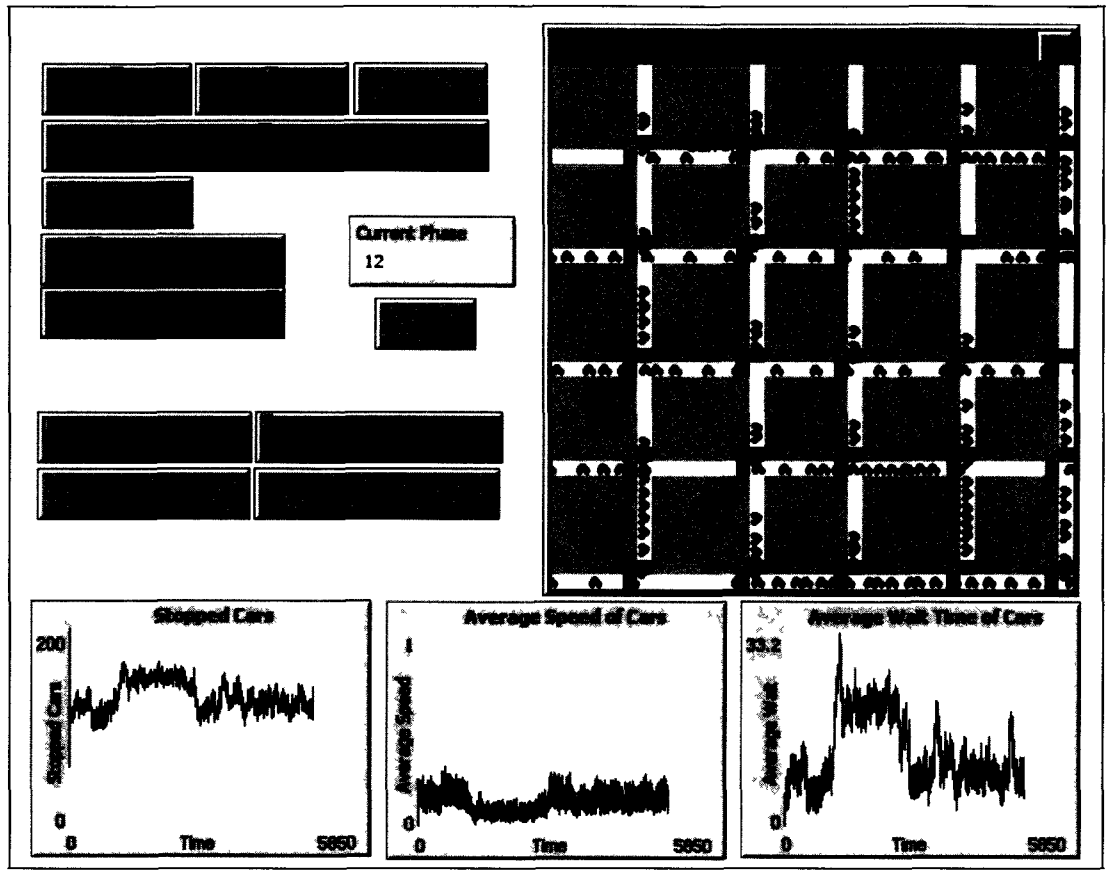


Figure 4.1: A screenshot of the initial NetLogo Traffic Grid model

a freely available multi-agent modeling environment which allows simple models to be created quickly and easily. The environment also offers an easy-to-use graphical interface where the user can observe the modeled system's operations, modify system parameters and create graphs of important data. Furthermore, NetLogo also provides many multi-agent models which can be used as the base for more advanced work, including the Traffic Grid (Wilensky, 2003) model used here. The same Traffic Grid model was also used in the traffic light control research completed by Gershenson (2005), which motivated its use here. Details on the operation of the Traffic Grid model are given below and an example of the model's user interface is provided in Figure 4.1.

4.2.1 Traffic Grid Driver Behaviour

The driver model used in the Traffic Grid model is extremely basic. First, vehicles only travel in one of the two possible directions (left to right or top to bottom), with no turning movements possible. Second, at each time step every vehicle either accelerates by a fixed amount (when no other vehicles are in front and no red light is present), sets their speed to slightly less than that of the vehicle in front of them (in the case they are travelling the same direction), or sets their speed to zero (when a crossing car is in front or a red light is present).

4.2.2 Traffic Grid Network

Like the driver behaviour, the traffic network within the Traffic Grid model is also very simple. Intersections are equally spaced throughout the landscape in a grid pattern. Intersections are connected to each other via one way roads, each of which is one car-width wide. The network is also toroidal in nature, so a vehicle which reaches the right or bottom edge of the landscape is automatically moved to the opposite side. Using this approach, the number of vehicles in the network remains fixed throughout the simulation. The initial (and final) number of vehicles within the network is specified as a system parameter using the provided interface.

4.2.3 Traffic Grid Signal Control

Since there are only 2 traffic flows and single lane roads, there are only two signals to be controlled at each intersection. In the initially provided model, the signals operate in a fixed-time pattern with all up/down lights being the same and all left/right lights being the opposite. The operation of these signals is one of the main changes made to the model, as described in Section 4.3

4.3 Traffic Grid Model Changes

While the Traffic Grid model provided with the NetLogo environment was a great starting point, two important changes were required to develop and evaluate the

adaptive traffic light control methods presented here. These changes are described in the next two subsections.

4.3.1 Vehicle Volumes

As described in Section 4.2.2, the vehicle volumes in the original Traffic Grid model remained constant. The first major change to this functionality was to remove the toroidal aspect of the traffic network. Instead, vehicles were inserted at the left or top end of each road and removed from the network when they reached the other side of the landscape. The second change made to the initial model was the inclusion of varying vehicle volumes. To achieve varying vehicle insertion rates, two variables were added to the model representing the expected number of cars inserted every minute (60 time steps) in either the left/right or up/down direction. At each time step, a vehicle might be inserted at the beginning of all left/right or all up/down roads. Whether to insert cars or not is decided probabilistically using the specified vehicle rates. The insertion rates for each direction followed one of many prespecified distributions, allowing an algorithm's adaptability and versatility to be investigated. A full description of the distributions used during evaluation is provided in Section 4.5.

4.3.2 Intersection Agent Behaviour

To allow for adaptive signal control, it is assumed that traffic signals no longer operate under a fixed timing schedule. Instead, an intelligent agent responsible for controlling a single set of signals is located at each intersection within the network. Each of these agents possess several key characteristics:

- Local Observation - Each controlling agent is capable of making local observations of traffic state. It is assumed that each agent is aware of the number of vehicles on approaching road segments at any given time. The observations of each agent, however, are limited to the incoming roads of that agent's intersection. There is no notion of global traffic state included within the model.

- Local Communication - Each agent is capable of communication with neighbouring intersections. Once again, this communication is limited to neighbours which are only one road segment away. No limitation is placed on the amount of communication that can be performed, however, the data communicated in this work is limited to the signal plan details of upstream intersection agents. It is also assumed that this communication framework would be used to generate the traffic state observations explained above. An example of how information flows within the network using communication is shown in Figure 4.2.

- Time Window - An agent stores observations for a specific amount of time. After this period of time has passed, older observations are forgotten completely.

4.4 System Control

4.4.1 System Parameters

There are several parameters which can affect the overall performance of the control algorithm.

- Window Length (WL): This represents the life span of traffic state observations. For example, if the window length parameter is 180, then there will be an observation stored for each of the last 3 simulation minutes. Observations older than this period are forgotten.

- Update Interval (UI): This determines how long agents should operate until they update their signal plan. When this number of simulation steps (seconds) have elapsed, each agent will calculate a new signal plan and begin to implement it immediately. For this reason, the update interval is generally a multiple of the cycle length.

- Current Time Weight (CTW): Initial observations found that agents often overcompensated for extremely recent changes in traffic flow, changing signals in a way that did not match the underlying traffic distribution. For this reason, the current time weight parameter was added to the algorithm. As can be seen from the algorithm detailed in Section 4.4.3, this parameter is used to force the

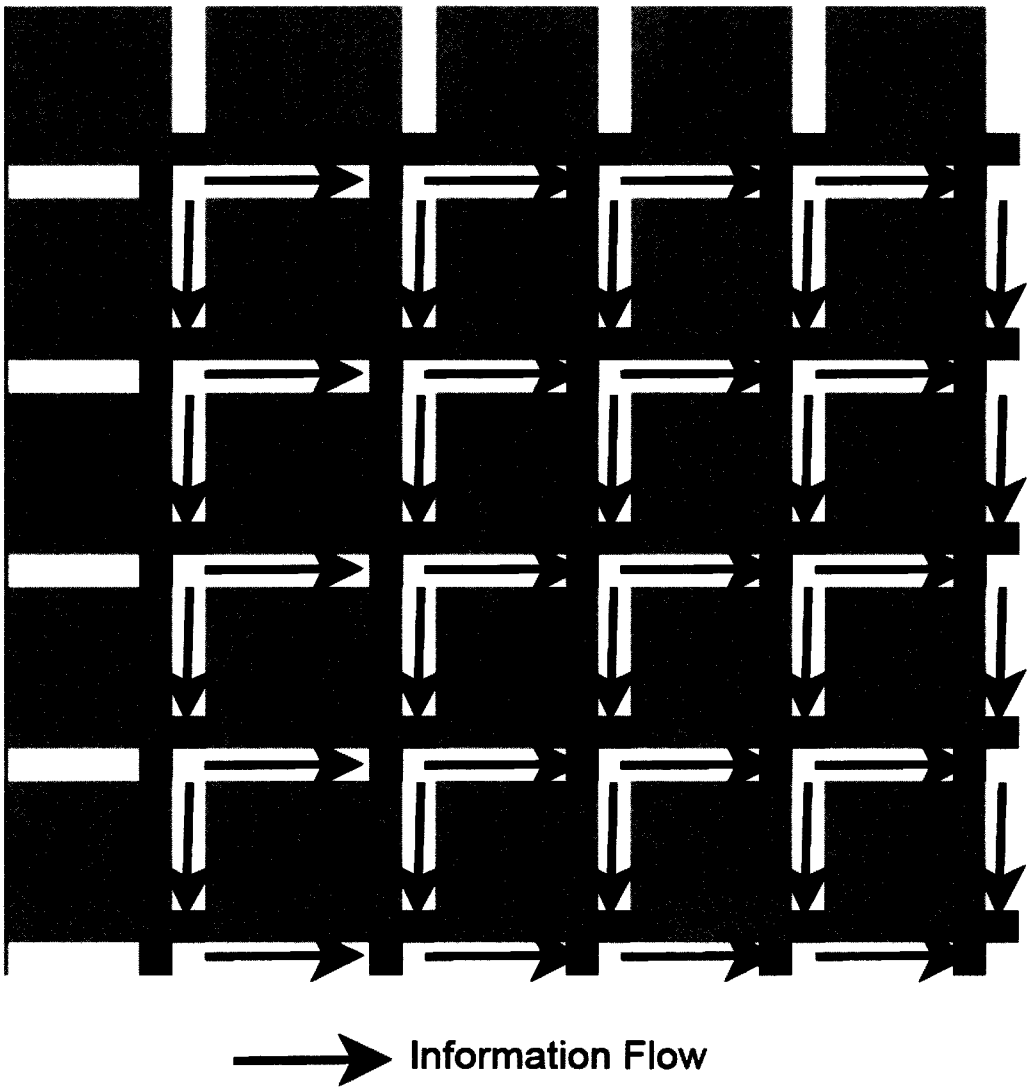


Figure 4.2: Information flow through the NetLogo network model

agent to balance between the plan it had been implementing and the plan it now wishes to implement. In other words, the CTW parameter adds inertia to the system. This stops the signal plans from fluctuating wildly and allows the agent to match the underlying traffic distribution more accurately.

- Neighbour Weight (NW): Several previous works on traffic control, including that by France and Ghorbani (2003), mention intersection coordination as an important aspect in traffic signal control. In this case, it was thought that if an agent coordinated with upstream neighbours on plan selection, that the downstream agent could select a plan that will better meet the short-term traffic requirements. This is because upstream neighbours will have considered the amount of vehicles present at their intersections when calculating their own signal plan and these vehicles will continue downstream when they receive a green light. To a certain extent then, agents can predict future traffic state by considering the decisions being made by upstream controlling agents. This is the main motivation behind the inclusion of the neighbour weight parameter in the controlling algorithm.

- Vehicle Insertion Rates (NS_Rate and WE_Rate): These rates determine the expected number of vehicles to be inserted onto the roads every minute. Due to the probabilistic insertion of vehicles, the actual number of vehicles inserted will vary. The insertion rates can also be modified throughout a simulation, allowing varying vehicle volumes, which demonstrate the adaptability and robustness of the control system.

- Cycle Length (CL): The global cycle length for all intersections within the model. While this value can be modified throughout a simulation, it remained constant (at 60 simulation steps) within this work.

- Minimum Green Time (MGT): This is the minimum amount of time a light must be in the green state during each cycle. This value is used to ensure that no approaches are starved for time which would result in vehicles waiting an unacceptable amount of time to pass through an intersection.

4.4.2 Model Control Algorithm

All entities within the model must be updated after every timestep. The algorithm specifying the actions taken for each timestep is shown in Algorithm 1. Lines 1-10 of the algorithm include details on how vehicle state (including location and speed) is updated each timestep. The actions performed by each intersection are shown in lines 11-19, including the possible signal plan update (line 17) which is presented in more detail within Algorithm 2. These steps involve the removal of observations that are now outside of the time window, as well as the addition of new observations based on the current traffic situation. Finally, the probabilistic insertion of vehicles (lines 20-27) is completed in such a way that, on average, NS_Rate and WE_Rate vehicles will be inserted every minute onto the N/S and W/E roads respectively.

4.4.3 Signal Plan Calculation

Algorithm 2 details the signal plan update process used by intersection agents within the NetLogo model. Since only two phases exist for each traffic signal within this simple model, only a single phase length needs to be calculated to determine the lengths of both (with the remaining cycle time being devoted to the other phase). The majority of the algorithm presents the steps for calculating the amount of green time for the West direction, with only the last line being devoted to determining the green length for the North direction. The first step of the algorithm (lines 1-3) requires the calculation of average volumes over the time window for both directions (AV_W and AV_N), as well as the proportion of this volume that was found in the lane approaching from the West (P_W). Using this proportion, an initial green time can be calculated by supplying the West direction with that proportion of the entire cycle (line 4).

After an initial green length has been calculated, the neighbour weight parameter can be applied to modify the green time to be more like those of the agent's upstream neighbours (line 5 of the algorithm). Here, the initial green time is decreased to a percentage of the original (determined by the value of the NW parameter) and increased by a value representing the length of green time in the west direction that

```
  /* Vehicle location and speed update */
1 foreach Vehicle do
2     if At red light or vehicle ahead travelling in opposite direction then
3         Speed = 0
4     else if Vehicle ahead travelling in same direction then
5         Speed = max(0, Speed of vehicle ahead - Acceleration)
6     else
7         Speed = min(Speed Limit, Speed + Acceleration)
8     end
9     Position = Position + Speed
10 end
  /* Update of intersection state */
11 foreach Intersection i do
12     Remove oldest observation from West_Observations
13     Remove oldest observation from North_Observations
14     Add current amount of vehicles on west approaching lane to
       West_Observations
15     Add current amount of vehicles on north approaching lane to
       North_Observations
16     if UI steps have passed since update signal plan update then
17         UpdateSignalPlan(i)
18     end
19 end
  /* Probabilistic vehicle insertion */
20 Num = Random number from 0-59
21 if Num < NS_Rate then
22     Insert vehicles at beginning of all N/S roads
23 end
24 Num = Random number from 0-59
25 if Num < WE_Rate then
26     Insert vehicles at beginning of all W/E roads
27 end
```

Algorithm 1: Control Loop for NetLogo Simulation

upstream neighbours are currently implementing (available through local communication). The set X used within the calculation on line 5 is the set of all upstream neighbours. Generally the size of X is 2, but boundary intersections can have 1 or 0 upstream neighbours (in the case of 0 neighbours, no NW calculation is performed).

After neighbouring plans have been taken into account, the final steps of the algorithm (lines 6-9) involve limiting the amount of change the agent can make and bounding the variables. The amount of change is limited based on the value of the CTW system parameter. As can be seen in line 6, the green time is calculated using a weighted sum of the current green time (CGT) and the green time calculated so far by the agent. Once this value is known, the value is bounded such that a minimum green time (MGT) is allowed for each direction every cycle. With the final amount of time dedicated to the West direction calculated (line 8), the amount of green time for the North traffic flow is calculated by subtracting the West time from the total cycle length (line 9). The controlling agent will then implement a plan using these two green lengths until the next update time.

```
   /* Calculate average volumes over the time window and proportion
      of total volume for the west direction */
```
1 $AV_W = Average(West\_Observations)$
2 $AV_N = Average(North\_Observations)$
3 $P_W = \frac{AV_W}{AV_W + AV_N}$
```
   /* Calculate initial green time for west light based on
      proportion and cycle length */
```
4 $TG_1 = P_W \times CL$
```
   /* Update the green time based on neighbour influence */
```
5 $TG_2 = (1 - NW)TG_1 + NW(\frac{\sum_{i \epsilon X} TG_i}{|X|})$
```
   /* Weight the green time using CTW and ensure the times are
      bounded */
```
6 $TG_3 = (1 - CTW)TG_2 + (CTW \times CGT)$
7 $TG_4 = max(TG_3, MGT)$
8 $TG_W = min(TG_4, (CL - MGT))$
9 $TG_N = CL - TG_W$

Algorithm 2: Signal plan update for an intersection in the NetLogo model

4.5 Experimental Setup

To investigate the ability of the proposed algorithm to control traffic flow, the performance of the adaptive approach was compared to that of several other control schemes, outlined below:

- Fixed 30/30: Fixed phase lengths of 30 steps each.
- Fixed 40/20: Fixed phase lengths of 40 steps for the West light and 20 steps for the North.
- Fixed 20/40: Fixed phase lengths of 20 steps for the West light and 40 steps for the North.
- Task Allocation (TA): This control scheme uses the algorithm developed by de Oliveira and Bazzan (2007).

A number of different traffic distributions were created to the performance of each control scheme over a wide range of circumstances. The distributions (described in detail below) consisted of fixed volumes (Fixed Even, Fixed 2:1, Fixed 4:1), smoothly varying volumes (Sin/Cos, Fluctuate 1, Fluctuate 2), randomly varying volumes (From File1-From File10) and the distribution used by de Oliveira and Bazzan (2007) which keeps traffic volumes fixed for long periods of time before drastically changing them.

- Fixed Even - Both the N/S and W/E rates are held at 7.5 cars per minute for the entire simulation.
- Fixed 2:1 - The W/E rate is held at 10 cars per minute, while the N/S rate is held at 5 cars per minute.
- Fixed 4:1 - The W/E rate is held at 12 cars per minute, while the N/S rate is held at 3 cars per minute.
- Task Allocation (TA) - This distribution is taken from the work of de Oliveira and Bazzan (2007). The distribution begins with 12 cars per minute in the

W/E direction and 3 cars per minute in the N/S direction. The values do not change, but the directions they are assigned to switch at the following specific time intervals (all listed in simulation minutes): 60, 180, 300, 480, 660. For example, after 60 simulation minutes the volumes will alternate and there will be 12 cars inserted per minute in the N/S lanes and only 3 cars per minute inserted into the W/E direction. This switch happens at every one of the time steps listed.

- Sin/Cos - This distribution uses the equations below to calculate new rates at every minute. R_{WE} represents the cars per minute in the W/E direction, while R_{NS} is the cars per minute inserted in N/S lanes. CM is a variable representing the current minute of the simulation (which will range from 0 to 719). Using these equations, a total of 10 vehicles per minute will be inserted in total, while the level in each direction will fluctuate between 0 and 10 vehicles per minute.

  $R_{WE} = (((cos\,(90 + CM)) + 1) * 5)$

  $R_{NS} = ((sin\,(CM) + 1) * 5)$

- Fluctuate 1 (F1) - In this distribution, the N/S rate is held constant at 7 cars per minute for the entire simulation, while the W/E rate varies. Beginning at 0 simulation minutes, the W/E rate is at 2.5 cars per minute. The W/E rate begins to increase at 90 simulation minutes, finally peaking at 11 cars per minute at 225 simulation minutes. This value is held until 300 simulation minutes, when it begins to decrease, reaching a low of 6 at 375 simulation minutes. It immediately begins rising again, reaching 11 cars per minute at 555 simulation minutes. This value is held until the end of the simulation (720 minutes).

- Fluctuate 2 (F2) - The W/E rate in this distribution varies as in Fluctuate 1. The N/S rate also varies in this distribution, being set to $11 - R_{WE}$. Like the Sin/Cos distribution, the total rate of insertion remains constant at 11 vehicles per minute, however the rates of both directions fluctuate.

- From File (FF1-FF10) - These distributions consists of random fluctuations of

the N/S and W/E rates every minute of the simulation. Since each distribution consists of 720 entries for each direction, they cannot be described in full here. Instead, the method used to create the distributions will be described, allowing similar distributions to be created easily. First, each direction is assigned a random rate between 0 and 11 to begin the simulation. At each simulation minute, each rate has a random normally distributed ($\mu = 0, \sigma = 1$) number added to it. The rates remain bounded between 0 and 11 however, to ensure the network does not become supersaturated. Using this method, the rates in both directions randomly fluctuate up and down throughout the entire simulation. Ten distributions generated using this method were saved and used as FF1-FF10 in testing.

Due to the random nature of the environment, a number of simulation runs are required to determine any significant conclusions. For this reason, each algorithm was simulated 25 times on each of the distributions. For each simulation, the average trip length for every vehicle was calculated and used to compare the performance of each algorithm. Since a shorter average trip length implies that vehicles (at least on average) travel faster through the network, this is a generally accepted performance criterion for comparing traffic control measures. When used for control within a simulation, some algorithms performed extremely poorly on several distributions. This poor performance would result in a large number of vehicles being present in the network as roads backed up and complete grid-lock stalled all traffic, resulting in extremely long simulation times as each vehicle must be updated every second. For this reason, an algorithm was determined to have 'failed' in controlling traffic in any case where the number of stopped vehicles within the network exceeded 2000. The results generated from these experiments are presented, in detail, within Section 4.6.

4.6 Experimental Results

The results found through simulation are presented in both Table 4.1 and Figure 4.3. Table 4.1 presents the average trip time for the entire simulation, along with the standard deviation in trip time. The results of the algorithm presented here are bolded under the 'Prop Avg' column, while the best other algorithm is also

Table 4.1: NetLogo simulation results for each algorithm/distribution combination

Distribution/Algorithm	30/30 Avg	30/30 SD	40/20 Avg	40/20 SD	20/40 Avg	20/40 SD	TA Avg[1]	TA SD	Prop Avg	Prop SD
Fixed Even	**224.80**	0 69	Failed	Failed	Failed	Failed	Failed	Failed	**231.67**	1 56
Fixed 2 1	281 06	34 79	**206.15**	1 64	Failed	Failed	Failed	Failed	**221.36**	1 21
Fixed 4 1	Failed	Failed	**192.10**	0 93	Failed	Failed	Failed	Failed	**199.23**	1 81
TA	**261.15**	12 63	Failed	Failed	Failed	Failed	Failed	Failed	**224.13**	1 88
Sin/Cos	**238.80**	11 70	Failed	Failed	Failed	Failed	Failed	Failed	**179.07**	1 05
F1	**302.95**	42 52	Failed	Failed	Failed	Failed	Failed	Failed	**244.76**	11 35
F2	**341.09**	57 99	514 40	29 67	Failed	Failed	Failed	Failed	**171.24**	0 95
FF1	367 64	59 83	**217.03**	60 48	Failed	Failed	Failed	Failed	**215.88**	1 88
FF2	**213.60**	4 34	Failed	Failed	312 59	23 31	Failed	Failed	**213.77**	1 02
FF3	**230.94**	22 42	685 37	59 73	Failed	Failed	Failed	Failed	**217.63**	1 86
FF4	**292.61**	86 30	356 38	23 84	Failed	Failed	Failed	Failed	**213.34**	1 26
FF5	312 12	57 31	**205.67**	1 29	Failed	Failed	Failed	Failed	**213.47**	1 07
FF6	**200.17**	3 43	264 15	20 16	772 78	81 58	380 27	62 51	**196.92**	0 96
FF7	**282.59**	55 59	410 92	41 53	Failed	Failed	Failed	Failed	**208.47**	1 52
FF8	195 31	1 91	417 11	29 03	221 67	11 68	**171.41**	26 18	**183.52**	1 26
FF9	**238.38**	19 69	Failed	Failed	Failed	Failed	Failed	Failed	**229.05**	4 79
FF10	381 49	122 85	**202.38**	1 57	Failed	Failed	Failed	Failed	**197.14**	0 86
Bests	2		3		0		1		11	
Failures	1		6		14		15		0	

bolded for easy comparison/reference. It can be noted from Table 4.1 that the best performance for the fixed distributions was found using fixed plans. Also, the number of vehicles within the first two distributions (Fixed Even and Fixed 2:1) matched exactly the proportion of green time assigned by the two fixed plans (30/30 and 40/20 respectively). In these cases, the proportional algorithm performs worse, but the average trip time increases by no more than 8%. The adaptive algorithm, however, can make up for these losses by performing much better than the fixed plans on distributions where the volumes vary throughout the simulation. The bottom rows of Table 4.1 also paint a clear picture on overall performance. As can be seen from the total failures for each algorithm, the task allocation approach failed on all but two of the distributions (even failing to perform well on the distribution it was originally evaluated on). The only algorithm that did not fail in any case was the adaptive algorithm proposed here. Also, the adaptive algorithm achieved the best travel time on 11 out of 17 total distributions (or approximately 65% of cases).

Figure 4.3 presents comparisons between the adaptive proportional algorithm and the best performing of the other control schemes. Within the figure, the average travel time for that distribution is plotted, along with error bars representing 1 standard deviation. From Figure 4.3, it is easily seen that the adaptive algorithm outperforms all others in the majority of cases and performs nearly as well in all other cases. The

[1]Using algorithm specified by de Oliveira and Bazzan (2007)

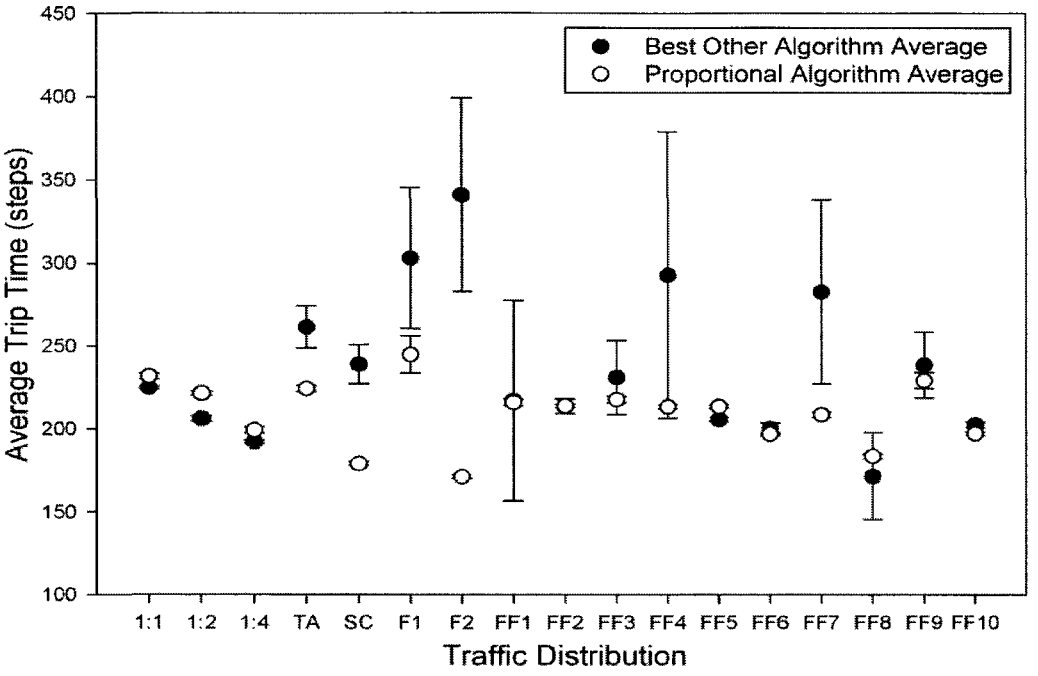


Figure 4.3: NetLogo proportional algorithm vs. best other algorithm, shown with 1 standard deviation error bars

standard deviations shown also demonstrate an advantage of the adaptive algorithm. In many cases, the fixed control schemes result in extremely high standard deviations, with the largest being over 29% of the mean trip time. In comparison, the adaptive algorithm maintains a standard deviation of less than 1% of the mean travel time in all but 2 of the cases (with the largest still being under 5%). The reason the adaptive algorithm has much less deviation than the fixed schemes is that the controlling agents will change signal plans to address congestion on incoming roads. This is especially important in a probabilistic environment, as a large number of vehicles can be inserted quickly, which a fixed plan may fail to handle appropriately. If the situation is serious enough, it can have long lasting effects on overall performance, as vehicles will continually fail to move efficiently through the network while high levels of congestion remain. The adaptive plan is successful as it infers a traffic model in real-time and uses that model to allocate resources in an effective manner.

4.7 Summary

In this section,an adaptive multi-agent traffic control approach was evaluated through a simple simulation of traffic. A model for implementing this multi-agent control scheme, requiring only vehicle sensors and local communication (two readily available tools), was developed within the NetLogo modelling environment (Section 4.2).

The overall effectiveness of the proposed system was compared on a wide range of traffic distributions against several simple control schemes (explained in Section 4.5). The results presented in Section 4.6 showed that overall, the adaptive algorithm performed much better than fixed traffic plans. It was shown through the simulation results that the multi-agent adaptive control algorithm was very robust, handling a wide range of traffic distributions successfully while other approaches failed. It was also shown that the algorithm maintained much less deviation in travel time than the fixed approaches, as the adaptive nature allowed it to cope with unexpected fluctuations in traffic volume.

With these benefits being mentioned, it is worth noting that there is still a large number of changes that must be considered to make this control scheme applicable to real-world traffic control. First, the algorithm must be extended to allow a variable amount of phases, as the number of phases at each intersection within a real traffic network can vary. Also, while the effectiveness of the algorithm has been shown on many simple distributions using an extremely simple traffic model, it must be confirmed that the algorithm is also successful when dealing with traffic volumes found in a real situation. Finally, the algorithm must address the more complex street networks found beyond this simulation, including a varying number of lanes, different numbers of incoming edges, as well as lanes with different purposes (e.g., turning lanes). In Chapter 5, a real-world traffic model is developed and a set of realistic vehicle routes is created. Following the model creation, Chapter 6 presents an improved control scheme which is capable of successfully controlling traffic within a more realistic traffic simulation.

Chapter 5

Modelling a Realistic Traffic Scenario in SUMO

5.1 Introduction

While Chapter 4 developed a simple algorithm and showed its ability to effectively control traffic, it also used a very simple network and traffic model for testing. This chapter deals with the creation of a realistic traffic model to test the intelligent control algorithm that will be proposed in Chapter 6. To create a realistic model, a section of the downtown area of the City of Ottawa was modelled within the SUMO simulation environment. Details on why the SUMO traffic simulator was chosen are provided in Section 5.2, while a brief description of the SUMO software package is supplied in Section 5.3. Motivation for the selection of the modelled area, as well as details on the creation of the network within SUMO are provided within Sections 5.4.1 and 5.4.2. Once the network was completed, vehicle routes to be used in the simulation were generated using data provided by the City of Ottawa for the modelled traffic network. The data supplied consisted of single day observations of vehicle numbers for each intersection within the modelled area. Details on the data supplied, as well as the process used to transform this data into vehicle routes, is given in Sections 5.4.3 and 5.4.4.

5.2 Reasons for Choosing Sumo

As can be seen from the summary of microscopic traffic simulators in Table 2.1, there are a number of available simulators to choose from. After analyzing the goals of this project, considering the direction of possible future work and investigating the offerings of the various simulators, it was decided that the SUMO traffic simulator should be used. There are a number of important features offered by the SUMO simulation environment which factored into this decision detailed below.

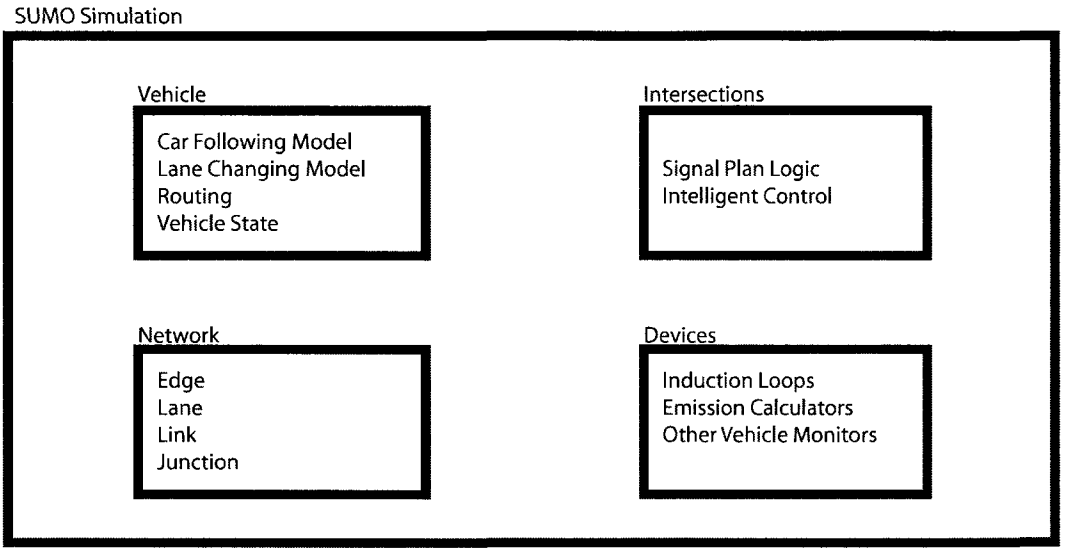


Figure 5.1: Several SUMO modules which may be modified to produce new behaviour/features

- Open Source: There are several advantages of using an open-sourced package such as SUMO. For one, if required, it can be modified in any way the user desires/requires (e.g., updated driver models or new traffic devices). Figure 5.1 shows several of the modules that can be modified within the SUMO simulation engine. Second, being able to view the source code can help in finding errors within one's own work. Also, the software is provided free of charge, which certainly fits into any budget that may exist. Further to these advantages, an open source project allows other researchers to easily check/validate the work undertaken, which generally results in more robust software.

- Development Community: As with a large number of open-sourced software packages, SUMO has an active development community. Members of the mailing list include people ranging from first time users, to the people responsible for developing SUMO from the beginning. This active community makes it easy to find support when working with the software, which can be invaluable whether you are just beginning with SUMO or are an experienced user. Also, it was found that developers working full-time on SUMO were involved in the mailing list, allowing problems that did arise to be addressed in a timely manner.

- Portability: While SUMO is only openly supported for the Windows and Linux operating systems, it can also be easily built for use under Mac OS X. While this may not be an important contributing factor for some users, it certainly has helped throughout the work conducted here. With many simulations to run, an extremely large amount of computing time has been required. Fortunately, there has been a number of available computers to execute simulations on; a number that would be much smaller had SUMO only supported one of the above mentioned operating systems.

- GUI: A graphical user interface (GUI) can be indispensable when initially developing a control system. It is much easier to identify problems as they occur using a GUI as opposed to text-based output. It is also easier to immediately evaluate the effect a control measure is having at an isolated location (e.g., a single intersection) when vehicle movements can be reviewed. The use of a GUI becomes less important after the system has been developed, but was truly helpful in developing the system presented here.

- Control of Simulation: A user can control every entity within a simulation easily through the supplied Traffic Control Interface (TraCI). This includes traffic signals (which was one of the main requirements of this work), as well as vehicle behaviour.

- Speed: The SUMO simulator was designed to execute quickly. In fact, SUMO can execute up to 100 000 vehicle updates per second on a 1GHz computer (SUMO Features, 2011). When running thousands of simulations, efficient execution is very important, as even a small increase in speed over a competitor results in significant time savings.

- Large Network Support: SUMO is capable of handling extremely large networks. It has been shown to be capable of modelling networks consisting of tens of thousands of edges (SUMO Features, 2011). While this factor is not very important for the work presented here (with a relatively small network), it is beneficial as it allows future work to expand to large networks if desired.

- Used by Others: SUMO has been used in several other traffic research works, including García-Nieto et al. (2011) and Passos and Rossetti (2010).

5.3 Included Applications

The SUMO software package supplies users with a number of useful programs for creating and running traffic simulations. The available programs are explained briefly below, while more detail on the use of a subset of these programs is given in subsequent sections.

Simulators

- SUMO: The base simulation of the entire SUMO software package which runs from a command line.

- GUISIM: A graphical version of SUMO, which takes the same input and creates the same output, along with a graphical representation of the simulation. GUISIM provides a number of options for automatic colouring of the simulation, such as changing edge colours to represent the speed, congestion, or emissions from each edge.

Network Generators

- NETCONVERT: Converts a wide range of possible inputs into SUMO readable road networks. Networks can be created from XML descriptions or automatically generated from OpenStreetMap (OpenStreetMap, 2011). NETCONVERT also supports (to varying degrees) import of networks from a few other traffic simulators (e.g., VISSIM (2011)).

- NETGEN: Allows for the automatic creation of abstract road networks. This can be used to easily generate a number of networks easily and quickly.

Route Generators

- DUAROUTER: Generates shortest path routes for vehicles. The input to DUAROUTER consists of a network, as well as demand definitions for the simulation. The demand definitions can be in the the form of specific trips (departure time, origin, destination) or flow specification (origin, destination, beginning interval time, ending interval time, number of vehicles). Origins and destinations can also be supplied as specific edges, or as districts (sections of the network).

- JTRROUTER: Generates routes based on traffic flows and turning ratios. JTRROUTER requires that turning ratios be specified over a number of intervals for each edge vehicles can leave. Vehicle flows (number of vehicles inserted on an edge over a specific interval), as well as 'sink' edges (edges where vehicles leave the network automatically upon arrival), must also be defined. From these inputs, JTRROUTER generates vehicle routes by probabilistically selecting the next edge for each vehicle at each junction until that vehicle reaches a sink edge. To prevent the (theoretical) chance of infinite trips, a maximum number of edges within a trip can be specified.

- DFROUTER: Generates routes based on observed induction loop values. This program is designed to take observations made by induction loops within real-world networks, where each entry/exit point of the network has an induction loop present. From these observations, routes that closely resemble those of the real-world network can be inferred for use within SUMO.

- OD2TRIPS: This program generates routes from data supplied in an origin-destination matrix.

5.4 Model Creation

This section details the model creation process, as raw data is transformed into several traffic simulator scenarios. There are three main steps required when creating a traffic simulation: definition of the road network, specification of traffic volumes and route

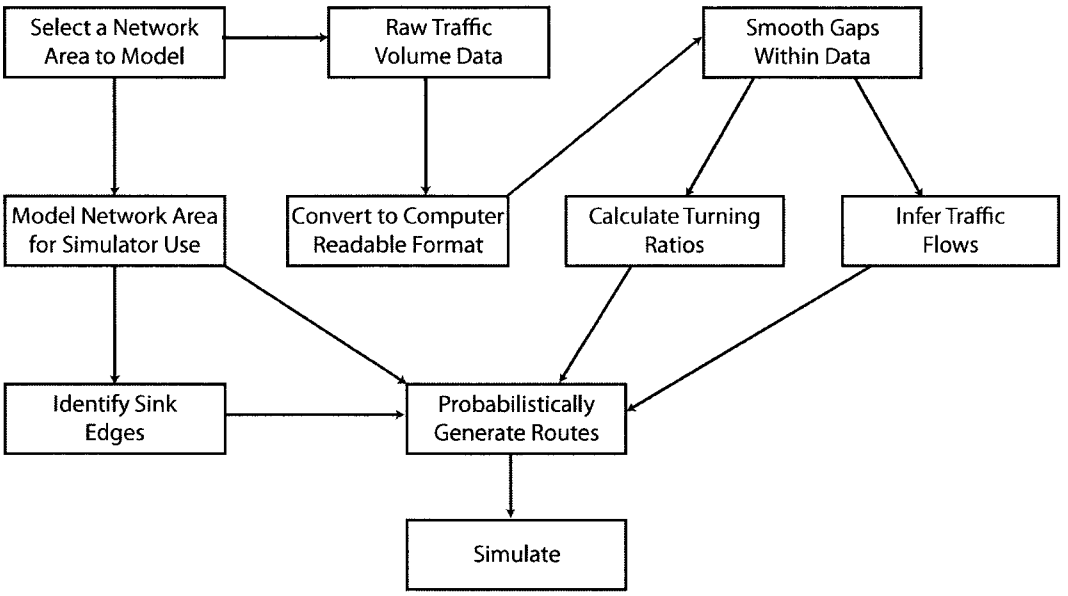


Figure 5.2: Flowchart showing the movement from initial network identification to simulation

generation. These three steps are explained in detail in Sections 5.4.1-5.4.4, while a flowchart detailing the movement from raw data to traffic simulation is provided in Figure 5.2.

5.4.1 Network Selection

The first step in creating the real-world traffic model was identifying and modelling a section of the road network. After considering many factors, it was decided to use a 9x7 grid-like area from the downtown area of the City of Ottawa. An outline of the area chosen can be seen in Figure 5.3. There were a number of factors motivating the choice of this network section, outlined below.

- Size: The modelled area consists of over 50 signalized intersections. While this may not be considered a large network, it is considerably larger than many of the networks used in previous traffic research (as can be seen from Chapter 3, where small networks of 1-4 intersections are common). Also, modelling the area involved encoding a large amount of information (traffic volumes, turning ratios) and since this is the first work to model traffic using this data, no tools

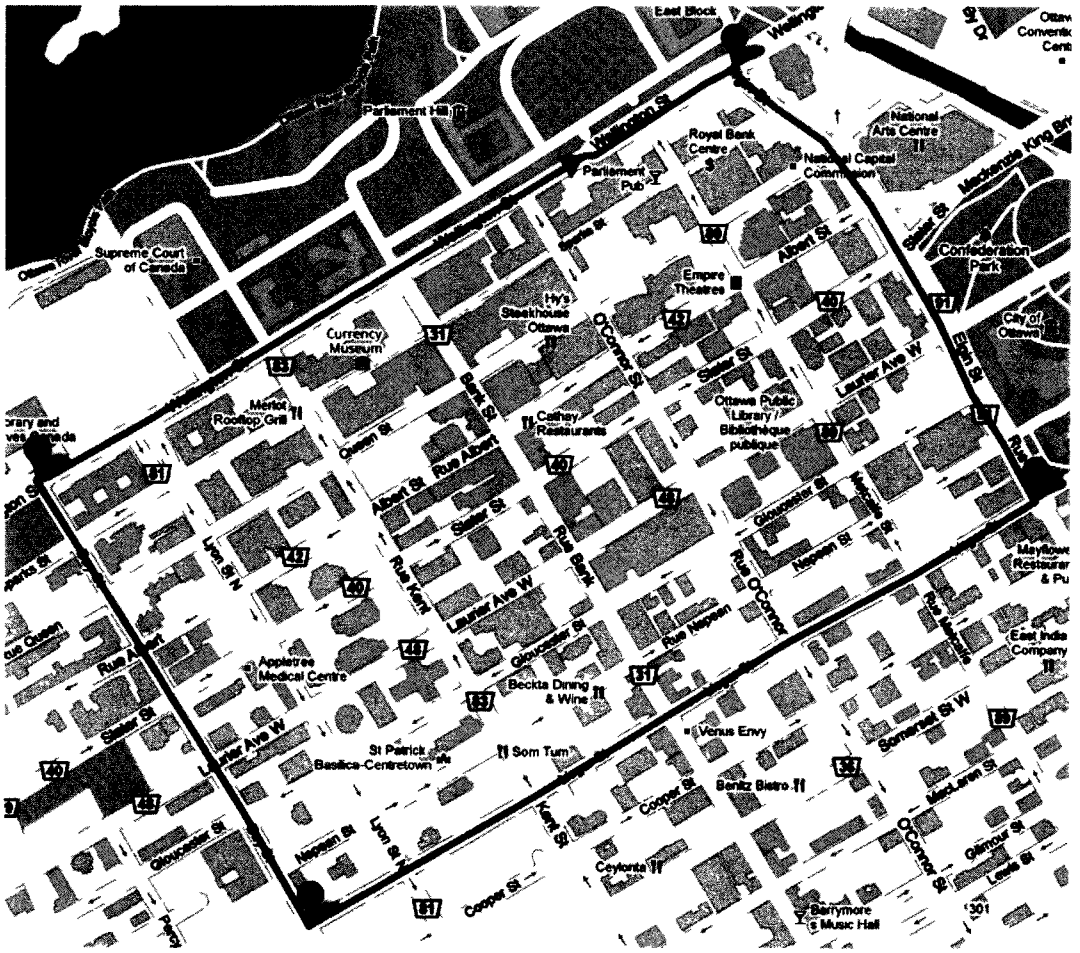


Figure 5.3: An outline from Google Maps of the traffic area modelled

existed to simplify the process. For this reason, choosing an extremely large network (for example, the entire city) would require a prohibitive amount of data entry.

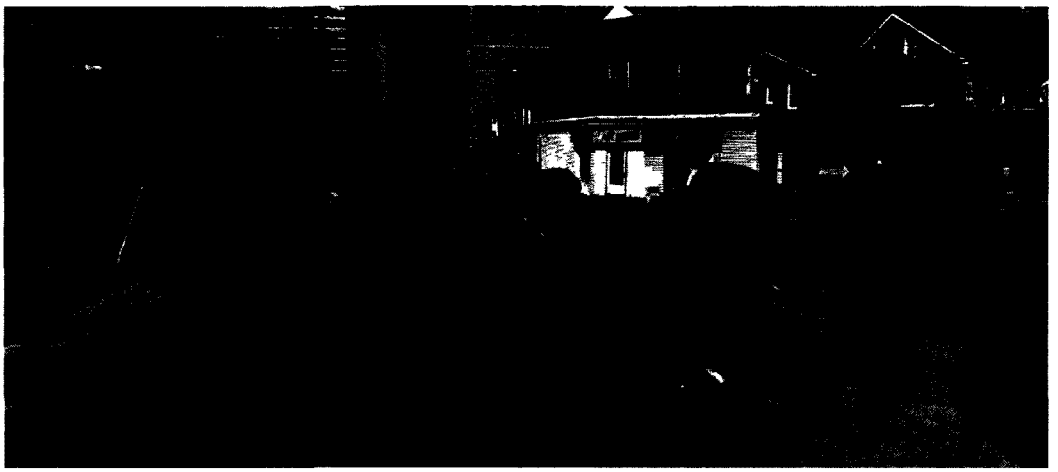
- Road Diversity: There is a wide range of road structure contained within this network. Bay St. (Figure 5.4(a)) and Lyon St., for example, are low volume, one-way residential streets with a small number of lanes and simple structure. This can be compared to Elgin St. (Figure 5.4(b)), which has a high number of lanes, high volume, turning lanes and advanced green signals. It was felt that this diversity would aid in demonstrating the effectiveness and generalizability of the proposed traffic controller.

- Available Information: To model the traffic flows found in the real-world, data representing this traffic must be available. Also, to evaluate the abilities of the adaptive distributed control structure developed, it would be advantageous to have the signal plans currently used to control the area available for comparison. Through the City of Ottawa, single-day traffic observations for each intersection were made available, which consisted of both the traffic volumes and turning ratios at each junction. Also, the City of Ottawa supplied a description of the signal plans currently used to control the intersections within the modelled area.

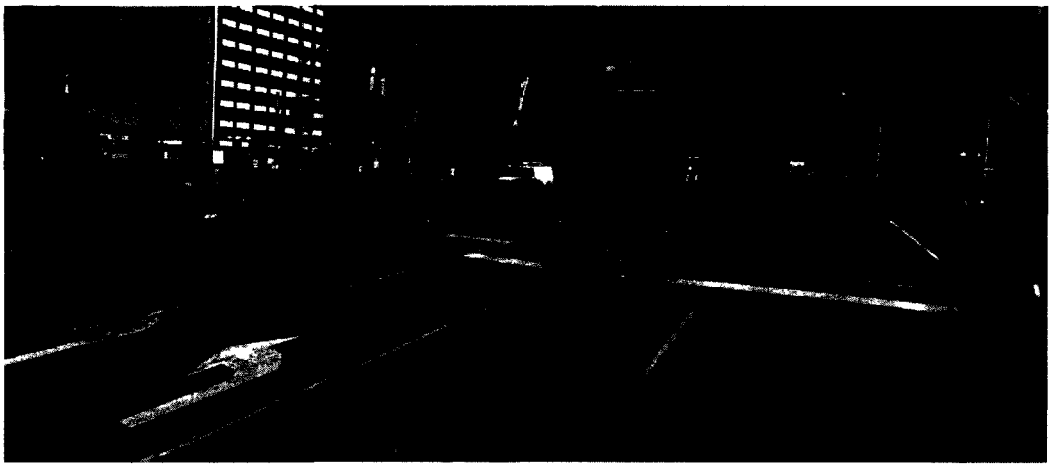
- Significance: As mentioned above, the area considered is part of the downtown core of Ottawa. This area experiences traffic volumes ranging from extremely low (non-peak hours on residential streets) to very high (peak hours on main streets). Again, it was thought that this variability would help to demonstrate the robustness of the system outlined here.

5.4.2 Network Modelling

After selecting the area to model, it must be translated into a SUMO traffic network. This process is made easier by the NETCONVERT tool supplied in the SUMO software package, which allows the import of networks directly from OpenStreetMap (OpenStreetMap, 2011). OpenStreetMap allows for a rectangular area of a traffic



(a) Bay St.



(b) Elgin St.

Figure 5.4: Example of intersections on Bay St. and Elgin St. from Google StreetView

network to be exported, which can then be imported into SUMO using the NETCONVERT tool. From Figure 5.5, which shows the network imported directly from OpenStreetMap, it can be seen that some unnecessary edges are included, extending far beyond the area to be modelled. For the purpose of this work. these outlying edges are trimmed such that only a small part of each edge protruded past the last street under consideration.

It should also be noted that the OpenStreetMap import incorrectly specified two

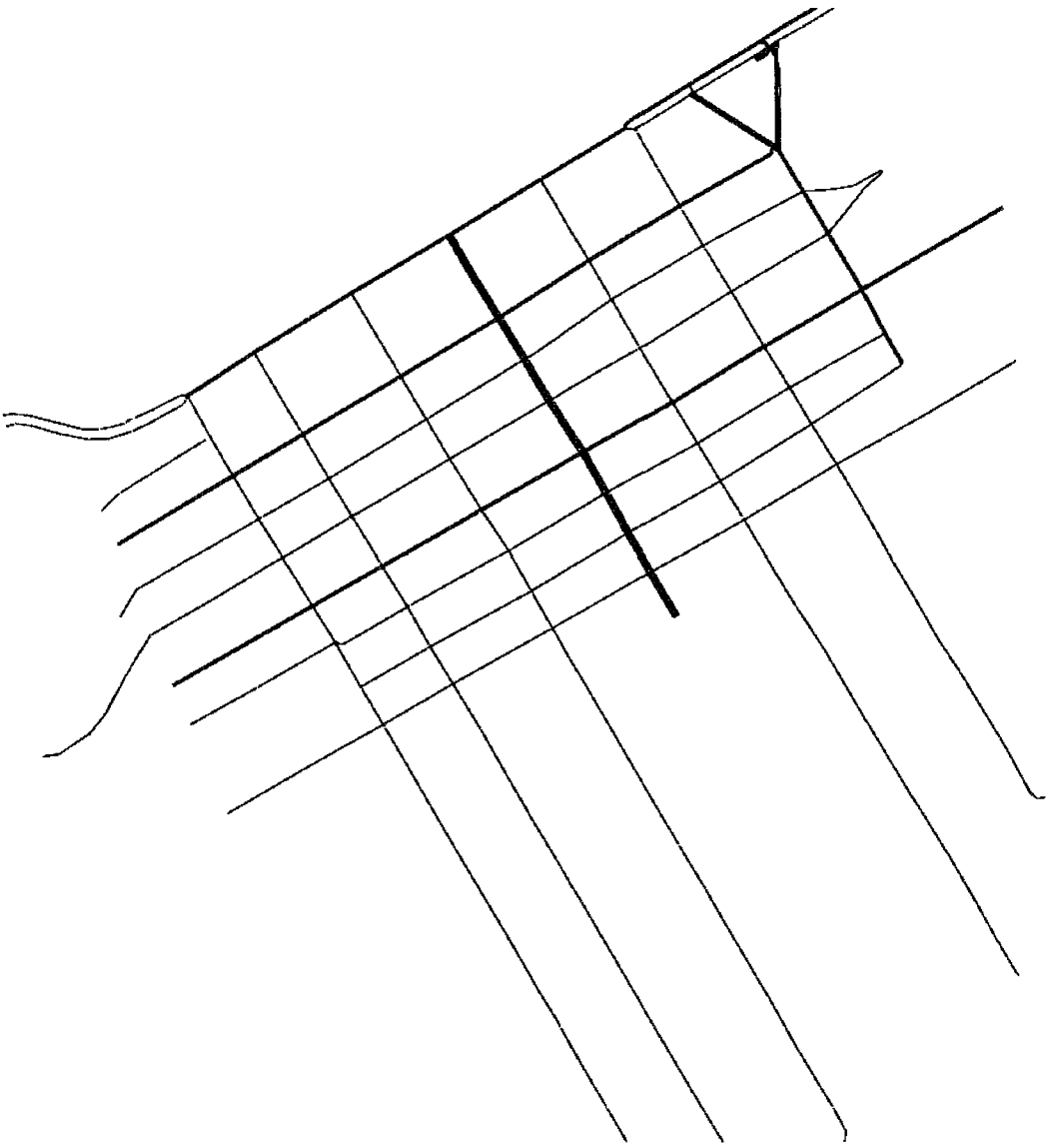


Figure 5.5: The initial SUMO network after OpenStreetMap import

extremely important traffic network features: number of lanes and the existence of turning lanes. To rectify this problem, Google Maps with Street View (Google Inc., 2011) is used to view actual network characteristics quickly and easily. Using the information gathered, modifications can be made to increase the accuracy of the street network within SUMO. One issue present is the fact that some street sections change the number of lanes between intersections. This is common when a turning lane is added to the street within a certain distance of an upcoming intersection. There is no easy method to model these situations within SUMO, so additional lanes must be added along the entire street section whenever a partial turning lane is required. While this increases the network capacity and decreases the accuracy of the model slightly, it was deemed a necessary adjustment. Also, since neighbouring intersections are discovered by looking at the originating node of an incoming edge, each set of neighbouring intersections must be connected via a single edge. To model road bends, however, SUMO divides what is a single edge in the real traffic network into a number of edges and nodes. For this reason, road curvature was also removed from the network and intersections were connected by a single edge. This modification, however, has little effect on the overall model as all intersections within the modelled area are connected with edges that are nearly straight. Figure 5.6 shows the completed traffic network which is used for experimental testing of controllers.

5.4.3 Traffic Volume Creation

After the traffic network is successfully modelled, the next step is to capture the traffic volumes within the network from the data provided. Unfortunately, the traffic volume data available is provided only in print form. A large amount of data entry is therefore required to transform this data to a computer readable form. During data entry, a single file detailing the supplied traffic volume information is created for each intersection within the network. An example of the data generated by this process can be seen in Table 5.1. There is no available traffic volume information for four of the intersections within the network. Volumes for these intersections then, are inferred from the exit/entry volumes of neighbouring intersections. A simple example of the inference process is shown in Figure 5.7, with the available information from

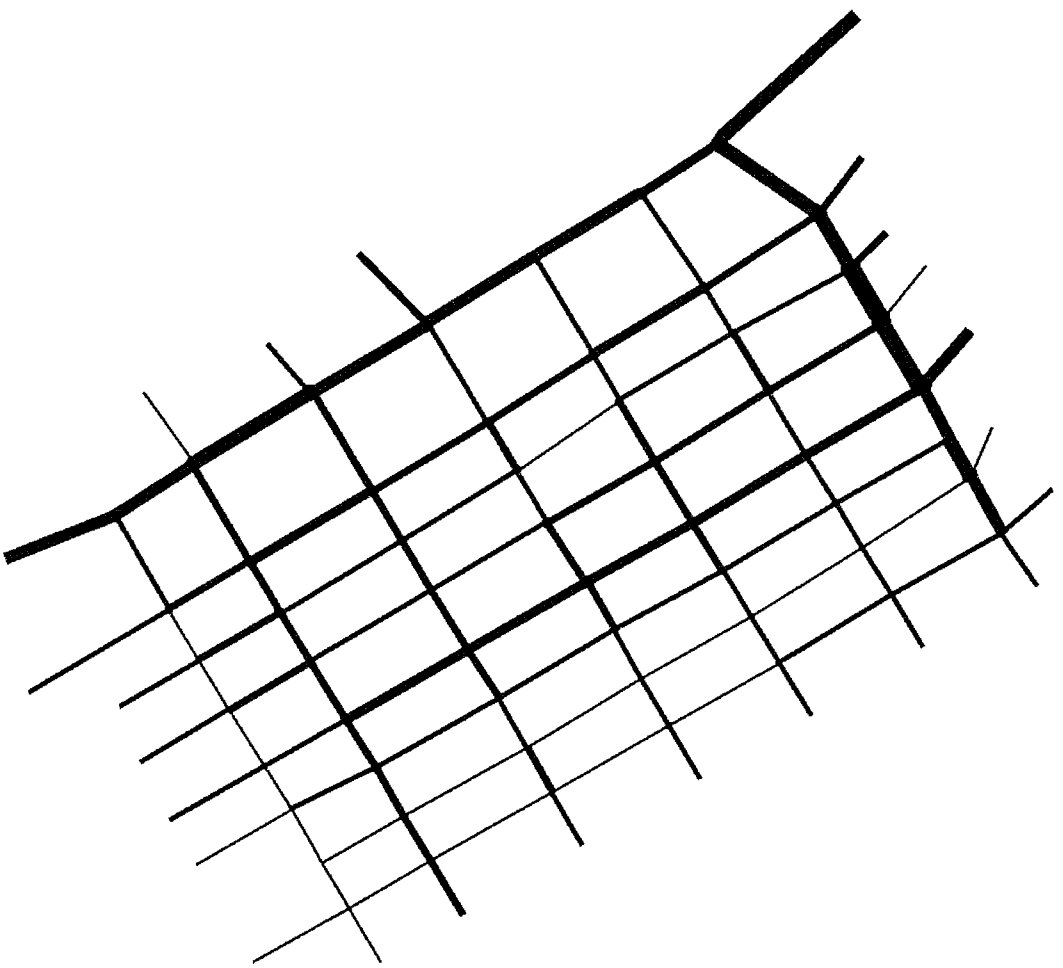


Figure 5.6: The final SUMO network used for experimental tests

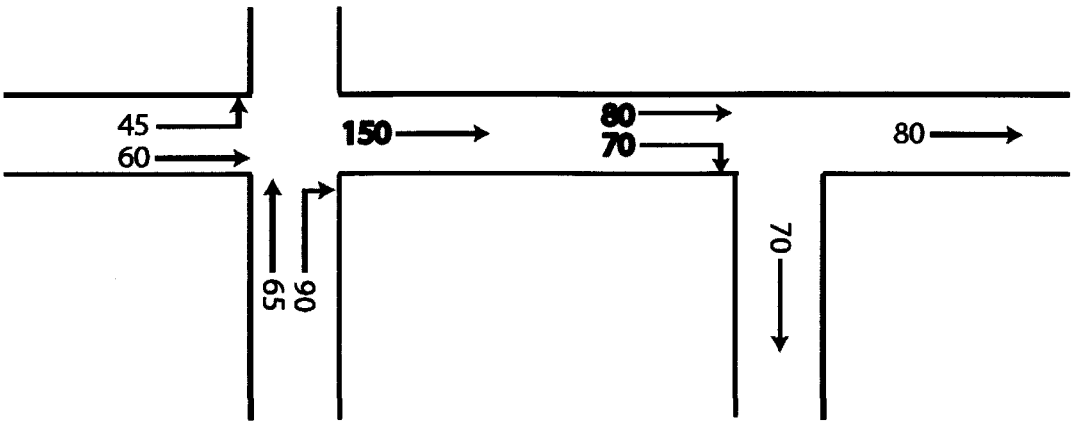


Figure 5.7: Volumes inferred (in red) for an intersection based on known neighbour volumes (in black)

neighbouring intersections shown in black and the inferred volumes for the unknown intersection in red. In this case, the incoming and outgoing volumes over the interval are equal; if they are not equal, the difference is addressed using a flow value (explained in Section 5.4.4). Also, it can be seen from Table 5.1 that there are gaps in the available data for each intersection between the periods of 10:00-11:30a.m. and 1:30-3:00p.m. To smooth the transition between the known information, new intervals of 30 minutes each are added to each intersection, with the volumes gradually increasing/decreasing in proportion to meet the demands of the next known interval. The algorithm used to automatically calculate the start time and volume of each new interval is given in Algorithm 3. This simple smoothing process is acceptable for this work, as the missing intervals are relatively short and infrequent. If the missing intervals were more frequent, a more advanced smoothing approach such as spline fitting (Reinsch, 1967) may be more appropriate.

Table 5.1: An example of the available traffic volume data for each intersection

Start	End	Northbound				Southbound				TOT	Eastbound				Westbound				TOT	FINAL
Start	End	LT	ST	RT	SUB	LT	ST	RT	SUB	TOT	LT	ST	RT	SUB	LT	ST	RT	SUB	TOT	FINAL
7:00	8:00	43	1	119	163	0	0	0	0	163	0	413	101	514	28	305	0	333	847	1010
8:00	9:00	57	1	213	271	0	0	0	0	271	0	537	95	632	28	342	0	370	1002	1273
9:00	10:00	69	1	174	244	0	2	0	2	246	0	388	66	454	34	348	0	382	836	1082
11:30	12:30	76	0	149	225	17	0	0	17	242	0	291	80	371	24	385	0	409	780	1022
12:30	13:30	67	1	131	199	0	4	0	4	203	0	275	95	370	22	385	0	407	777	980
15:00	16:00	51	0	131	182	0	0	0	0	182	0	383	93	476	22	392	0	414	890	1072
16:00	17:00	71	0	128	199	0	0	1	1	200	0	413	96	509	22	505	0	527	1036	1236
17:00	18:00	81	0	156	237	0	4	7	11	248	0	384	79	463	27	469	0	496	959	1207
8 Hour Total		515	4	1201	1720	17	10	8	35	1755	0	3084	705	3789	207	3131	0	3338	7127	8882

Input: $Last\_Volume$, $Last\_Interval\_End$, $Next\_Volume$, $Next\_Interval\_Start$

```
/* Calculate number of intervals between available measurements
   */
```

1 $Fill\_Intervals = \frac{Next\_Interval\_Start - Last\_Interval\_End}{0:30}$;

```
/* Calculate the change in volume required during each interval
   */
```

2 $Delta\_Volume = \frac{Next\_Volume - Last\_Volume}{Fill\_Intervals+1}$;

```
/* Set start time and volume for each new interval              */
```

3 **for** *i from 1 to* $Fill\_Intervals$ **do**

4 $\quad Start\_Interval_i = Last\_Interval\_End + (i-1) \times 0:30$;

5 $\quad Volume\_Interval_i = round(Last\_Volume + i \times Delta\_Volumes)$;

6 **end**

Algorithm 3: Used to fill in missing intervals within the data

5.4.4 Scenario Creation

The SUMO software package includes a tool (JTRROUTER) for automatically generating routes probabilistically from supplied traffic flow and turning ratios. The method of calculating these values from the available data (explained in Section 5.4.3) is detailed below.

Turning Ratios

The turning ratios for each incoming edge are rather easy to calculate from the available data. The total vehicles exiting each incoming edge is supplied, as well as the partial sums for all turning possibilities. It is easy, then, to calculate the turning ratio for each direction i using the equation below. This must then be done for each interval and each incoming edge to determine the turning ratios to be used throughout the simulation.

$$Ratio_i = \frac{Count_i}{Total}$$

Traffic Flows

Traffic flows (the number of vehicles to be inserted/removed on an edge over each interval) are slightly more difficult to calculate, but are required to model the net gain/

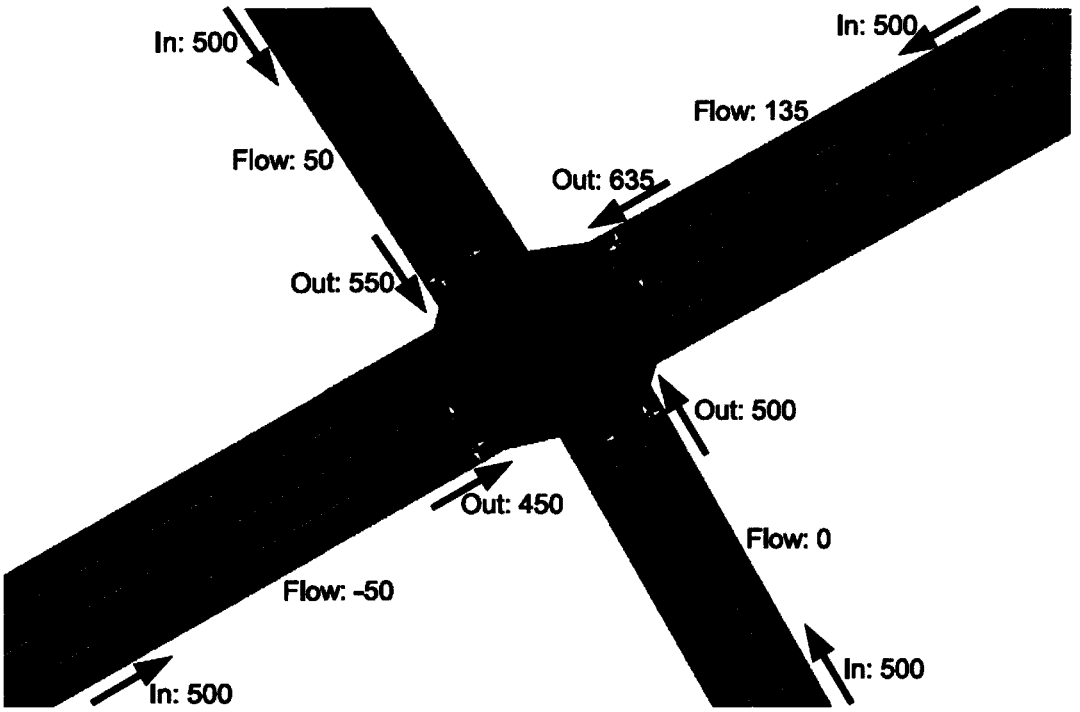


Figure 5.8: An example of flow computation for the four incoming edges of an intersection

loss of vehicles between intersections (e.g., from entering or leaving locations along the edge). While the total number of incoming vehicles is known for each edge, the flow depends on the number of vehicles entering the edge from neighbouring intersections. Using information available from the network description, the origin intersection of each incoming edge can be determined. The number of vehicles entering the edge from the neighbouring intersection can then be compared to the number exiting the terminal end of the incoming edge to determine the overall flow. Several examples of flow calculations are shown in Figure 5.8. A SUMO-specific issue is the lack of a negative flow model, which is capable of capturing a net loss of vehicles on an edge over time (SUMO's route generator supports only the addition of vehicles over time on an edge). For this reason, any edge that has a net loss of vehicles over an interval must be handled within the simulation itself by probabilistic removal of vehicles over the specific time interval. Algorithm 4 shows the process executed during each timestep to ensure the correct number of vehicles are removed from the edge over the interval.

```
1 foreach Edge in simulation with a negative flow interval do
2 |   Vehicles_To_Remove =
  |   Number of vehicles still to be removed from edge in this interval;
3 |   Remaining_Interval_Time =
  |   Number of timesteps left in the edge's current interval;
4 |   Random_Number = Random number in [0,1] ;
5 |   if Random_Number <= Vehicles_To_Remove / Remaining_Interval_Time then
6 |   |   Remove a random vehicle from edge;
7 |   |   Vehicles_To_Remove = Vehicles_To_Remove − 1;
8 |   end
9 end
```

Algorithm 4: Used to probabilistically remove vehicles from a edged that require a negative flow over an interval

Sink Edges

With the turning ratios and traffic flows defined, the last piece of information required to generate the routes is a list of 'sink edges' These sink edges represent exit points of the network, with a vehicle's route being terminated when it reaches any sink edge. For this work, the sink edges are defined as every boundary edge where the roadway continues outside the intersections being considered.

Route Creation

Finally, the traffic flows, turning ratios, sink edges and network structure are used as input for the JTRROUTER program. This program uses the defined traffic flows for each edge to determine how many cars must be inserted during each interval. Start times are then generated for each vehicle that must be inserted into the simulation throughout the simulated time. The route each inserted vehicle follows can then be generated probabilistically based on the turning ratios for each edge it travels. The route generator continues to choose new edges for the vehicle until it reaches a sink edge, at which point the route is terminated. Once these routes are generated, they can be used as input for a SUMO simulation, where the vehicles will be inserted at their specified start time, follow their route and be removed from the simulation upon its completion.

The available data was used to run the JTRROUTER program 15 times, which

generated 15 different sets of vehicles routes. These 15 traffic scenarios were then used to test the algorithm described in Chapter 6 and generate the results presented in Chapter 7.

5.5 Summary

This chapter covered the creation of a realistic traffic model, based on real-world data supplied by the City of Ottawa, within the SUMO traffic simulation environment. Reasons for choosing the SUMO environment (including portability, available support, and open source licensing) were outlined (Section 5.2) and a brief description of the available programs included within the SUMO software package (Section 5.3) were provided.

The chapter included an insight into the reasoning used when selecting a traffic area to model (Section 5.4.1), as well as details on the modelling process. These details (shown in Section 5.4.2) included the initial automated import from OpenStreetMap (OpenStreetMap, 2011), the removal of the unnecessary edges from the automatically generated network and the modification of the network to better match that of the real network. Also, Section 5.4.2 outlined several small changes that were made to the network which made the modelling process much easier without sacrificing the realism of the network model.

After describing the network creation process, Section 5.4.3 explained the process of generating vehicle volumes and turning ratios for the network based on the data supplied by the City of Ottawa. These volumes were then used in Section 5.4.4 to create vehicle routes which could be used within the SUMO simulation environment.

With a realistic traffic model now available, the following chapter will detail the control algorithm used by agents within the system to generate signal plans. The vehicle routes which were generated (detailed in Section 5.4.4) are then used to test the algorithm and compare the algorithm's performance, with the results being presented in Chapter 7.

Chapter 6

Distributed Adaptive Traffic Control Algorithm

6.1 Introduction

Chapter 4 detailed a simple algorithm for traffic control that was evaluated within the NetLogo (Wilensky, 1999) modelling environment. This chapter introduces an improved controlling agent model, as well as a new algorithm that is capable of operation within a real traffic network. The algorithm presented has three important characteristics: it adapts signal plans based on current (and predicted) traffic measurements, it is a distributed system and it relies solely on localized communication/ computation.

The remainder of the chapter is organized as follows. First, the small number of assumptions that are required when considering this algorithm are outlines in Section 6.2. The improved agent model is then presented in Section 6.3, including an explanation of all available information (both constant and observed/calculated) the agent is aware of. A detailed explanation of all of the system parameters used within the control algorithms is provided in Section 6.4. Section 6.5 explains the key characteristics (mentioned above) of the control system in detail, including explanation of the advantages each characteristic carries. The algorithm used by agents when observing the network is given in Section 6.6, while the signal plan update algorithm used by agents is detailed in Section 6.7. Further to the signal plan update algorithm, Sections 6.7.3-6.7.7 provide algorithms for several proposed methods of traffic volume calculation. Finally, a summary of the chapter's contents is provided in Section 6.8.

6.2 Assumptions and Limitations

A small number of requirements are assumed to be met within this system. First, it is assumed that each controlling agent has some means of communicating with

neighbouring intersections (by what means this is achieved is considered beyond the scope of this work). Second, it is assumed that traffic sensors which are capable of measuring the number of passing vehicles are located at the beginning and end of each lane within the network, allowing for an accurate calculation of vehicle numbers. While this is generally not the case in the real-world, sensors for this purpose do exist and could be used in a real-world application. Further work in the area of vehicle-to-vehicle and vehicle-to-infrastructure communication could, however, present a more accurate and cost-effective method of generating these traffic measurements. These, too, are considered beyond the scope of the research reported here.

6.3 An Improved Intersection Control Agent

Due to the added complexity of a real-world traffic scenario, an intersection control agent requires an additional awareness of a number of properties. This added knowledge allows the agent to deal with both the varying intersection structures found in a real-world network, as well as the additional vehicle/signal dynamics required when controlling real traffic. The various types of local information each agent is now responsible for is detailed in Sections 6.3.1 and 6.3.2. It should be noted that the intersection control agent still relies on only locally available data (whether through direct observation or local communication).

6.3.1 Constant Information

It is assumed within this work that the intersection agent knowledge presented in this section remains static. In the real world, this assumption generally holds true. In the event that this information did change, it would simply require an update to the state of the intersection control agent, which would reflect these changes immediately in the controlling algorithm. This is one advantage over a fixed strategy, that may require entirely new signal plans to deal with these changes (e.g., removal of a lane from an incoming edge).

- Network Edges (IE/OE): The intersection control agent is aware of both its incoming (IE) and outgoing (OE) edges within the network. This knowledge

includes the properties of those edges: length, number of lanes, speed limit and origin intersection.

- Turning Lanes (TL): In some cases within a real-world traffic network, several lanes of an intersection are assigned a dedicated phase to allow vehicles that may have trouble proceeding through an intersection to do so safely and easily. Within this work, the turning lanes (along with their phases) remain constant, and are derived from the data provided by the City of Ottawa. Turning lanes could, however, be modelled with an adaptable design, allowing the turning lane signal to be aborted in real time if no vehicles are present. This leaves more time to the competing flows at the intersection and increases the number of vehicles which can pass through the intersection during the cycle.

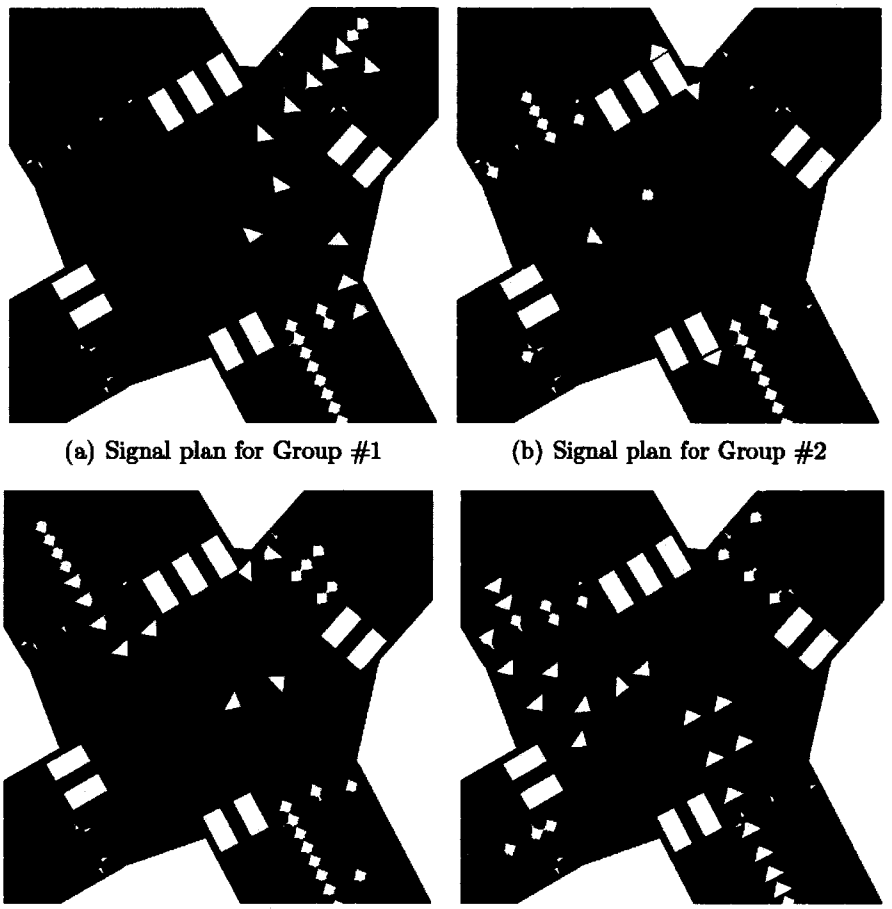
- Neighbours (N): Intersection control agents must be aware of their neighbouring intersections, as communication with neighbours is required for traffic state observation. Not only is the communication necessary for the basic calculation of traffic volume, but another interesting method of calculating traffic volumes through neighbour-to-neighbour communication is presented in Section 6.7.7.

- Groups (G): A group, in the domain of an intersection control agent, represents a set of compatible (non-opposing) traffic flows. Each intersection controls a number of groups, each of which has a related set of signals. This signal set is setup in such a way that it allows all flows within that particular group to proceed, while the others must stop (or at least yield, in the case of turning right during a red light). Once again, the groups used within this work were derived from those specified within the signal plans provided by the City of Ottawa. An example of a number of groups/signal plans is shown in Figure 6.1.

6.3.2 Observed and Calculated Information

Some information is not available through analysis and design of the traffic system. The information presented below consists of knowledge that must be observed or calculated by the intersection control agent.

Figure 6.1: Example of an intersection with four groups, showing the four corresponding signal sets



(a) Signal plan for Group #1

(b) Signal plan for Group #2

(c) Signal plan for Group #3

(d) Signal plan for Group #4

- Traffic Observations: The control algorithm requires traffic observations to calculate signal plans based on traffic volumes. These observations are made using sensors within the road network at the beginning and end of each edge (beginning sensor information is communicated from neighbour to neighbour). These observations are made at specific intervals (defined by a system parameter), and remain in the intersection control agent's memory for a specific time window (also a system parameter). As with the work described in Chapter 4, once the

time window has passed, the information is forgotten.

Another way of identifying the number of vehicles present is through the use of vehicle-to-intersection communication. Using this approach, the data produced by sensors (which can carry inaccuracy due to noise) is replaced with data communicated from the vehicles themselves (e.g., through smartphones or other electronic devices). This type of approach was not used here, but further discussion on this topic can be found in Section 8.2.3.

- Green Times: The agent calculates green times using the algorithm specified in detail within Algorithm 6. One green time must be specified for each of the intersection's groups. Each group's green time then represents the time within the intersection's cycle at which that group's signal set will be enacted.

6.4 System Parameters

There are a number of parameters used within the control system, each of which is explained below.

- Update Interval (UI): Used to determine how often each intersection agent should calculate a new signal plan. A shorter update interval means the intersection agent can change the signal plan more quickly in response to changing traffic volumes.

- Window Length (WL): Represents the length of time traffic observations are stored in memory. A better estimate of overall traffic volumes can be found using longer window lengths, as the larger number of observations smooths any outlying values. This allows the controlling agent to create a more accurate model of the traffic situation when creating signal plans. Older observations, on the other hand, can become irrelevant and contribute to poor signal plan generation. An optimal window length, then, would be long enough to smooth anomalous values while not being so long that the observations are no longer useful.

- Cycle Length (CL): The total length of time to be dedicated to the set of phases. Typically, shorter cycle lengths are more effective in low volume situations, while higher cycle lengths can alleviate problems found in high traffic volume situations. For this work, however, the cycle lengths specified by the City of Ottawa signal plans are used for each intersection throughout the day. These cycle length values vary depending on the current time of day, and also consist of different values for each intersection.

- Volume Calculation Method (VCM): This is the method used to calculate volume measurements based on the observed traffic data. Five separate volume calculation methods have been implemented and evaluated, all of which are described in more detail within Sections 6.7.3-6.7.7.

- Observed Data (OD): Three different methods of making vehicle observations have been compared. The first (NV) counts only the number of vehicles on an incoming edge, while the second (NVPL) measures the number of vehicles on an edge per unit of length. The third approach (SV) measures the total number of vehicles currently stopped on an edge.

- Observation Interval (OI): This parameter determines the number of simulation steps (seconds) between observations. For example, observations may be made every second, or they may be made every 10 seconds to decrease communication levels.

- Edge Balance (EB): With many groups consisting of multiple incoming edges, it may be beneficial to consider only the edge with a higher level of congestion (referred to as ME in future sections). This way the plan will allocate green time based on the busiest of the group's edges, instead of averaging the volumes equally. This may help reduce congestion in cases where the traffic volumes may vary widely between the edges within a group. Another value this parameter may take is $MESL$, which considers only the maximum edge within the group if that edge's volume is significantly larger (twice the size of) than the edge with the smallest volume measurement.

- Offset (OFF): The offset time of the intersection signal plan, representing the time within the cycle that the first phase will begin. As with the cycle length, offset values were taken from the data provided by the City of Ottawa. Discussion of dynamic generation of offset and cycle length is included in Section 8.2.1.

- Minimum Time (MT): The minimum amount of time to be assigned to each phase within the signal plan. This ensures that all traffic flows are allowed to move at least once per phase to avoid starvation. Throughout this work, the minimum time parameter is set to 8 seconds per phase, with 5 seconds of green time and a 3 second safety interval (which sets all lights to red) to clear the intersection.

Further to these parameters, which exist within all aspects of this work, two more parameters are necessary when using alpha-beta filtering to predict traffic volumes. Discussion of these two parameters is included in Section 6.7.6 which describes the alpha-beta filtering process in detail.

6.5 Key Algorithm Characteristics

There are a number of important characteristics possessed by the algorithm presented in this chapter. These characteristics, as well as their importance to traffic control, are described in the following sections.

6.5.1 Adaptive

As will be thoroughly documented in Chapter 7, adaptive control in the traffic domain offers a significant performance advantage when compared to static control methods. As would be expected, traffic volumes fluctuate constantly and an adaptive control mechanism is able to adjust to these fluctuations in real-time. In comparison, fixed timing signal plans fail to adjust based on current traffic information. Also, with a lack of sensor networks to generate observations regularly, fixed signal plans can be created based on data measured infrequently by human operators present at an

intersection. At the time of their use, these observations may be out of date, or non-representative of typical traffic volumes in the first place (in the case of an anomalous traffic situation during the observation period).

The adaptive nature of the algorithm presented here can also have economical benefits over fixed-timed counterparts. Bell and Bretherton (1986) has shown that the efficiency of fixed timing plans can degrade at a rate of 3% per year. To maintain the same level of efficiency, employees must be paid to travel into the traffic network and observe the traffic state first hand. An adaptive controller, however, bases decisions on current traffic levels and will operate with consistent effectiveness over time without any tuning.

6.5.2 Distributed Control

A distributed approach to traffic control is beneficial for a number of reasons. First, it allows for theoretically unlimited scalability, whereas a centralized system may struggle to maintain enough computing power for effective control as the network continually grows. This is especially important when considering real time control, where a centralized system may have to calculate new plans on a cycle-by-cycle basis. The latency inherent in a centralized system may make real-time control infeasible, as solutions generated may no longer be effective once the time required to generate them has passed. A distributed control scheme, on the other hand, requires each agent to generate only a single signal plan no matter how large the network becomes.

Second, a distributed approach offers a more robust solution, which can be extremely important in geographically dispersed real-time systems. For example, a power outage at a centralized control station may eliminate all network controlling abilities, reverting the signals to a basic default plan. The same power outage, however, is unlikely to effect all controlling agents within a distributed system, leaving some level of intelligent control intact. Agents in the immediate vicinity of any issue could also infer inputs for any neighbours experiencing problems, allowing reasonably intelligent control to be maintained. Also, if the problem is highly localized, the large majority of controlling agents will maintain the same level of performance that would be expected if no problems had occurred.

6.5.3 Local Computation and Communication

The local nature of the algorithm developed here is another possible implementation advantage. With only neighbour-to-neighbour communication, the architecture of the communications network used can remain quite simple, as intersection control agents can be connected to their neighbours using any short-range communication technology. In a centralized system which spans a large area, some of of these technologies (e.g., short-range wireless communication) may not be feasible as they are unable to send data the required distance.

Along with the distributed nature of the system, local communication increases the robustness and reliability of the entire network control system. Using wired communication (a common medium), a single break in the line may cut communication to all agents beyond the break. With local communication, however, a single break in the line can only affect communication between two agents, leaving all others unaware of any problem.

6.6 Intersection Control Agent Update Loop

This section describes the update process each intersection performs after each time step of a traffic simulation. During this update process, the agent may observe traffic, calculate new signal plans or do nothing. Algorithm 5 details the steps taken by a single intersection throughout the update process.

On the first line of the algorithm, the agent checks if the specified observation interval has passed. If it has, old observations must be removed from the observation list and new observations need to be made for each incoming edge (as can be seen from the loop on line 2). Lines 5-11 are responsible for storing either the number of vehicles, the number of vehicles per unit of length or the number of stopped vehicles for each lane on the edge (depending on the value of the observed data parameter). Data is stored for each lane (and not each edge) so turning lanes are able to maintain their own volume measurements, as they form groups that are separate to that of the rest of the edge. Once the stored traffic observations have been updated, the agent may, if the specified number of steps between update intervals has passed, calculate

a new signal plan for implementation (lines 15-17). Details of the algorithm used are included in Section 6.7.

```
   Input: i - Intersection to be updated
 1 if OI steps have passed since last observation then
 2     foreach Incoming edge e in IE_i do
 3         foreach Lane l of e do
 4             Remove observation with age older than WL from observation list
 5             if OD = NV then
 6                 Add number of vehicles on l to observation list
 7             else if OD = NVPL then
 8                 Add (Number of vehicles on l) / (Length of l) to observation list
 9             else if OD = SV then
10                 Add number of vehicles stopped on l to observation list
11             end
12         end
13     end
14 end
15 if UI steps have passed since last intersection update then
16     Update signal plan for i using Algorithm 6
17 end
```

Algorithm 5: Main update loop for an intersection control agent

6.7 Intersection Control Update Algorithm

The signal plans implemented at each intersection are created by the controlling agent using Algorithm 6. Figure 6.2 presents a flowchart detailing the process of signal plan calculation from available information. Red paths within Figure 6.2 represent information that is not adapted by the control algorithm presented here, but could be included in future extensions. Initially, traffic observations are generated from traffic sensor data and information available from neighbouring intersections. These observations, along with road network information, are used to calculate traffic volume measurements for all groups of the intersection. The traffic volume calculation is shown in a different colour, as a number of different algorithms (presented in Sections 6.7.3-6.7.7) have been developed to calculate the volume from the available observations. Many other approaches to traffic control use learning algorithms based on

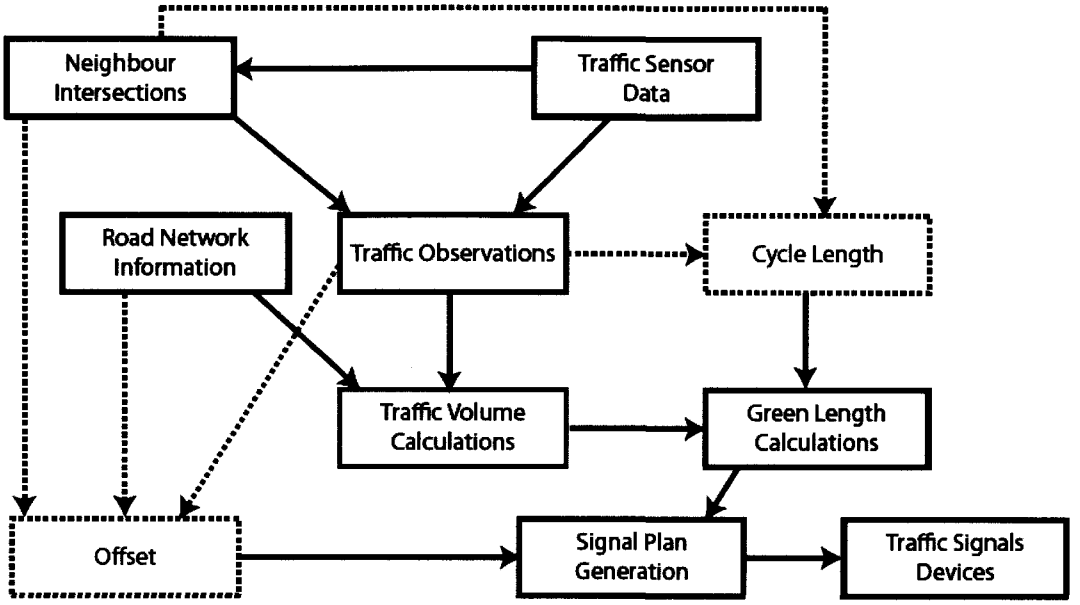
a fixed traffic network; these approaches, however, will suffer degraded performance in the event of network changes such as lane closures. The controlling agents detailed here, however, maintain a current view of the road network which could easily be modified when network changes are made. Including this information allows the controlling agents within the network to create more accurate models of the traffic situation, leading to more effective control decisions/signal plans. The calculated volumes are combined with the current cycle length to generate green lengths for each group. While this work uses fixed cycle lengths (as described in Section 6.4), adaptive cycle lengths could be generated, along with signal plans, based on available information pertaining to traffic and neighbour state. These adaptive cycle lengths could be used to improve intersection efficiency by increasing/decreasing the length of the cycle based on traffic volumes. Once green lengths for each group have been calculated, they are combined with an offset value to generate the signal plan, which can then be passed to the traffic signal control devices. As with cycle lengths, the offsets are not adapted by the control algorithm here, but future extensions to this work could generate offset values in real-time based on traffic state and information available about the road network and neighbouring intersections.

There are two main steps an agent must perform when creating a signal plan: volume calculation and timing calculation. A detailed description of these steps, in relation to the algorithm provided, is given in the following two sections.

6.7.1 Volume Calculation

Within Algorithm 6, lines 2-20 detail the steps taken to calculate a measurement of current (or predicted) traffic volume for each of the intersection's groups. To begin calculating a group's volume measurement, the agent first calculates a volume measurement for each edge included in the group (lines 5-14). These edge volume measurements are calculated as averages based on the number of lanes, to prevent a bias toward edges with a higher number of lanes (this volume measurement can be viewed as a saturation level for the entire edge). The actual volume calculation for each lane can be performed in a number of ways, explained in Sections 6.7.3-6.7.7. The volume measurement for each edge is stored in memory and also added to the

Figure 6.2: Flowchart detailing the process of signal plan calculation, with possible dynamic extensions included in red



group's total volume measurement. Once all edge volumes have been calculated, the group volume measurement is divided (as with the lanes of each edge) by the number of edges to prevent a bias toward groups with a larger number of edges. The agent also keeps track of the total volume measured by all groups (line 20), which is used later to calculate the proportion of the total intersection volume contributed by each group.

As mentioned briefly in Section 6.4, the edge balance (EB) parameter can specify how to calculate the final group volume. If the algorithm is set to balance the edges, the algorithm will set the group's volume to be that of the edge with the maximum volume (instead of an average over all edges). This prevents congestion buildup on an edge with higher volume than all other edges within its group, whereas averaging over all edges may result in too little green time dedicated to the most congested edge.

Input: i - The intersection to generate a plan for

1 $Total\_Cars = 0$
2 **foreach** *Group g in* G_i **do**
3 $AvgCars_g = 0$
4 $Num\_Edges = 0$
5 **foreach** *Incoming edge e of* IE_i **do**
6 **if** *e is part of the group g* **then**
7 $Edge\_Counts_{g,e} = 0$
8 $Num\_Lanes = 0$
9 **foreach** *Lane l of e not in TL* **do**
`/* CalculateVolume using a method specified in Sections 6.7.3-6.7.7 */`
10 $Edge\_Counts_{g,e} = Edge\_Counts_{g,e} + CalculateVolume(l)$
11 $Num\_Lanes = Num\_Lanes + 1$
12 $Num\_Edges_g = Num\_Edges_g + 1$
13 $Edge\_Counts_{g,e} = \frac{Edge\_Counts_{g,e}}{Num\_Lanes}$
14 $Avg\_Cars_g = Avg\_Cars_g + Edge\_Counts_{g,e}$
15 $Avg\_Cars_g = \frac{Avg\_Cars_g}{Num\_Edges_g}$
16 **if** $EB = ME$ **then**
17 $Avg\_Cars_g = max(Edge\_Counts_g)$
18 **else if** $EB = MESL$ *and* $max(Edge\_Counts_g) > 2 \times min(Edge\_Counts_g)$ **then**
19 $Avg\_Cars_g = max(Edge\_Counts_g)$
20 $Total\_Cars = Total\_Cars + Avg\_Cars_g$
21 $Total\_Len = 0$
22 **foreach** *Group g in* G_i **do**
23 $Proportion_g = \frac{Avg\_Cars_g}{Total\_Cars}$
24 $Time_g = round(Proportion_g \times CL)$
25 **if** $Time_g < MT$ **then**
26 $Time_g = MT$
27 $Total\_Len = Total\_Len + Time_g$
28 **while** $Total\_Len \mathrel{!}= CL$ **do**
29 **if** $Total\_Len < CL$ **then**
30 Chose random group g and add 1 to $Time_g$
31 $Total\_Len = Total\_Len + 1$
32 **else**
33 Chose random group g with $Time_g > MT$
34 Subtract 1 from $Time_g$
35 $Total\_Len = Total\_Len - 1$
36 $Cur\_Time = SelectOffset(i)$
37 **foreach** *Group g of i* **do**
38 $Green\_Switch_{i,g} = Cur\_Time$
39 $Cur\_Time = Cur\_Time + Time_g$

Algorithm 6: Green Length Calculation for a Single Intersection

6.7.2 Timing Calculation

Once volume measurements are available for each of the groups, a timing plan can be generated which assigns phase lengths based on current traffic volumes (lines 21-35). Using the volume measurement for each group, along with the total volume measured for all groups, a proportion of total volume can be assigned to each group (line 23). Multiplying this proportion by the total cycle length (line 24) assigns that proportion of the cycle time to each of the groups. These times are also bounded such that each group is given at least a minimum amount of green time each cycle (line 26).

With the rounding of times (since the agent is dealing with discrete time steps, all times must be integer values), the sum of green lengths attributed to all groups may not add exactly to the cycle length. For this reason, one second of green time is randomly added or subtracted (based on whether the cycle length is shorter or longer than the total time allocated) to a randomly selected group. This process is repeated until the total length allocated equals the cycle length (lines 28-35).

Finally, the agent can assign switching times within the cycle for each group, which specify the time at which the corresponding light phase should begin. Shown in lines 36-39, this process begins by selecting the offset value for the intersection (this is a preset value taken from the City of Ottawa's signal plans, but could be determined intelligently) and storing it as the current green switch time (line 36). The agent then iterates over all groups, setting the current group's green switch time to the current green switch time and increasing the current green switch time by the length of the group's phase (lines 37-39). The agent can then immediately begin to operate the intersection's signals based on the newly calculated signal plan.

6.7.3 Simple Average (SA)

The initial method of calculating volume measurements from observations was a simple average over the entire time window. While this method is easy to implement, it maintains no notion of time and thus cannot capture any temporal changes in traffic flow. Algorithm 7 contains the steps used in calculating the volume using this method.

1 $Total = 0$
2 **foreach** *j from 1 to* WL **do**
3 $\quad Total = Total + \frac{Num\_Vehs_{l,j}}{WL}$
4 **end**

Algorithm 7: Volume calculation on a lane l using a simple average over the time window

6.7.4 Time Sensitive Average (TS)

The time sensitive average (Algorithm 8) assigns decreasing weight to an observation as the observation's age increases. This weighting allows the volume calculation to be time-sensitive over the window, with older observations still having effect even, though the level of the effect is diminished.

1 $Total = 0$
2 **foreach** *i from 1 to* WL **do**
3 $\quad Total = Total + \frac{Num\_Vehs_{l,i} \times i}{WL}$
4 **end**

Algorithm 8: Volume calculation on a lane l using a time sensitive average over the time window

6.7.5 Unbiased Time Sensitive Average (UTS)

After closer analysis, the time sensitive average method of volume calculation carries with it a hidden bias. The observations made immediately previous to volume calculation will measure volumes on some edges which are currently given a green light, and some which are currently stopped by a red light. The vehicle counts on the edges with a red light then, may be much higher than the average if the light has stopped the flow for a significant period. Unfortunately, the time sensitive average explained above would give these measures the most weight within the calculation. The unbiased method, shown in Algorithm 9, associates the age of an observation with the number of cycles that have passed since it was observed. This way, all measurements within a single cycle will be weighted evenly, with observations from older cycles carrying less weight than those from more recent cycles.

1 $Total = 0$
2 $Cycles\_Kept = \lfloor \frac{WL}{Cycle\_Length} \rfloor$
3 **foreach** *i from 1 to* WL **do**
4 $\quad Cur\_Age = Cycles\_Kept - \lfloor \frac{i}{Cycle\_Length} \rfloor - 1$
5 $\quad$ **if** $Cur\_Age < 0$ **then**
6 $\quad\quad$ break
7 $\quad$ **end**
8 $\quad Total = Total + \frac{Num\_Vehs_{l,i}}{2^{2 \times Cur\_Age}}$
9 **end**

Algorithm 9: Unbiased, time sensitive calculating of a lane's volume over the time window

6.7.6 Alpha-Beta Filter (AB)

Alpha-beta filtering (Wikipedia, 2011) is a simple method used to minimize the effects of noisy measurements in situations where inconsistent data may be present. Alpha-beta filters use two state variables, which signify the current position and velocity of the measurement. Two additional system parameters are included within the system when using the alpha-beta filter:

- Alpha (ALPHA): Determines the proportion of the error to include when correcting the predicted position/value.
- Beta (BETA): Determines the proportion of the error to include when correcting the predicted velocity/rate of change.

Although the terms used may suggest otherwise, the filter can be applied to any system in which the first state (position) can be predicted based on the other state (velocity). Used here for traffic volume measurement, the position of the system represents the traffic volume measurement, while the velocity represents the rate of change of that measurement. Including information regarding the rate of change can allow for more accurate prediction of future traffic volumes, if the assumption that the rate of change will remain constant over a short period of time holds true.

Algorithm 10 presents the steps taken in applying an alpha-beta filter to traffic volume prediction. An update of the traffic prediction, using the specified algorithm, is completed every time an intersection requires new traffic volume measurements

for signal plan calculation The initial steps in using the alpha-beta filter (lines 1-4) require a calculation of the observed traffic volume, which can be done using any method of volume calculation (here using the method from Section 6 7 4) Once this measurement has been calculated, the algorithm can compute an error amount (the difference between the predicted volume and the measured volume) which is used to modify the velocity and the predicted value (lines 5-7) From these new values, the predicted velocity can be applied to the last prediction of traffic volume to create a newly predicted volume measurement (line 8), which is constrained to values greater than or equal to zero, as a volume less than zero is impossible

```
   /* All values used are initialized at the beginning of a
      simulation */
1  Total = 0
   /* Calculate the observed volume over the last interval */
2  foreach i from 1 to WL do
3      Total = Total + Num_Vehs_{l_i} × i / WL
4  end
5  Error = Total − Last_Predicted_Value
   /* Update the predicted value and velocity to include error */
6  Last_Predicted_Value = Last_Predicted_Value + (ALPHA × Error)
7  Last_Predicted_Vel = Last_Predicted_Vel + (BETA × Error)
   /* Create prediction for the next interval */
8  Last_Predicted_Value = Last_Predicted_Value + Last_Predicted_Vel
9  Last_Predicted_Value = max(Last_Predicted_Value, 0)
10 return Last_Predicted_Value
```

Algorithm 10: Calculation of a lane s volume using alpha-beta filtering

6.7.7 Neighbour Communicated Volumes (NCV)

The approaches to volume calculation specified in Sections 6 7 3-6 7 5 are purely reactive methods of calculating traffic volume This section proposes a new method which uses current traffic volumes, along with neighbouring intersection's estimations calculated using any of the above specified methods, to predict traffic volumes over a short period of time This method requires a slight change to intersection control agent behaviour instead of maintaining vehicle counts for incoming edges, control agents now maintain measures of how many vehicles exit the intersection onto each

lane over the time window. Using this information, the proportion of vehicles exiting the neighbouring intersection onto each lane can be calculated (line 2). The total number of vehicles on each incoming edge of the neighbouring intersection can then be multiplied by the proportion expected to come to the lane, creating an estimate of incoming traffic over a short-term period (lines 3-5). This prediction of incoming traffic is then added to the current number of vehicles on the lane (line 6) to get a short-term prediction of traffic volume.

1 $Total = 0$
2 $Neighbour\_Prop$ = *Proportion of vehicles leaving neighbour intersection and entering this lane*
3 **foreach** *Incoming edge e of neighbour* **do**
4 $\quad Total = Total + Number\ of\ vehicles\ on\ e \times Neighbour\_Prop$
5 **end**
6 $Total = Total + Current\ number\ of\ vehicles\ on\ lane$

Algorithm 11: Volume calculation for a lane using immediate observation and data communicated from neighbouring intersections

6.8 Summary

This chapter has outlined an adaptive and distributed algorithm which can be used to control real-world traffic. The chapter began by outlining the controlling agent model, including a description of the information each agent requires to perform signal plan updates (Section 6.3). Section 6.4 outlines the system parameters which determine the overall behaviour of the agents within the system, while Section 6.5 discusses the characteristics (distributed, adaptive and localized) which allow the algorithm to effectively control traffic in a scalable, robust way. Sections 6.6 and 6.7 then provided detailed explanations of the algorithms used for both traffic observation (Algorithm 5) and signal plan updating (Algorithm 6). Within Section 6.7, several algorithms which can be used to calculate traffic volumes are also outlined.

With the definition of a realistic traffic model completed (see Chapter 5) and the creation of an intelligent control algorithm presented here, the following chapter presents an investigation into the abilities of the algorithm to effectively control real-world traffic. This investigation includes results of optimization experiments for

the system parameters, as well as a comparison of the adaptive control approach's performance to that of a fixed signal control scheme.

Chapter 7

Experimental Results

7.1 Introduction

This chapter presents the experimental results generated through simulation of the realistic traffic model developed in Chapter 5. The chapter's first main goal is to present the findings from several parameter optimization experiments, which aim to find suitable values for the algorithm's system parameters. With suitable values selected, the adaptive control system outlined in Chapter 6 is used to control the traffic, with the results being compared to those produced using a fixed control scheme based on signal plans provided by the City of Ottawa. These results show that using an adaptive approach can result in more effective network control, as the fixed control scheme fails to handle unexpected traffic values effectively. Several examples are also included, which document situations in which the fixed control scheme failed to effectively control the traffic situation.

The entire chapter is outlined as follows. Initially, Section 7.2 explains the experimental setup used to evaluate the presented algorithm. Sections 7.3-7.4 then presents the findings of various experiments carried out to find suitable algorithm parameters. These sections include the results of each investigation, as well as a discussion of why these results were found in each case. Section 7.5 then presents a comparison of the adaptive control approach to a fixed-phase signal controller. This includes a description of the fixed plans, a comparison of the overall performance of both control approaches (Section 7.5.1) and example scenarios showing the failure of the fixed controller (Section 7.5.2).

7.2 Experimental Setup

Due to the stochastic nature of traffic control, a number of traffic scenarios had to be considered to make any meaningful conclusions. In all, fifteen (the number was chosen to balance the computation time requirements and the need for repetition) sets of vehicle routes were generated using the process outlined in Section 5.4.4. Each of these vehicle route scenarios were simulated for each control measure presented in this section and the results presented throughout this chapter are averages over all fifteen of these scenarios. Each scenario consisted of 39 600 time steps (each of which simulates one second), for a total of 11 hours of simulated time. Details on how the scenarios were generated from the available data is provided in Section 5.4.4, while the actual route specification files for use in SUMO are available on-line (McKenney, 2011).

Observations regarding traffic state were made at regular intervals (every 5 time steps, in general) and saved in output files for each simulation run. The observed information included the number of vehicles and average vehicle speed for street segments, entire streets and the entire simulation. This data was then parsed and compiled into the results included within this chapter and also in Appendix A.

7.3 Parameter Sensitivity Analysis

There are a number of parameters within the presented control algorithm which can affect the overall performance of the control system (for a detailed explanation of the parameters, see Section 6.4). The complex nature of the traffic control system proposed, however, makes it difficult to predict the various relationships between variables. Through simulation, however, the effectiveness of various values of each parameter can be examined. It would be computationally impractical, given the available hardware and sequential nature of current traffic simulation software, to simulate all scenarios for each possible combination of parameter values (a total of 118 125 simulation runs of approximately 1 hour each would be required to evaluate all combinations of the values used here). For this reason, the various values of each parameter were compared while suitable values for all other parameters were held

Table 7.1: Window Length Simulation Parameters

Window Length (WL)	60, 120, 300, 600, 900, 1800, 3600
Update Interval (UI)	120
Volume Calculation Method (VCM)	Unbiased Average
Observed Data (OD)	Number of Vehicles on Edge
Observation Interval (OI)	10 Steps
Edge Balance (EB)	Maximum Edge

constant, resulting in a total of 420 simulated scenarios. The following sections include results generated through the investigation of varying values for these important system parameters.

7.3.1 Window Length

The first important parameter to be investigated was the window length, which determines the length of time traffic observations are stored. In total, seven different window lengths were investigated, ranging from 60 steps (one minute) to 3600 steps (one hour). A list of all window lengths evaluated, as well as the other parameters used for each simulation is shown in Table 7.1.

After simulating the 15 scenarios for each of the window lengths, the average speed over the entire simulation was produced for each window length value. These averages, along with 1 standard deviation error bars, are presented in Figure 7.1. Using a paired two-tail T-test ($\alpha = 0.05$) it was found that all window length combinations but one (900/3600) resulted in statistically significant differences in average simulation speed. From Figure 7.1, however, it can be seen that the window length of 60 is the only value which resulted in a large performance decrease. Using a short time window (such as 60 seconds) forces the controlling agent to make decisions based on a smaller amount of information. This information, however, can be misleading in that it may represent only the most recent traffic flow, and not the overall traffic flow during the current time interval. Using a larger time window allows the algorithm to create a better view of the underlying traffic model. Due to the temporally sensitive nature of the volume calculation, the most recent observations (e.g., the last 60 seconds) are still weighted heavily using a longer window, but older observations also factor into the calculated volume measurement. This allows the algorithm to better match the

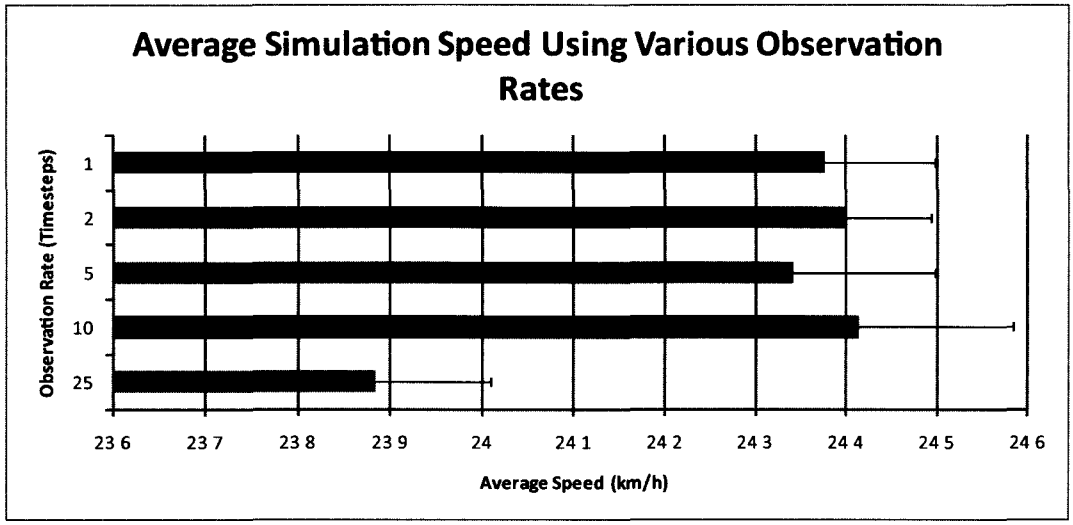
Table 7.2: Observation Interval Simulation Parameters

Observation Interval (OI)	1, 2, 5, 10, 25
Window Length (WL)	900
Update Interval (UI)	120
Volume Calculation Method (VCM)	Unbiased Average
Observed Data (OD)	Number of Vehicles on Edge
Edge Balance (EB)	Maximum Edge

Figure 7.2 shows the average simulation speed realized using the 5 parameter values investigated. Overall simulation speed is extremely similar when using observation intervals of 10 seconds or less, but begins to drop significantly when using an observation interval of 25 seconds. As with the window length evaluation above, a paired two-tail T-test ($\alpha = 0.05$) was used to test for a statistically significant difference between the different observation intervals. From these tests, it was found that there was no significant difference between any of the observation intervals from 1-10 seconds. The same tests, however, showed that using an observation interval of 25 seconds produced results that were significantly different from all of the other values tested. It is concluded then, that frequent measurement is required to infer an accurate traffic model; however, the system is capable of operating with measurements taken at 10 second intervals. The significant decay in performance when using an update interval of 25 seconds can be easily explained by analyzing a few properties of the road network and simulated traffic. The average edge length within the simulation is approximately 125m, while average simulation vehicle speeds can typically be around 25km/h (or approximately 7m/s). Considering these values, vehicles will traverse edges in an average of less than 20 seconds, which means the presence of these vehicles on edges they travel will often not be measured at all when using an update interval of 25 seconds. As with very small window lengths, this results in an inaccurate view of network state and poor controller performance.

When considering implementation, an estimate of the worst-case amount of data that must be communicated by an intersection within the modelled network can easily be generated. First, each observation for a single lane would require both a lane identifier and a vehicle count. Assuming both of these values can be represented as integers with a size of 4 bytes each, this would be a total of 8 bytes per lane per

Figure 7 2 Summary of observation interval parameter investigations showing average speed attained with 1 SD error bars



observation From the modelled network, the highest number of lanes on a single edge is 4, while the highest number of outgoing edges (which require observation communication) is also 4 This results in a total communication of $8 \times 4 \times 4 = 128$ bytes (or 1Kib) sent at each observation interval, which is feasible using a simple low-bandwidth communication framework Also, in the current implementation, all observations are sent at the same time which would require this maximum amount of bandwidth at a single time Observation updates could be staggered, improving the balance of work and decreasing the demand on the communication network (assuming a 10 second observation interval, each intersection would require a maximum bandwidth of 0 1Kib/s)

A further implementation of this algorithm could have intersection control agents which adaptively determine the optimal update interval based on current traffic conditions Agents could then find a balance between two objectives minimizing the bandwidth used in communicating observations and maximizing the performance of traffic signals within the network

Table 7.3: Observed Data Simulation Parameters

Observed Data (OD)	Number of Vehicles on Edge Stopped Vehicles on Edge Number of Vehicles per Unit Length
Window Length (WL)	900
Update Interval (UI)	120
Volume Calculation Method (VCM)	Unbiased Average
Observation Interval (OI)	10 Steps
Edge Balance (EB)	Maximum Edge

7.3.3 Observed Data

The Observed Data parameter specifies which measurement of volume should be used: total vehicles on an edge, number of vehicles on an edge per unit of length or number of stopped vehicles on an edge. To evaluate the performance of the control algorithm while observing these different measurements, each of the 15 vehicle route scenarios were simulated with the parameters found in Table 7.3.

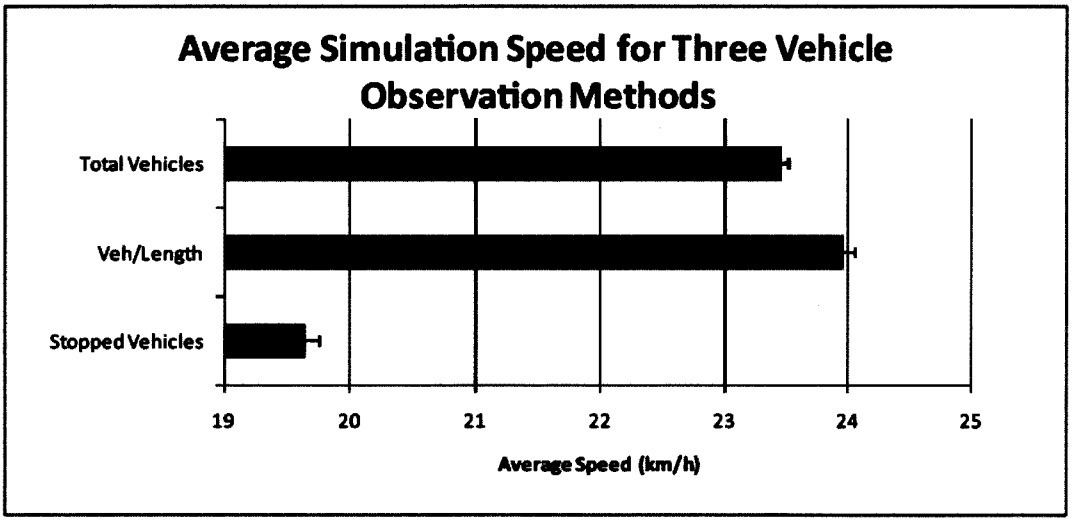
Figure 7.3 charts the performance difference found using the three types of observed data. It is obvious that using the number of stopped vehicles on an edge results in significantly slower average speeds than when using either of the other methods. Measuring only the stopped vehicles fails to measure the actual traffic volume, which relies on the number of approaching vehicles as well as the number of stopped vehicles. The performance decay seen when measuring only stopped vehicles would be most obvious in a situation where the intersection offset allows green waves to form in the dominant traffic direction. In this case, there very well may be 3 vehicles stopped in a direction with a small traffic volume, while no vehicles are stopped in the direction with a large vehicle volume due to successful offsetting. Devoting a larger proportion of time to the smaller traffic flow would not be wise, even though it may temporarily address the difference in stopped vehicles. While counting the total number of vehicles and the number of vehicles relative to the length of the edge have similar overall results, factoring edge length into the observation allows for a closer representation of congestion. For this reason, measuring the number of vehicles relative to the edge length resulted in a statistically significant increase in average simulation speed.

These results are advantageous from an implementation point of view, as vehicle

counts (which can be obtained by comparing sensor data from the beginning and end of a road section) are more readily available than information such as the number of vehicles currently stopped at an intersection.

Another possibility, which was not investigated here, is to use a linear combination of a number of the possible observations. This type of approach could use both the number of stopped vehicles and the number of vehicles approaching the intersection to infer a more accurate model of the current traffic situation. This improved model could possibly allow for improved controller agent performance. This investigation is left as future work.

Figure 7.3: Summary of observed data parameter investigations showing average speed attained with 1 SD error bars



7.3.4 Edge Balancing

As mentioned in Section 6.4, there may be cases in which a single edge within a group has a much higher volume than the average volume of the group. In this case, it may be beneficial to change the group's average to instead be the volume calculated for that specific edge. To test this hypothesis, three sets of simulations were performed: averaging all edges within a group, setting the group average to that of the highest

Table 7.4: Edge Balance Simulation Parameters

Edge Balance (EB)	Avg. All Edges, Max Edge if Significant Difference, Maximum Edge Always
Window Length (WL)	900
Update Interval (UI)	120
Volume Calculation Method (VCM)	Unbiased Average
Observed Data (OD)	Number of Vehicles on Edge
Observation Interval (OI)	10 Steps

edge volume if the highest edge volume is significantly (2 times) larger than the smallest edge volume and setting the group average to that of the highest edge value every time. A summary of all other parameters (which remained constant) is given in Table 7.4.

Figure 7.4 shows the average speeds attained over the 15 traffic simulations for each of the three edge balancing techniques investigated. While it can be seen that all three techniques performed similarly, with a maximum difference in average simulation speed of slightly more than 1 km/h, there was also a statistically significant difference between all three approaches (using a paired two-tailed T-test). It can also be seen from the figure that selecting the maximum edge only if it is much larger than the least edge volume results in a lower average speed than averaging all edges (with a decrease in speed of approximately 1.9%). Calculating a group's volume based on that group's highest edge volume on the other hand, resulted in an increase of slightly over 3%. This increase in average speed shows that there is a small advantage in placing priority on a busy edge within a group, instead of averaging all edges within a group equally. While the difference is small, this type of balancing will only have a noticeable effect in cases where the edges of a group have widely differing traffic volumes. No investigation was preformed to analyze the average edge difference within this model, but the effect of balancing should increase if the algorithm was applied in an environment where the edge difference within groups is significant on a regular basis.

Figure 7.4: Summary of edge balancing parameter evaluations showing average speed attained with 1 SD error bars

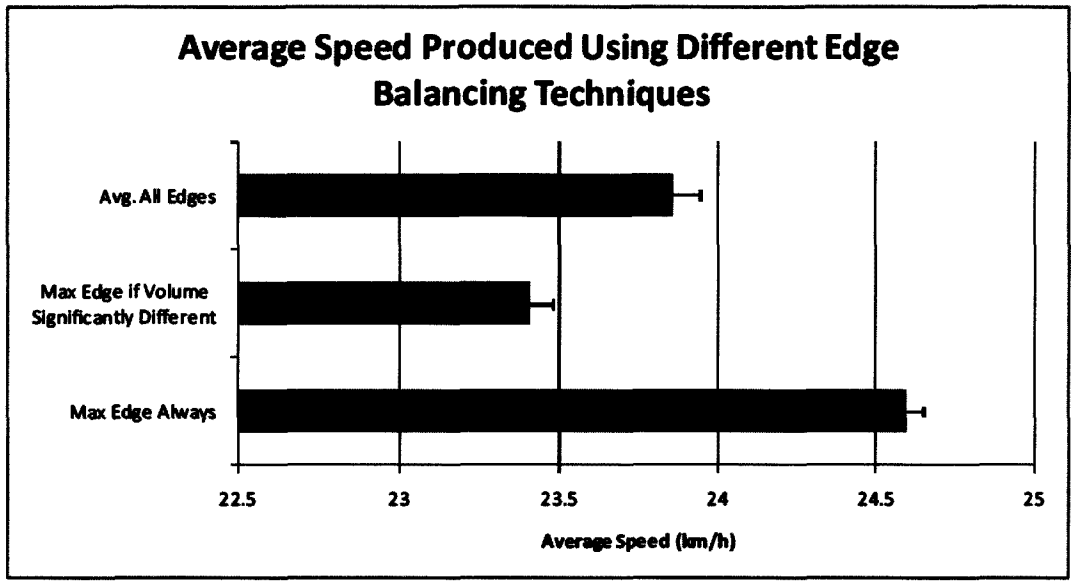


Table 7.5: Update Interval Simulation Parameters

Update Interval (UI)	120, 300, 600, 1200, Every Cycle
Observed Data (OD)	Number of Vehicles on Edge
Window Length (WL)	900
Volume Calculation Method (VCM)	Unbiased Average
Observation Interval (OI)	10 Steps
Edge Balance (EB)	Maximum Edge

7.3.5 Update Interval

The update interval parameter determines how often each intersection should calculate a new signal plan. This section presents an investigation into the effects this parameter may have on the overall performance of the control algorithm. A total of five different update interval values were compared through simulation, with the first four updating each intersection every 120, 300, 600 and 1200 timesteps respectively. The fifth parameter value required each agent to update its own signal plan at the beginning of each cycle (as the first phase begins). Table 7.5 lists the parameters used within the control algorithm when investigating these 5 values.

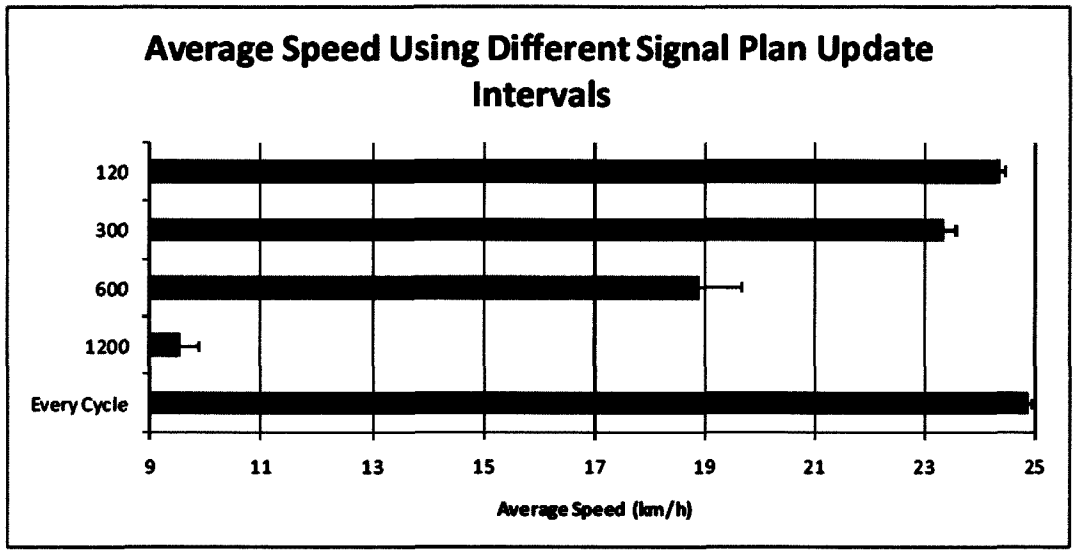
Figure 7.5 presents the average simulation speed attained using the various parameters evaluated. It can be seen from the first four values (120, 300, 600 and 1200) that there is obvious performance decay when increasing the time between signal plan updates. The final value, where agents update their plans at the beginning of each cycle, achieved an average speed even higher than the fixed values evaluated (slightly over 0.5km/h faster than an update interval of 120s). This advantage, however, may occur because a large number of cycle lengths within the City of Ottawa signal plans happen to be less than 120s. One conclusion that can be drawn from these results is that there is an obvious performance advantage when updating signal plans frequently. In fact, it was found that each update interval resulted in a statistically significant increase in average speed over the next highest update time (as with all previous parameter investigations, this hypothesis was tested with a paired two-tailed T-test). When waiting a prolonged period (10 minutes or greater) between signal plan updates, the observations used to generate the signal plan in use become invalid and traffic flows are no longer served effectively. With the update interval and overall performance having a seemingly inverse relationship, it may then be advantageous to instead eliminate the cycle altogether and implement a system which is capable of selecting from any available phase at any time. More discussion on this idea will be presented in Section 8.2.1.

7.3.6 Volume Calculation Method

This work has developed a number of ways in which traffic volume calculations can be made (outlined in Sections 6.7.3-6.7.7). This section includes an analysis of the overall control system performance when using these different methods. Table 7.6 details the parameters used when simulating the five different volume calculation methods. It should be noted that the alpha-beta filter method uses an alpha value of 0.55 and a beta value of 0.20. Details on how these parameters were chosen is given in Section 7.4.

A summary of the average simulation speed attained using each volume calculation method is presented in Figure 7.6. It can be seen within this figure that all volume calculation methods performed reasonably well, with less than 2km/h separating the

Figure 7.5: Summary of update interval parameter evaluations showing average speed attained with 1 SD error bars



best and worst average speed. Surprisingly, the method designed to avoid a bias toward the most recently stopped traffic flow performed worse than all of the other methods implemented. These results seem to imply that it may be beneficial to bias a signal plan toward the traffic flow that has been waiting at the time of signal plan calculation, as the other two reactive methods (simple average and time sensitive average) perform significantly better. It is interesting to note that volume calculation enabled through neighbour communication performs nearly as well as the method with the highest average speed. As can be seen in Section 6.7.7, the neighbour communicated volume predicts future volumes based on three pieces of information:

- The current number of vehicles on incoming edges.
- The current number of vehicles on incoming edges of neighbouring intersections.
- The estimated proportion of vehicles that enter the intersection's incoming edges from each incoming edge of each neighbour.

As proposed, this method only takes into account information regarding entire edges.

Table 7.6: Volume Calculation Method Simulation Parameters

Volume Calculation Method (VCM)	Simple Average (6.7.3) Time Sensitive Average (6.7.4) Unbiased Time Sensitive Average (6.7.5) Alpha-Beta Filter (6.7.6) Neighbour Communicated (6.7.7)
Update Interval (UI)	120
Observed Data (OD)	Number of Vehicles on Edge
Window Length (WL)	900
Observation Interval (OI)	10 Steps
Edge Balance (EB)	Maximum Edge

The calculation could be improved by taking into account available information pertaining to inter-lane connections. For example, all vehicles within a left turning lane must end up on the same edge, as they are unable to travel in any other direction. This method could then increase its predictive ability, which should lead to performance gains.

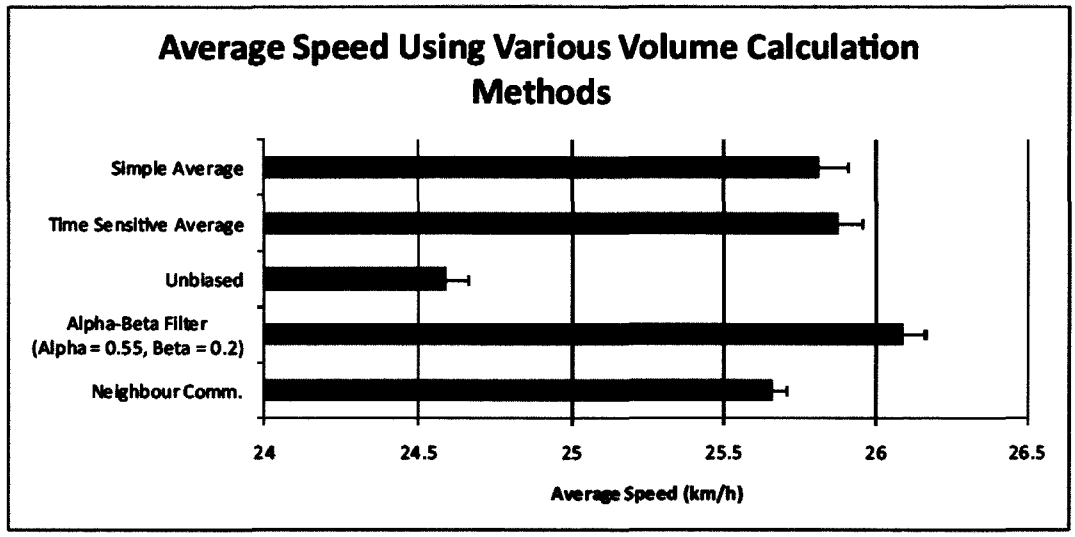
The highest average speed was attained using an alpha-beta filter to predict the future volumes based on previous measurements. This is an important result as it shows the need to not only create an estimate of current traffic volume, but to also consider the rate at which the traffic volume is changing.

7.4 Alpha-Beta Sensitivity Analysis

The alpha-beta filter requires the specification of both the alpha and beta parameter values. These values generally range from 0.0 to 1.0 and can greatly affect the overall prediction accuracy of the filter. By varying the two parameters within the simulation environment, a landscape can be generated which shows the overall effectiveness of the different parameter values. Figure 7.7 presents this landscape, showing average simulation speed for alpha and beta combinations ranging from 0.0-1.0 (in 0.05 increments).

From Figure 7.7, it can be seen that a wide range of alpha and beta values produce similar results. In fact, favourable results are produced in all cases where the alpha parameter's value is greater than 0.15. Response to change is typically realized more

Figure 7.6: Summary of average speed (with 1 SD error bars) attained using various volume calculation methods

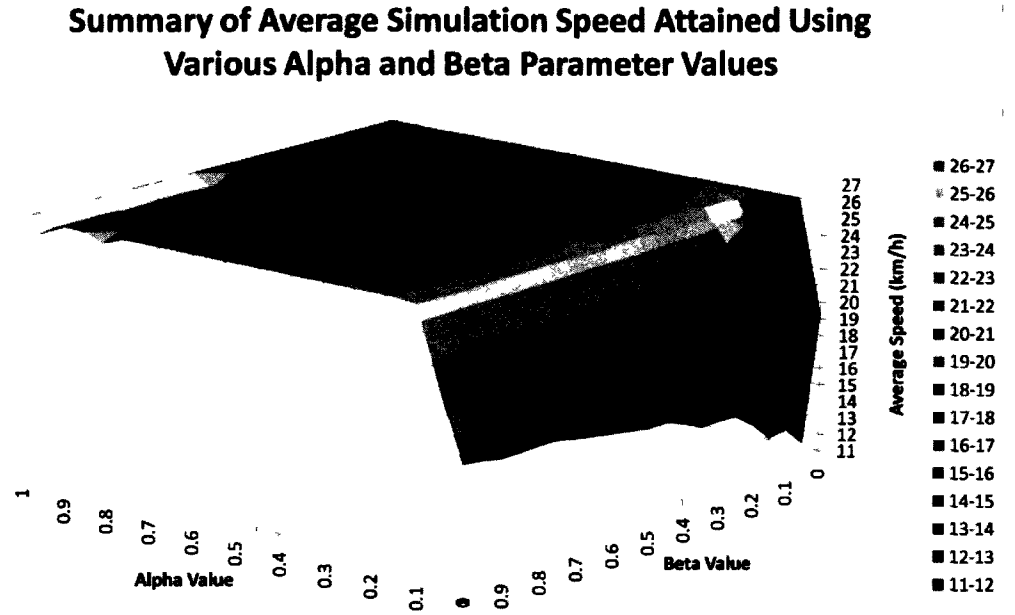


quickly when higher alpha and beta values are used. The fact that the best results are realized with higher alpha values once again shows that timely response to change is important in the traffic domain. After considering all evaluations, the highest average speed was attained using an alpha value of 0.55 and a beta value of 0.20. This result is used in Section 7.3.6 for comparison to other volume calculation methods.

7.5 Fixed City Plans vs. Adaptive Control

The most significant contribution of this work is the performance comparison of a simple, adaptive and distributed control system to that of fixed signal plans in traffic scenarios similar to those found in the real world. To investigate the performance differences, a set of fixed signal plans based on data supplied by the City of Ottawa was used. Each intersection is associated with a set of signal plans, along with a schedule specifying which signal plan to implement during different times throughout the day. Each signal plan within this set specifies the phase length for each phase at the intersection, as well as an offset value which determines when the first phase should begin. Table 7.7 shows an example signal plan for a single intersection, while

Figure 7.7: Average speed attained using various alpha/beta combinations



an example signal plan schedule is included in Table 7.8. The remainder of this section summarizes the performance of these fixed plans in comparison to that of the adaptive control approach presented within this work.

In selecting parameters to evaluate the adaptive control system, it would make sense to choose each parameter value based on the value that achieved the highest average speed through simulation in Section 7.3. It was found, however, that parameter values used in a previous experimental investigation resulted in better overall performance than when using this method. For this reason, the results presented

Table 7.7: Example signal plans for an intersection

Plan	AM Peak	Off Peak	PM Peak	Night	Weekend	PM Rush
Cycle	60	55	55	55	55	55
Offset	44	23	22	36	22	22
EW Thru	25	29	30	25	25	28
NS Thru	35	26	25	30	30	27

Table 7.8: Example weekday signal plan schedule for an intersection

Time	0:15	7:00	9:30	15:00	18:00	22:30
Plan	Night	AM Peak	Off Peak	PM Peak	PM Rush	Night

Table 7.9: System parameter values used when comparing to fixed signal plans

Volume Calculation Method (VCM)	Time Sensitive Average
Update Interval (UI)	120
Observed Data (OD)	Number of Vehicles on Edge
Window Length (WL)	3600
Observation Interval (OI)	10 Steps
Edge Balance (EB)	Maximum Edge

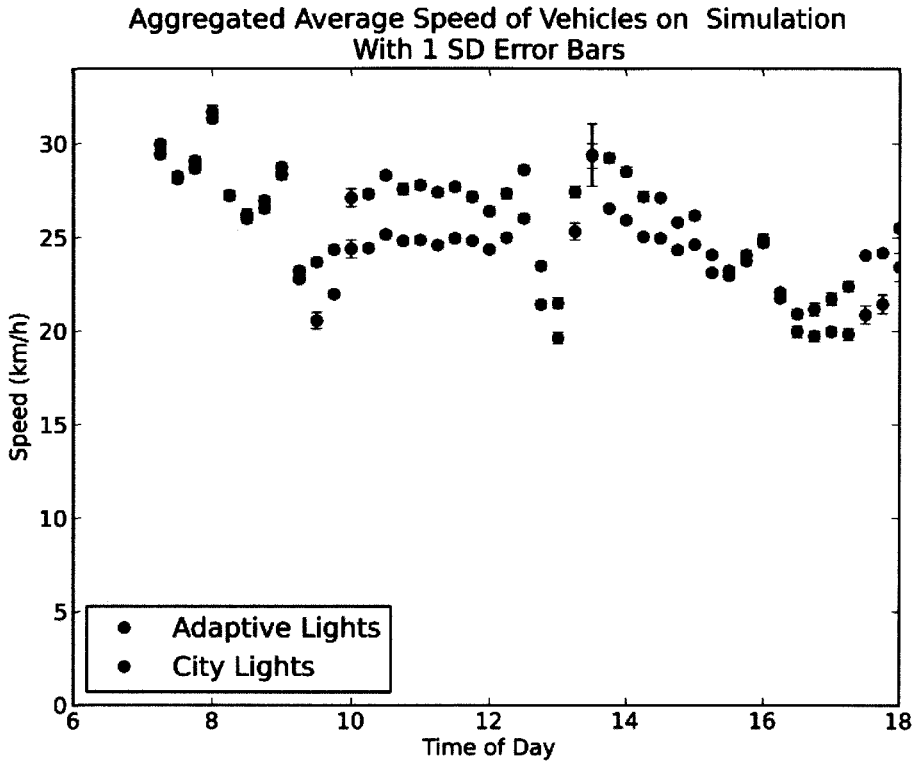
here use the parameter values found in Table 7.9.

7.5.1 Average Simulation Speed Comparison

Figure 7.8 presents a comparison of the average simulation speeds observed when using both the fixed City signal plans and adaptive signal plans. The measurements presented are aggregated over 15 minute intervals to present an average vehicle speed over each of the intervals. In the majority of the intervals within Figure 7.8, the use of adaptive signal plans results in increased average vehicle speed.

While it is easy to see from Figure 7.8 that adaptive control has a performance advantage over fixed signal plans, it is difficult to quantitatively compare the two using the provided data. Figure 7.9 presents a much more quantitative comparison, showing the overall percentage increase in vehicle speed for several time frames when using adaptive signal control. Using the approach described within this work, an increase in average speed of 6.59% is achieved over the entire simulated interval (7a.m.-6p.m.). Using the parameters specified here, however, an entire hour will be required to initialize the time window observations used to compute signal plans. If this first hour, in which the controlling agents are not able to infer an accurate traffic model, is not included within the calculation, the percentage increase in speed increases to 7.37%. The histogram presented in Figure 7.10 further quantifies the results from Figure 7.8, showing the number of 15 minute intervals which attained various percentage increases in speed. Figure 7.10 shows that only 2 intervals (of 44 in total) resulted in

Figure 7.8: Aggregated average simulation speed over 15 minute intervals for both fixed and adaptive lights



a lower speed when using the adaptive control system. It should once again be noted that one of these intervals was the initial 15 minutes of simulation, when the agents would not have enough information to accurately identify traffic volumes.

7.5.2 Examples of Fixed Signal Plan Failure

The largest problem found when using fixed signal plans arises when the actual traffic volume varies significantly from the expected traffic volume. This section presents several examples where the fixed signal plans failed to operate effectively during simulation. The performance of the fixed signal plans within these situations is compared to that of the adaptive control system to further show the advantage of adaptive traffic signal control.

Figure 7.9: Average increase in speed when using adaptive signal control

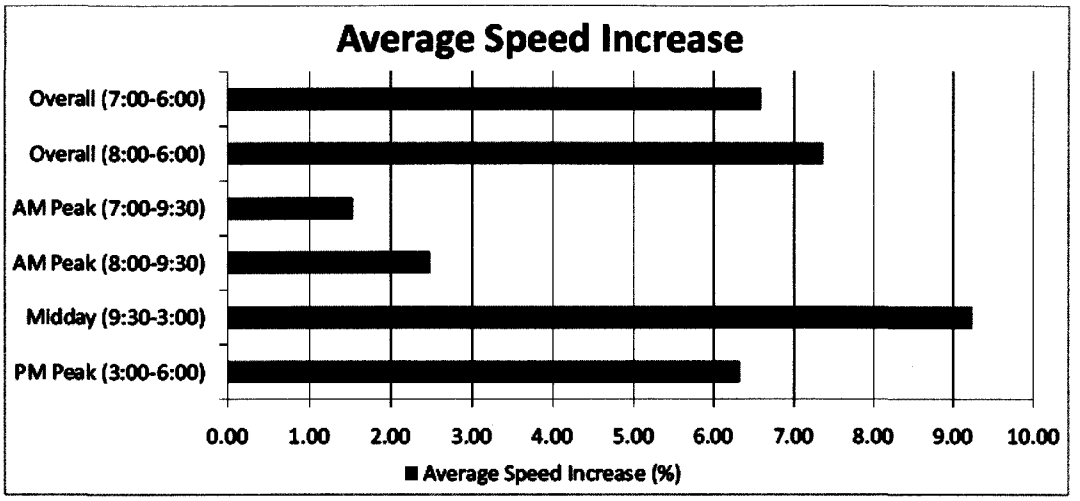
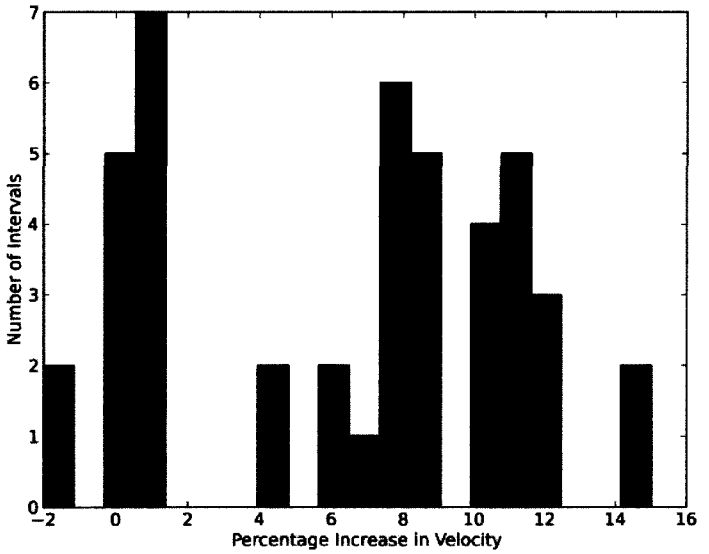


Figure 7.10: Distribution showing number of intervals with certain speed increases



The first example of fixed signal plan failure is presented in Figure 7.11. This figure shows the aggregated average vehicles (Figure 7.12(a)) and speed (Figure 7.12(b))

for a single road section within the simulation using both fixed and adaptive signal schemes. Between the times of 8a.m. and 10a.m., it is obvious from Figure 7.12(a) that there is an unexpected traffic volume which is not handled effectively by the fixed signal plans. The fixed plans fail to adapt to this unexpected volume, causing the total number of vehicles on the road segment to continually increase for a period of time, peaking at nearly 60 vehicles. The adaptive traffic signals, however, detect this unexpected volume and devote more signal time to the required phase, maintaining a much lower number of vehicles. The increase in vehicles when using the fixed signal plans results in a much slower average speed on the same road segment, as is evident in Figure 7.12(b). Once again, since the number of vehicles does not increase significantly when using adaptive signals, the speed on the road segment remains at approximately the same level throughout the interval.

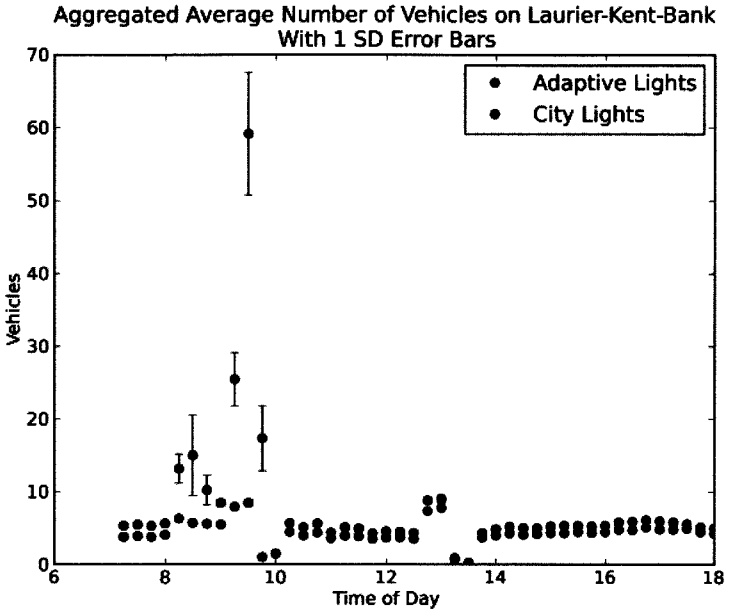
Another example, similar to the one above, is provided in Figure 7.12, this time showing average speed and number of vehicles for an entire street (16 road segments). In this example, the fixed signal plans produce very similar results to the adaptive signal control until the evening rush period (5p.m.). At this point, the increase in volume is not handled effectively by the fixed control scheme, causing the average number of vehicles on the street to increase significantly. In this case, the adaptive signal plans also experience an increased number of vehicles, but the increase is much smaller than that found using a fixed signal plan. As in the previous example, the increased number of vehicles results in a decrease in average vehicle speed, as shown in Figure 7.13(b). Similar to the number of vehicles, both control approaches experience a drop in average vehicle speed. The adaptive lights, however, maintain a higher average speed and also recover to the original speed faster than the fixed controller. Appendix A presents several more examples which further demonstrate the problems that may arise while using a fixed-time signal control approach.

7.6 Summary

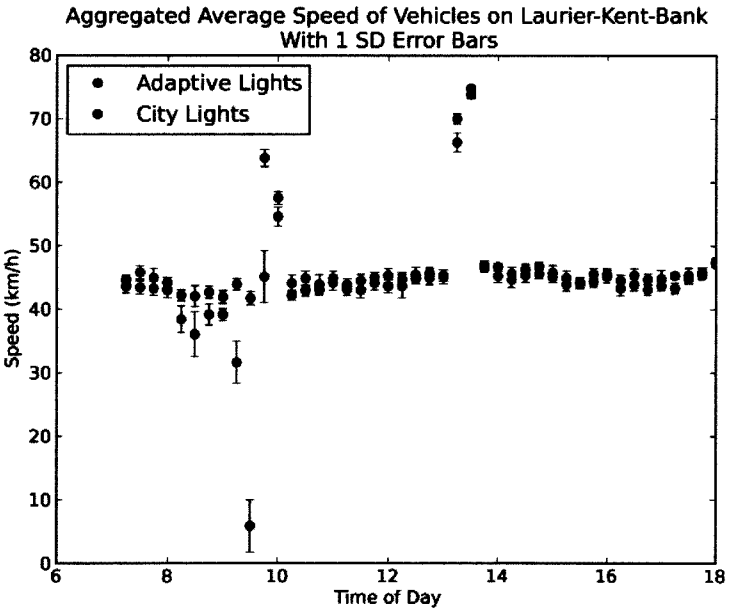
This chapter presented the results generated while investigating the properties of the control algorithm presented in Chapter 6. The first experiments carried out (Section 7.3) investigated the effects of various values for all of the algorithm's important

system parameters. Using a set of parameters that performed well, the algorithm's performance was compared to that of a fixed signal control schedule (Section 7.5). The fixed signal scheme itself was based on the signal plans used to control the area modelled by the City of Ottawa (an example plan can be found in Tables 7.7 and 7.8). Section 7.5.1 presented a comparison between fixed and adaptive control clearly showing that the adaptive approach was more versatile, robust and effective when controlling the signals within the traffic network (see Figures 7.8, 7.9 and 7.10). Finally, the chapter presented several specific examples, showing the problems that may arise when using a fixed control scheme in Section 7.5.2.

Figure 7.11: Aggregate average number of vehicles and speed on a single section of road using fixed and adaptive signal plans

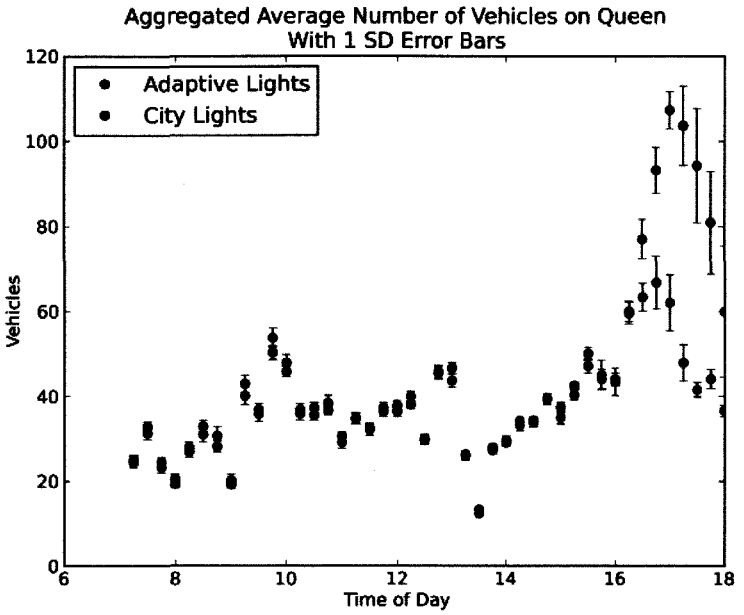


(a) Average vehicles on the road segment.

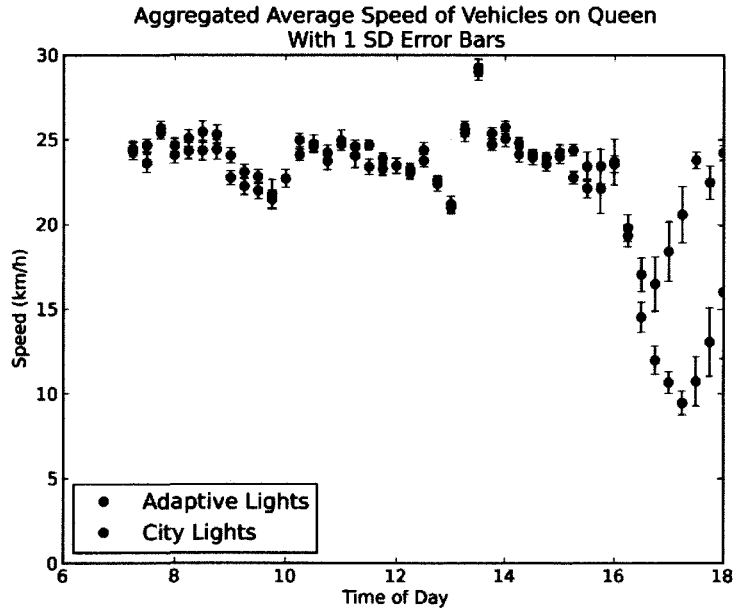


(b) Average vehicle speed on the road segment.

Figure 7.12: Aggregate average number of vehicles and speed for an entire street using fixed and adaptive signal plans



(a) Average vehicles on the street through the simulation.



(b) Average vehicle speed on the street.

Chapter 8

Conclusions and Future Work

This thesis is motivated by the need for increased investigation into intelligent control within vehicular traffic networks. As traffic volumes in major cities around the world continue to increase, the application of technological advancements in the area of traffic control will become more important. Through a review of the available intelligent traffic control research, several weak points have been identified. First, many control algorithms that have been proposed have been evaluated on very small and simplistic traffic networks, leaving their real-world applicability unclear. Also, many of the works presented perform investigations using simple or completely static traffic distributions. While it is acceptable to show that a control approach is capable of dealing with these types of scenarios, evaluating control systems using traffic distributions defined from real-world data is required in the future to show that these approaches are truly useful.

This thesis has addressed the above limitations as described in the contributions described in the following section.

8.1 Summary of Key Contributions

8.1.1 Review of Previous Intelligent Traffic Signal Control Work

Chapter 3 presented a review of the previous work completed in the area of intelligent traffic signal control. This included discussion of numerous approaches that have been applied in the traffic control domain, as well as a summary of the advantages and disadvantages inherent in the use of each approach. Using the identified advantages, a number of important characteristics for a real-time intelligent signal control system were outlined including adaptability, applicability to real-world scenarios, distribution of computation and reliance on locally available data.

8.1.2 Real-World Traffic Model

Within Chapter 5, a traffic model was developed within a realistic microscopic traffic simulator. Both the traffic network and vehicle volumes within that network were created using measurements taken from a real traffic network within Ottawa, Ontario. Not only is this traffic model used to evaluate the effectiveness of the algorithms developed within this thesis, but it can also be used in the future to investigate the performance of other control approaches. The process of model creation was also detailed within the same chapter, allowing others to develop further models more efficiently.

8.1.3 Algorithm Development and Experimentation

This thesis (Chapters 4 and 6 specifically) proposed a traffic control algorithm that had several important characteristics. First, the controlling agents make decisions based on current traffic volumes, allowing them to adapt signal plans within the network to efficiently control traffic flows. Second, the algorithm relies only on locally available information which adds robustness not seen in centralized control architectures. This local communication also requires a very small amount of bandwidth, allowing an implementation to use an inexpensive and simple communication infrastructure. This reliance on local information also allows the control system described here to scale to very large networks, which is yet another advantage over many of the previously developed control systems. The proposed control system also relies on a set of parameters which determine the overall functionality of the controlling agents and can be optimized to maximize system performance. While these parameters have been optimized for this specific use case, they could be further optimized in any other traffic environment, allowing the control system to be applied successfully in many locations with minimal intervention. Finally, the algorithm makes very few technological assumptions (e.g., there is no reliance on intelligent driver models) and has been evaluated within a simulation environment which modeled a real-world traffic network.

The algorithm outlined in Chapter 6 also relied on one of a number of possible volume calculation methods (outlined in Sections 6.7.3-6.7.7). These methods ranged

from purely reactive calculations to predictive methods which analyze the rate of change of volume measurements.

Chapter 6 experimented with the final algorithm, analyzing the control performance using a wide range of parameter values. After optimizing the parameters for this particular traffic network, the adaptive control system's performance was compared to that of a fixed control scheme based on signal plans made available by the City of Ottawa. It was shown that the adaptive algorithm controlled traffic within the network more efficiently than the fixed timing plans. The difference in performance was shown to be caused by an inability of the fixed traffic plans to address unexpected vehicle flows within the network. This conclusion further affirmed the need for real-time adaptive control within congested traffic networks.

8.2 Future Work

8.2.1 Intelligent Signal Control

There are several future directions for intelligent traffic signal control, involving both the algorithm presented here and other possible approaches. The algorithm presented should, in the future, be applied to a different traffic network than the one included within this thesis (one possible candidate is the traffic scenario generated for a section of the Greater Toronto Area MATSIM - Toronto (2011)). This type of work could investigate the effectiveness of the parameters, which have been optimized specifically using the network presented here, to efficiently control traffic elsewhere. Completing an investigation such as this would provide insight into the global applicability of the parameters, showing whether they are effective in more than one scenario or if they must be optimized in every application. Furthermore, evaluating the proposed algorithm on various other networks will demonstrate whether the adaptive approach is truly applicable in a number of traffic situations and not just the one used here.

Another extension, that may be applied to the presented algorithm or another control system, is the dynamic generation of cycle lengths and signal offsets. These two values have been shown to have a drastic effect on the overall performance of a signal plan, but no work on their real-time optimization has been completed here.

While the optimization of cycle length may be straightforward, choosing an offset value in a real-world traffic situation can be difficult. Much of this difficulty arises from the need to choose a single flow to offset with from a number of flows present at each neighbour. Questions also arise when deciding if an offset value should be set to maximize coordination with a single flow, or if an offset should be balanced to coordinate, to a certain degree, with a number of flows.

Another area of research involving intelligent signal control is the cooperation/ coordination between intersections within a network. While offset generation is an important part of coordination, controlling agents could also control phase and cycle lengths in a cooperative manner to avoid supersaturation at any one point within the network.

Finally, intelligent signal control may move away from a cycle-centric approach, with pre-calculated cycle lengths and phase orders. Instead, controlling agents at each intersection could decide at any given time which phase should be implemented, based on the current traffic situation. This allows control at an intersection to become even more dynamic, with phases that are unneeded at the time (such as advanced turning situations) being skipped completely.

8.2.2 Traffic Simulation and Modelling

Traffic modelling and simulation software also need to evolve continuously to enable traffic researchers to work efficiently and effectively. One future research area that promises to greatly increase efficiency is the parallelization of traffic simulation. While many of the traffic simulators available today are 'fast', they are often programmed for sequential execution. At the same time, there are a wide range of powerful parallelization approaches readily available that can greatly increase the speed at which simulations execute (e.g., multi-core computers, clusters, graphics processing units). In fact, GPU (graphics processing unit) computing (Owens et al., 2008) may be a particularly promising choice for traffic simulation, as the GPU devices are designed to be used in situations where the same operation must be executed a large number of times with different data (e.g., all vehicles in a network require their state to be updated). Using a single GPU device can allow program execution speeds that are

hundreds of times faster than a single computer, at a cost which is generally a fraction of the price of a desktop PC. This increase in execution speed would allow researches to generate results faster and investigate control systems more thoroughly.

In the future, microscopic traffic simulation may also move away from the car-following models used today (which were originally designed for single lane modelling) and begin to focus on autonomously controlled vehicles which are not bound in the same ways as vehicles using a car-following model (e.g., they do not have to be in a specific lane at any time, but instead can be between two lanes during a lane change). This approach would allow simulations to become more realistic and also help to further develop intelligent vehicle control algorithms.

Considering the amount of work required to develop the traffic model used within this thesis, automation of model creation would also be an area which could improve intelligent traffic systems research overall. This, of course, will involve cooperation between traffic researchers and traffic authorities in the development of systems which can transform the available traffic data (whether it is from sensors within the network or actual vehicle counts performed by employees) into a simulation-ready traffic model. It would also be beneficial if this work developed a standard for representation of traffic models, allowing the production of tools to convert the available model for use in any of the available simulation packages. With this standardization, a database of real-world traffic scenarios can be built and shared within the traffic research community. This type of database would allow traffic researchers to easily evaluate the performance of control strategies on a number of the publicly available datasets, enabling them to gain insight into the overall effectiveness of their control system. Several of these databases exist for other types of problems, such as the travelling salesman problem (TSPLIB, 2011) and pattern recognition (UCI Machine Learning Repository, 2011). Cooperation between researchers and traffic authorities could also lead to the implementation of a real-time model generation system. Using the data available from existing sensor networks within many urban environments, along with the communication infrastructure already in use by several traffic authorities, traffic models could be inferred which represent the current traffic state within the network. These models could be used to evaluate real-time traffic control systems

in development and could also be stored for later investigation of future systems.

Another future research area involves the inclusion of pedestrians within traffic models, as signal plans are often influenced by pedestrians within urban environments. While some work has already been completed in this area (Helbing et al. (2005)), little work has completed in the domain of intelligent traffic control. Future work will strive to include pedestrian flows within the realistic urban traffic models so the relationship between pedestrians and vehicles can be further investigated.

8.2.3 Intelligent Traffic Systems

While the efficiency of intersection control is an important determining factor in overall network performance, there is another part of the system that requires intelligent control: the drivers. Within a city, there are thousands (or hundreds of thousands) of drivers moving through the network during the day. These drivers are all making naive decisions as they are completely unaware of the traffic state anywhere except where they currently are. A large area of future research will involve providing useful information to the drivers (allowing them to make informed decisions) and also influencing the actual behaviour of drivers to improve network efficiency. This can be done by enabling both vehicle-to-network communication, in which information can be exchanged between vehicles and control devices within the network, and vehicle-to-vehicle communication, in which vehicles can share available information with each other. Passing data in such a way could allow drivers to be presented with an overall view of the current network state, allowing them to make informed decisions. Vehicle-to-network communication could also improve the way in which intelligent signal control functions within the network. First, vehicles could communicate directly with intersection controlling agents, eliminating the need for complex sensor networks. Also, vehicles can act as messengers which bring information from neighbouring intersections, eliminating the need for an interconnected intersection communication network. This, of course could make the system even more robust, as it is extremely unlikely that all vehicles would lose a message which would be lost if the communication link between intersections was broken in a more traditional communication approach.

Within the area of driver influence, dynamic vehicle routing is a particularly interesting area in which driver behaviour can be modified to decrease the average trip time within the network. While global positioning systems (GPS) can provide routes for drivers, these routes are typically generated using only the distance of the trip. An improved system could include real-time (or even predicted) traffic volumes which would allow the length, in time, of different routes to be estimated. Drivers can then be routed in a way that minimizes the trip time, and not just the distance covered. This type of approach would not only decrease trip time, but would also help to balance network traffic as vehicles would be routed around supersaturated network areas.

Bibliography

K. Almejalli, K. Dahal, and M.A. Hossain. Intelligent Traffic Control Decision Support System. In *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 688–701. Springer Berlin/Heidelberg, 2007a.

K. Almejalli, K. Dahal, and M.A. Hossain. GA-Based Learning Algorithms to Identify Fuzzy Rules for Fuzzy Neural Networks. *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*, pages 289–296, October 2007b.

K. Almejalli, K. Dahal, and M.A. Hossain. Real Time Identification of Road Traffic Control Measures. In *Advances in Computational Intelligence in Transport, Logistics, and Supply Chain Management*, volume 144 of *Studies in Computational Intelligence*, pages 63–80. Springer Berlin/Heidelberg, 2008.

K. Almejalli, K. Dahal, and M.A. Hossain. An intelligent multi-agent approach for road traffic management systems. *18th IEEE International Conference on Control Applications*, pages 825–830, July 2009.

AudiWorld. Audi, researchers at four U.S. universities begin work on solutions to urban mobility challenges. `http://www.audiworld.com/news/11/urban-mobility/`, August 2011.

P.G. Balaji and D. Srinivasan. Multi-Agent System in Urban Traffic Signal Control. *Computational Intelligence Magazine, IEEE*, 5(4):43–51, 2010.

G. Balan and S. Luke. History-based traffic control. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 616–621. ACM, 2006.

M.C. Bell and R.D. Bretherton. Ageing of fixed-time signal plans. In *Proceedings of the Second IEE Conference on Road Traffic Control*, 1986.

H.G. Beyer and H.P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford, 1999.

S. Chiu and S. Chand. Adaptive traffic signal control using fuzzy logic. In *Fuzzy Systems, 1993., Second IEEE International Conference on*, pages 1371–1376. IEEE, 1993a.

S. Chiu and S. Chand. Self-organizing traffic control via fuzzy logic. In *Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on*, pages 1897–1902. IEEE, 1993b.

Chih-hsun Chou and Jen-Chao Teng. A fuzzy logic controller for traffic junction signals. *Information Sciences*, 143(1-4):73–97, 2002.

M.C. Choy, R.L. Cheu, D. Srinivasan, and F. Logi. Real-time coordinated signal control using agents with online reinforcement learning. *Proc. 82nd Annual Meeting of the Transportation Research Board*, pages 1–21, 2003a.

M.C. Choy, D. Srinivasan, and R.L. Cheu. Cooperative, hybrid agent architecture for real-time traffic signal control. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 33(5):597–607, 2003b.

Seung-Bae Cools, C. Gershenson, and B. D'Hooghe. Self-Organizing Traffic Lights: A Realistic Simulation. In *Advances in Applied Self-organizing Systems*, Advanced Information and Knowledge Processing, pages 41–50. Springer London, 2008.

J. Cuena. Knowledge-based models for adaptive traffic management systems. *Transportation Research Part C: Emerging Technologies*, 3(5):311–337, October 1995.

B.C. da Silva, E.W. Basso, A.L.C. Bazzan, and P.M. Engel. Dealing with non-stationary environments using context detection. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 217–224, 2006.

D. de Oliveira and A.L.C. Bazzan. Swarm Intelligence Applied to Traffic Lights Group Formation. *VI Encontro Nacional de Inteligencia Artificial*, pages 1003–1012, 2007.

D. de Oliveira, P.R. Ferreira, A.L.C. Bazzan, and F. Klügl. A swarm-based approach for selection of signal plans in urban scenarios. *Ant Colony, Optimization and Swarm Intelligence*, 3172:143–156, 2004.

D. de Oliveira, A.L.C. Bazzan, B.C. da Silva, E.W. Basso, L. Nunes, R. Rossetti, E. de Oliveira, R. da Silva, and L. Lamb. Reinforcement learning based control of traffic lights in non-stationary environments: a case study in a microscopic simulator. In *Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS06)*, pages 31–42, 2006.

B. De Schutter, S.P. Hoogendoorn, H. Schuurman, and S. Stramigioli. A multi-agent case-based traffic control scenario evaluation system. In *Intelligent Transportation Systems, 2003. Proceedings. The IEEE 6th International Conference on*, volume 1, pages 678–683. IEEE, 2003.

K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the first international conference on Autonomous agents*, pages 404–412, 1997.

K. Dresner Multiagent traffic management: Opportunities for multiagent learning. *Learning and Adaption in Multi-Agent Systems*, 3898:129–138, 2006

K. Dresner and P. Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *AAMAS '04 Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, volume 2, pages 530–537. IEEE Computer Society, 2004.

K. Dresner and P. Stone. Multiagent traffic management: An improved intersection control mechanism. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 471–477. ACM, 2005.

K. Dresner and P. Stone. Traffic intersections of the future. *AAAI'06 proceedings of the 21st national conference on Artificial intelligence*, 2:1593–1596, 2006.

K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 31:591–656, 2008.

A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

N.V. Findler and J. Stapp. A Distributed Approach to Optimized Control of Street Traffic Signals. *Journal of Transportation Engineering*, 118:99–110, 1992.

J. France and A.A. Ghorbani. A multiagent system for optimizing urban traffic. *Intelligent Agent Technology, 2003. IAT 2003. IEEE/WIC International Conference on*, pages 411–414, 2003.

J. García-Nieto, A. Carolina Olivera, and E. Alba. Swarm intelligence for traffic light scheduling: Application to real urban areas. *Engineering Applications of Artificial Intelligence*, 2011.

C. Gershenson. Self-organizing Traffic Lights. *Complex Systems*, 16:29–53, 2005.

Google Inc. Google Maps with Street View. `http://maps.google.com/help/maps/streetview/`, July 2011.

A. Hegyi, B. De Schutter, S. Hoogendoorn, R. Babuska, H. van Zuylen, and H. Schuurman. A fuzzy decision support system for traffic control centers. *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, pages 358–363, 2001.

D. Helbing, R. Jiang, and M. Treiber. Analytical investigation of oscillations in intersecting flows of pedestrian and vehicle traffic. *Phys. Rev. E*, 72(4):046130, 2005.

T.H. Heung, T.K. Ho, and Y.F. Fung. Coordinated road-junction traffic control by dynamic programming. *Intelligent Transportation Systems, IEEE Transactions on*, 6(3):341–350, 2005.

J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, Ann Arbor, 1975.

R. Hoyer and U. Jumar. An advanced fuzzy controller for traffic lights. *Annual Review in Automatic Programming*, 19:67–72, 1994.

L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

J. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection).* The MIT Press, 1992.

S. Lammer and D. Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 4, 2008.

J.H. Lee and H. Lee-Kwang. Distributed and cooperative fuzzy controllers for traffic intersections group. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 29(2):263–271, 1999.

M.J. Lighthill and G.B. Whitham. On kinematic waves. I. Flood movement in long rivers. *Proc. R. Soc. Lond. A*, 229(1178):281–316, 1955.

MATSIM - Toronto. `http://www.matsim.org/scenario/toronto`, July 2011.

D. McKenney. Sumo routing files. `http://sikaman.dyndns.org:8888/index.php?page=traffic-project`, August 2011.

M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, 1998. ISBN 0262631857.

D.J. Montana. Strongly typed genetic programming. *Evolutionary computation*, 3 (2):199–230, 1995.

D.J. Montana and S. Czerwinski. Evolving control laws for a network of traffic signals. In *GECCO '96 Proceedings of the First Annual Conference on Genetic Programming*, pages 333–338. MIT Press, 1996.

MoreVTS. `http://sourceforge.net/projects/morevts/`, April 2011.

Y. Murat and E. Gedizlioglu. A fuzzy logic multi-phased signal control model for isolated junctions. *Transportation Research Part C: Emerging Technologies*, 13(1): 19–36, February 2005.

J. Niittymaki and M. Pursula. Signal control using fuzzy logic. *Fuzzy sets and systems*, 116(1):11–22, 2000.

OpenStreetMap. `http://www.openstreetmap.org`, February 2011.

J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. GPU Computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang. Review of Road Traffic Control Strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

C.P. Pappis and E.H. Mamdani. A Fuzzy Logic Controller for a Trafc Junction. *Systems, Man and Cybernetics, IEEE Transactions on*, 7(10):707–717, 1977.

L.S. Passos and R. Rossetti. Traffic Light Control Using Reactive Agents. In *Information Systems and Technologies (CISTI), 2010 5th Iberian Conference on*, pages 1–6. IEEE, 2010.

M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons, Inc., 1994.

A. Rathi. A control scheme for high density sectors. *Transportation Research Part B: Methodological*, 22(2):81–101, 1988.

C.H. Reinsch. Smoothing by spline functions. *Numerische Mathematik*, 10(3):177–183, October 1967.

D.I. Robertson and R.D. Bretherton. Optimizing networks of traffic signals in real time-the SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1): 11–15, 1991.

J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 475–482. ACM, 2002.

J. Sánchez, M. Galán, and E. Rubio. Genetic algorithms and cellular automata: A new architecture for traffic light cycles optimization. In *Evolutionary Computation, 2004. CEC2004. Congress on*, volume 2, pages 1668–1674. IEEE, 2004.

J. Sánchez, M. Galán, and E. Rubio. Applying a Traffic Lights Evolutionary Optimization Technique to a Real Case: "Las Ramblas" Area in Santa Cruz de Tenerife. In *IEEE Transactions on Evolutionary Computation*, volume 12, pages 25–40. IEEE, 2008.

J. Sánchez, M. Galán, and E. Rubio. Traffic signal optimization in "la almozara" district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. *Intelligent Transportation Systems, IEEE Transactions on*, 11:132–141, March 2010.

N. Schurr, J. Marecki, M. Tambe, P. Scerri, N. Kasinadhuni, and J. Lewis. The future of disaster response: Humans working with multiagent teams using DEFACTO. In *AAAI Spring Symposium on AI Technologies for Homeland Security*, 2005.

J.C. Spall and D.C. Chin. A model-free approach to optimal signal light timing for system-wide traffic control. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, pages 1868–1875. IEEE, 1994.

J.C. Spall and D.C. Chin. Traffic-responsive signal timing for system-wide traffic control. *Transportation Research Part C: Emerging Technologies*, 5(3-4):153–163, August 1997.

D. Srinivasan, M.C. Choy, and R.L. Cheu. Neural Networks for Real-Time Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, 7(3): 261–272, September 2006.

M. Steingröver, R. Schouten, S. Peelen, E. Nijhuis, and B. Bakker. Reinforcement learning of traffic light controllers adapting to traffic congestion. In *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2005)*, pages 216–223, 2005.

Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S. Tadaki, and S. Yukawa. Traffic jams without bottlenecks: experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10, 2008.

SUMO Features. `http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Sumo_at_a_Glance`, June 2011.

SUMO Traffic Simulator. `http://sumo.sourceforge.net`, July 2011.

G. Theraulaz, E. Bonabeau, and J.N. Denuebourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society B: Biological Sciences*, 265(1393):327–332, February 1998.

TSPLIB. `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`, July 2011.

UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml/`, July 2011.

M. Vasirani and S. Ossowski. A market-inspired approach to reservation-based urban road traffic management. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 617–624. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

VISSIM. `http://www.vissim.com/`, April 2011.

A. Vogel, C. Goerick, and W. Von Seelen. Evolutionary algorithms for optimizing traffic signal operation. In *Proceedings of the European Symposium on Intelligent Techniques*, 2000.

W. Wei and Y. Zhang. FL-FN based traffic signal control. In *Fuzzy Systems, 2002. FUZZ-IEEE'02. Proceedings of the 2002 IEEE International Conference on*, pages 296–300, 2002.

W. Wei, Y. Zhang, J.B. Mbede, Z. Zhang, and S. Jingyan. Traffic signal control using fuzzy logic and MOGA. *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, 2:1335–1340, 2001.

M.P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

M.A. Wiering. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1151–1158, 2000.

M.A. Wiering, J. Van Veenen, J. Vreeken, and A. Koopman. Intelligent traffic light control. *ERCIM News, European Research Consortium for Informatics and Mathematics*, 53:40–41, 2003.

M.A. Wiering, J. Vreeken, J. Van Veenen, and A. Koopman. Simulation and optimization of traffic in a city. *Intelligent Vehicles Symposium, 2004 IEEE*, pages 453–458, 2004.

Wikipedia. Alpha-Beta Filter. `http://en.wikipedia.org/wiki/Alpha_beta_filter`, June 2011.

U. Wilensky. NetLogo. `http://ccl.northwestern.edu/netlogo`. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999.

U. Wilensky. NetLogo Traffic Grid model. `http://ccl.northwestern.edu/netlogo/models/TrafficGrid`. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 2003.

M.J. Wooldridge. *An introduction to multiagent systems*. John Wiley and Sons, 2002.

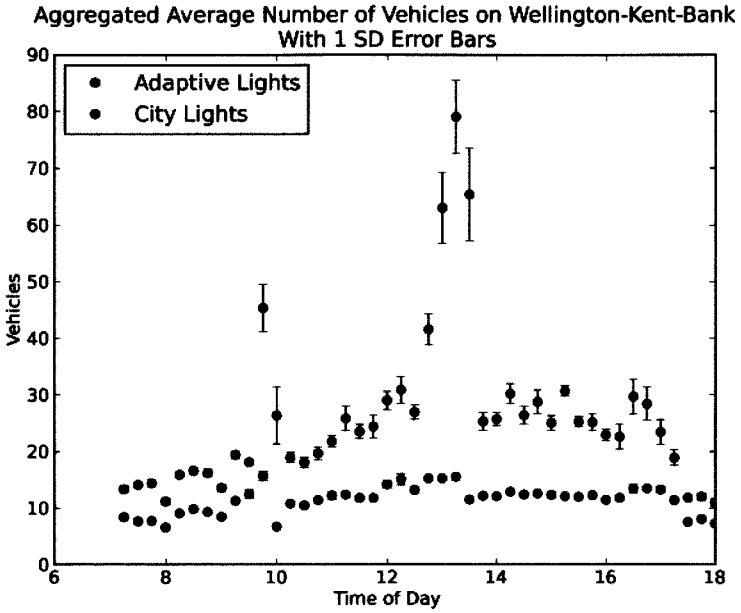
L.A. Zadeh. Fuzzy logic: computing with words. *Fuzzy Systems, IEEE Transactıons on*, 4(2):103–111, 1996.

Appendix A

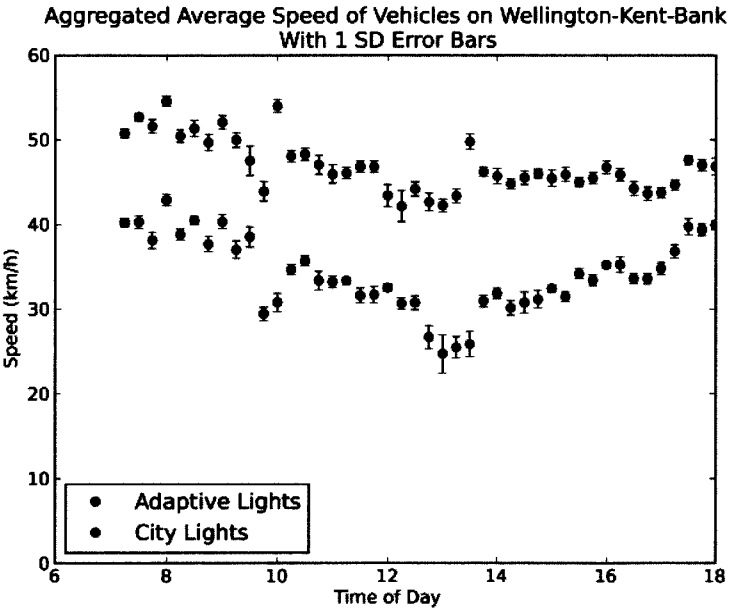
Further Examples of Fixed vs. Adaptive Signal Plans

This appendix includes a number of figures which further demonstrate the performance advantaged of adaptive signal over a fixed traffic light controller. Each of the figures included here present both the average vehicle counts and average vehicle speed aggregated over 15 minute intervals for specific areas of the network. Also, each figure further shows the problematic nature of a fixed signal controller, as the number of vehicles rise dramatically in a number of cases. Furthermore, it can be seen that while the adaptive algorithm can also experience decreased traffic flow during high volume scenarios, it is capable of recovering from these situations quicker than a fixed control approach.

Figure A.1: Aggregate average number of vehicles and speed for Wellington between Kent and Bank

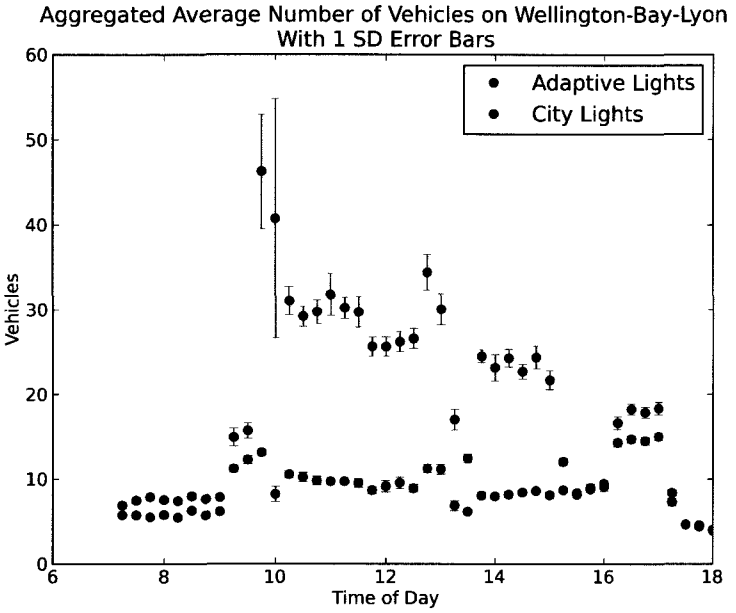


(a) Average vehicles

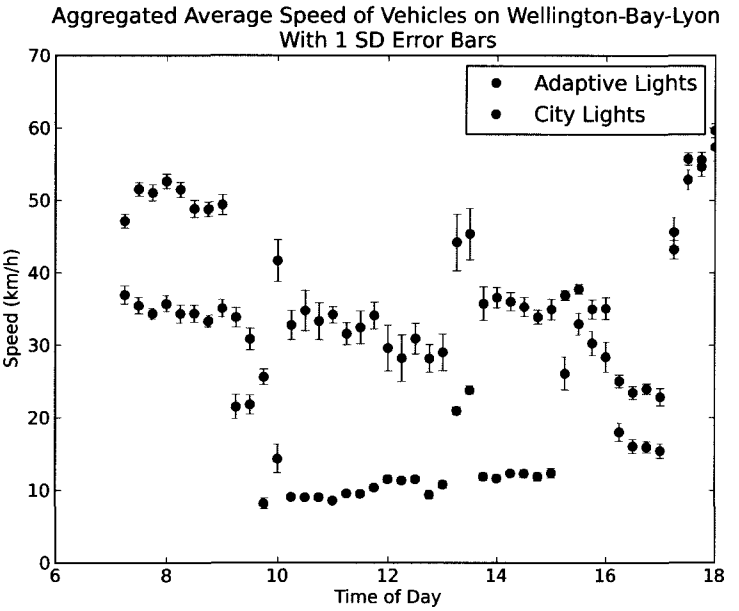


(b) Average vehicle speed

Figure A.2: Aggregate average number of vehicles and speed for Wellington between Bay and Lyon

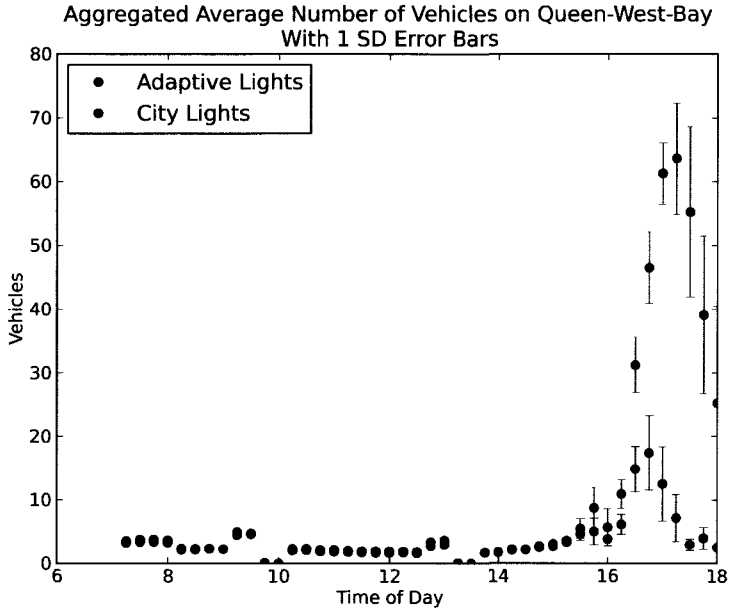


(a) Average vehicles

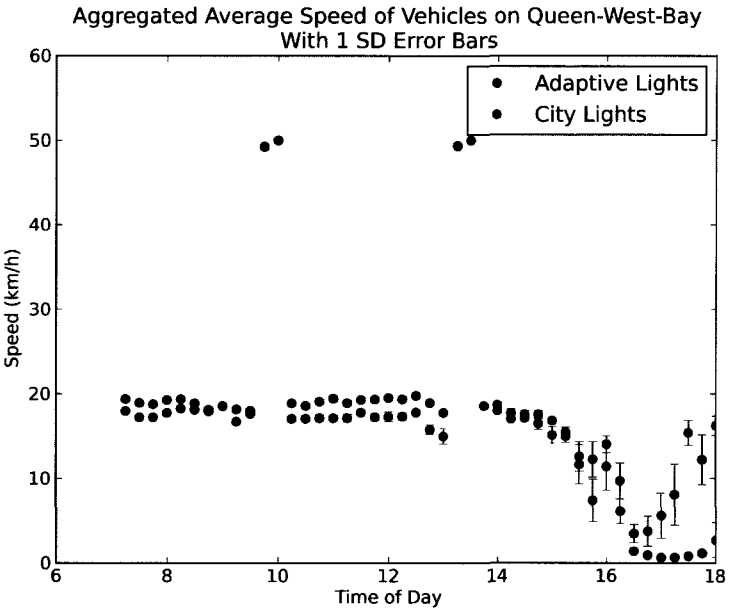


(b) Average vehicle speed

Figure A.3: Aggregate average number of vehicles and speed for Queen between West and Bay

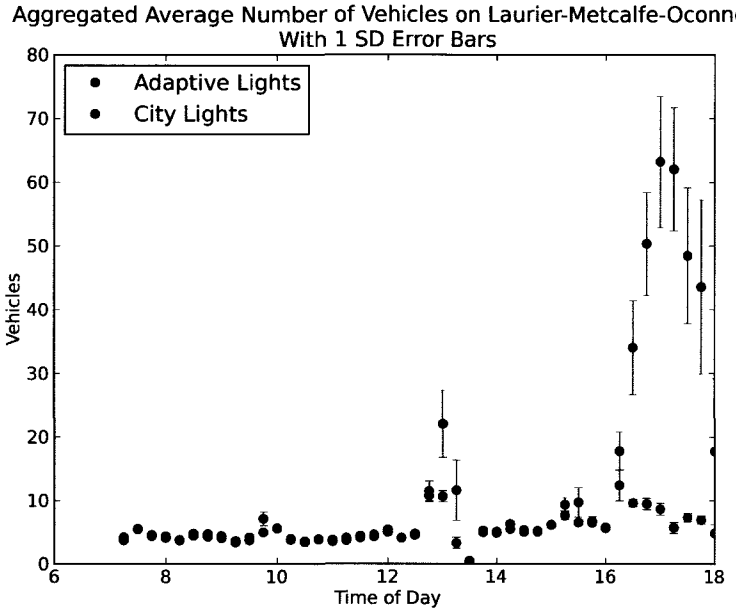


(a) Average vehicles

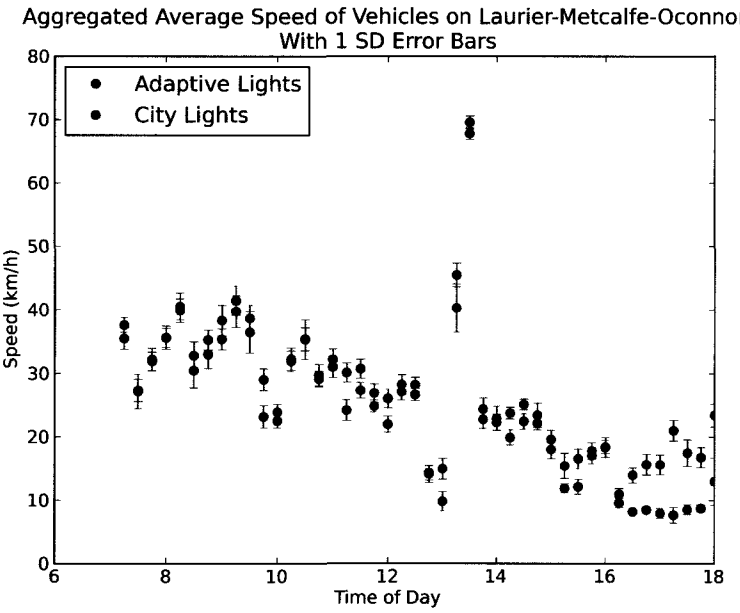


(b) Average vehicle speed

Figure A.4: Aggregate average number of vehicles and speed for Laurier between Metcalfe and O'Connor

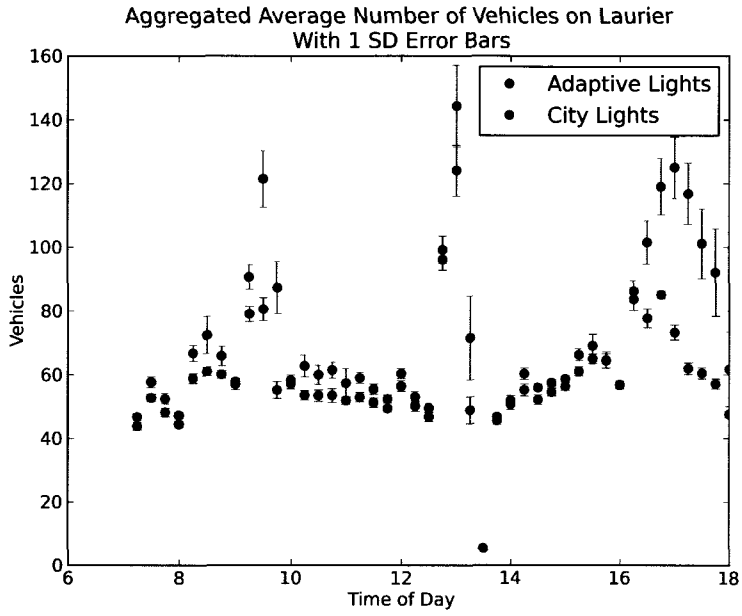


(a) Average vehicles

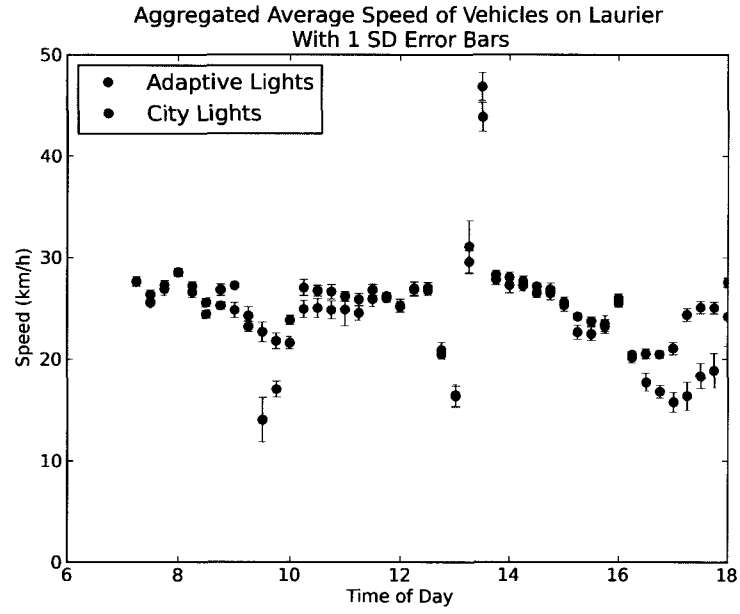


(b) Average vehicle speed

Figure A.5: Aggregate average number of vehicles and speed for the entirety of Laurier Ave.



(a) Average vehicles



(b) Average vehicle speed