Deep Reinforcement Learning Models for Real-Time Traffic Signal Optimization with Big Traffic Data

by
Matthew Muresan

A Thesis
presented to the University Of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Civil Engineering

Waterloo, Ontario, Canada, 2021

EXAMINING COMMITTEE MEMBERSHIP

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

- External Examiner: Dr. Henry Liu, Professor, Civil and Environmental Engineering, University of Michigan
- Supervisor: Dr. Liping Fu, Professor, Civil and Environmental Engineering, University of Waterloo
- Internal Member: Dr. Bruce Hellinga, Professor, Civil and Environmental Engineering, University of Waterloo
- Internal Member: Dr. John Quilty, Professor, Civil and Environmental Engineering, University of Waterloo
- Internal-External Member: Dr. Jun Liu, Professor, Applied Mathematics, University of Waterloo

DECLARATION

ABSTRACT

One of the most significant changes that the globe has faced in recent years is the changes brought about by the COVID19 pandemic. While this research was started before the pandemic began, the pandemic has exposed the value that data and information can have in modern society. During the pandemic traffic volumes changed substantially, leaving the inefficiencies of existing methods exposed. This research has focussed on exploring two key ideas that will become increasingly relevant as societies adapt to these changes: Big Data and Artificial Intelligence.

For many municipalities, traffic signals are still re-timed using traditional approaches and there is still significant reliance on static timing plans designed with data collected from static field studies. This research explored the possibility of using travel-time data obtained from Bluetooth and WiFi sniffing. Bluetooth and WiFi sniffing is an emerging Big Data approach that takes advantage of the ability to track and monitor unique devices as they move from location to location. An approach to re-time signals using an adaptive system was developed, analysed, and tested under varying conditions. The results of this work showed that this data could be used to improve delays by as much as 10% when compared to traditional approaches. More importantly, this approach demonstrated that it is possible to re-time signals using a readily available and dynamic data source without the need for field volume studies.

In addition to Big Data technologies, Artificial Intelligence (AI) is increasingly playing an important role in modern technologies. AI is already being used to make complex decisions, categorise images, and can best humans in complex strategy games. While AI shows promise, applications to Traffic Engineering have been limtied. This research has advanced the state-of-the art by conducting a systematic sensitivity study on an AI technique, Deep Reinforcement Learning. This thesis investigated and identified optimal settings for key parameters such as the discount factor, learning rate, and reward functions. This thesis also developed and tested a complete framework that could potentially be applied to evaluate AI techniques in field settings. This includes applications of AI techniques such as transfer learning to reduce training times. Finally, this thesis also examined framings for multi-intersection control, including comparisons to existing state-of-the art approaches such as SCOOT.

ACKNOWLEDGEMENTS

I would like to express sincere thanks and appreciation to my supervisor, Dr. Liping Fu, who has provided guidance, motivation and substantial help in the development of my research and research ideas. Dr. Fu's guidance and advice has been invaluable in this endeavour, and this thesis would not have been possible without his support.

I would also like to thank the members of my family, including my wife Tiffany, parents Radu and Ngamta, siblings Joel, John-Luke, Heman Neginoth and Julia. I have always cherished the advice and help that my family has provided throughout my studies.

I am also grateful for the friendship, advice and support of my current and former colleagues at the iTSS lab, especially Kamal Hossain, who helped me become familiar with academic life and the requirements for successful research during both my Master's and PhD studies.

Academic studies are challenging, and life brings many uncertainties. I thank God every day for all the accomplishments He has helped me complete, both with my academic and non-academic life.

In every thing give thanks:
for this is the will of God in Christ Jesus concerning you.
— 1 Thesselonians 5:18

For my wife, Tiffany, who has always supported me throughout my studies. I love you.
For my children, Elizabeth and Iulian, who have brought joy as I move to the next phase of life.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Networks
ANOVA	Analysis of Variance
API	Application Programming Interface
BIC	Bayesian Information Criterion
CV	Connected Vehicles
CAV	Connected and Autonomous Vehicles
CCG	Canadian Capacity Guide
CFP	Cyclic Flow Profiles
CNN	Convolutional Neural Networks
COM	Component Object Model
CTM	Cell Transmission Model
DQN	Deep Q Networks
DNN	Deep Neural Networks
DRL	Deep Reinforcement Learning
FHWA	United States Federal Highway Administration
FIFO	First-in-First-Out
GTHA	Greater Toronto and Hamilton Area
HCM	Highway Capacity Manual
IP	Internet Protocol
I2V	Infrastructure to Vehicle
LWR	Lighthill, Whitham, and Richards
MAC	Media Access Control
MDP	Markov Decision Process
NN	Neural Networks
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
RHODES	Real Time Hierarchical Optimized Distributed Effective System
ReLU	Rectified Linear Unit
SCOOT	Split Cycle Offset Optimization Technique

SCATS	Sydney Coordinated Adaptive Traffic System
SUMO	Simulation of Urban Mobility
TMC	Turning Movement Counts
UTCS	Urban Traffic Control System
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure

Part I

INTRODUCTION AND LITERATURE REVIEW

This part is divided into two chapters. The first gives a broad overview of the key topics and literature related to this work while the second provides an in-depth literature review on previous research that has been done related to the topics of signal control, traffic modelling, big data, and machine learning.

1 INTRODUCTION

1.1 BACKGROUND

Traffic congestion is a major concern in many cities, and is estimated to cost nearly $4 billion in major Canadian urban centers (Transport Canada et al., 2006). A recent study done in the Greater Toronto Area for Metrolinx revealed that commuters in the Greater Toronto and Hamilton Area (GTHA) spend an average of 1,494 hours in traffic per 1000 vehicle kilometres travelled. When compared to free-flow travel speeds, during peak hours, commuters in the GTHA can expect to spend 63% longer on their commute (HDR Corporation, 2008). These congestion problems are the norm in many cities worldwide, and are consistently considered as one of the most significant problems facing urban centres as they grow. The negative effects of congestion extend beyond the inconvenience of additional commuting time, and include environmental damage and risks to health, lost productivity from reduced economic output, and higher vehicle operating costs. In the GTHA, these effects are estimated to cost the economy $2.7 billion each year.

When faced with the prospect of increasing travel demand, city planners and policy makers have two approaches that they can consider: expanding the capacity of the network, or increasing the network's efficiency. While the first approach is the simplest and most straightforward, it is often expensive, ineffective, and may not be possible in older, established cities. In contrast, the second approach aims to optimise the operation and effectiveness of existing infrastructure, which can often be a more cost-effective way of alleviating congestion. This research focuses specifically on one of the major tools for improving the efficiency of a traffic network, namely, traffic signal control.

1.2 TRAFFIC SIGNAL CONTROL THEORY

Traffic signals were developed in response to increasing demand for transportation, in turn, caused by growth in cities and auto ownership and the need to manage these systems to ensure safety and efficiency. The first control systems were developed in the late 1800's and were manually operated semaphores. Since then, traffic signals have come to play an integral role in the modern transportation system, and today's networks could not function without them. It is estimated that in the United States alone, over 270,000 traffic signals are in operation (Federal Highway Administration, 2017b). Large cities often manage thousands of traffic lights; for example, the City of Toronto had 2334 active traffic lights in 2017 (City of Toronto, 2017). The United States

Federal Highway Administration (FHWA) outlines the purpose of traffic signals as the following (Federal Highway Administration, 2017b):

- Provide for the orderly and efficient movement of traffic
- Maximise the traffic served at the intersection
- Reduce the frequency and severity of certain crashes
- Provide appropriate accessibility to pedestrians and minor streets

While seemingly straightforward, modern traffic signals have a number of parameters that govern their operations and performance. The important aspects of these systems are outlined in the following sections.

1.2.1 *Traditional Traffic Control Systems*

While technological advances have resulted in the development of a number of advanced signal control systems, adoption of advanced systems has been slow. As such, many municipalities and regions are still operating traditional signal control systems, which account for nearly 95% of traffic signals in the United States (Zhao and Tian, 2012). In these systems, signal controllers could operate either locally with signal timing plans being stored in the controller or in a network connected to centralised computers so that their timing plans can be set remotely. These timing plans can be described based on their *cycle length*, which is the total time required for a complete sequence of signal indications, and the *green time* allocated to each phase. In coordinated signals operating on the same corridor, the *offset*, which is the difference in time between the master clock and the start of the controller's first phase, is also specified. Figure 1.1 illustrates the signal timing of a corridor and the underlying relationship.

In many municipalities, signal timing plans are created based on field data collected at different times of the day. These plans are usually designed with the aid of software, such as TRANSYT or SYNCHRO, which solve for optimal timing plans based on a set of assumptions. For traditional methods, it is often assumed that traffic is unsaturated (i.e. does not exceed the intersection capacity), that traffic flows at a constant speed, and that traffic is in a steady state. This last assumption means that a repeatable set of signal timing plans can be created for a particular time of day (Lo, 1999). While special timing plans could be created for abnormal weather conditions or other situations where traffic behaviour is expected to change, the fundamental assumptions remain the same.

The most commonly employed method in the United States and Canada is the process outlined in the Highway Capacity Manual (HCM). The crux of this approach is based on determining a signal timing plan that ensures that the *capacity* provided to each movement is sufficient and meets all the minimum requirements for the intersection's safe operation (e.g. providing enough green time for pedestrians). For example, the City of Toronto expects that the volume-to-capacity ratio (v/c) for all approaches to be less than 1.0 (City of Toronto ITS Operations, 2016). This v/c ratio is also known

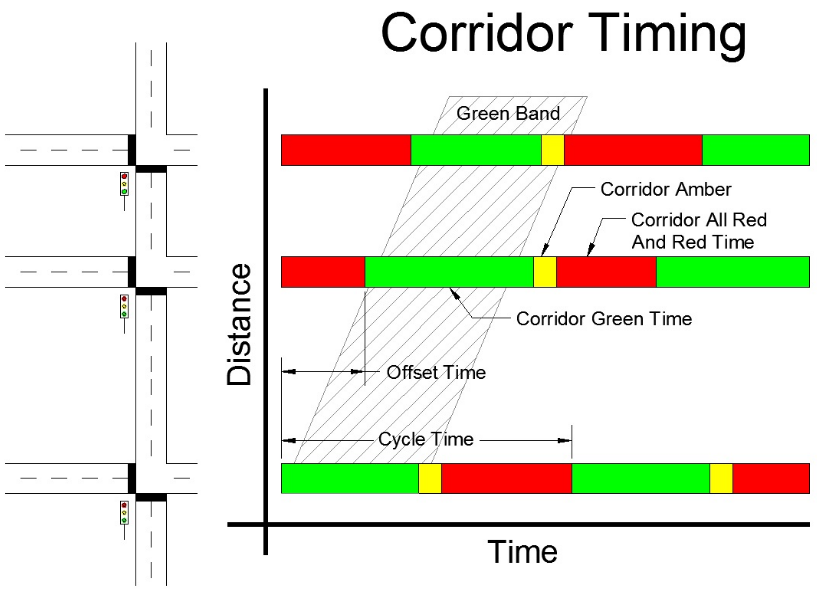


Figure 1.1: Signal Timing Diagram Highlighting Key Signal Timing Parameters

as the *degree of saturation* and obtained from the demand, green time for that movement, cycle length, and the saturation flow rate. Although the HCM contains a number of instructions and factors that can be used to calculate and estimate the saturation flow rate and effective green times for a variety situations, timing plans are often only developed using the demands and requirements of peak periods, and the timing plan is assumed to be adequate for non-peak periods. Under this method, coordinated corridors are timed to maximise the *bandwidth* of the corridor, which is the maximum number of vehicles that can pass from signal-to-signal without stopping if they travel at the corridor design speed (Kamarajugadda and Park, 2003).

These traditional systems require accurate field data before timing plans can be calculated. This creates a further limitation: if the field data are not updated the system will not be able to respond to changes in traffic. Studies have also shown that these timing plans are not generated in a consistent manner, and many agencies do not have defined policies concerning how signals are timed (Federal Highway Administration, 2017b). Various studies and surveys have shown that poor signal timing can be a significant contributor to traffic delay. In the United States, it is estimated that poor timing causes 10% of all delay on major roadways. One strategy commonly employed to address these shortcomings is the addition of vehicle detectors to traditional signal controllers, which are usually induction loops that detect when vehicles are waiting for a signal. These devices are commonly used on minor approaches, and can be used to trigger special movements (such as advanced left turns) and allow the signal controller to continue serving green to the major approach when no vehicles are waiting on the minor street or skip unneeded phases. These control strategies include "semi-actuated" configurations where only the minor movements have detectors, and "fully-actuated" where all approaches have detectors. On coordinated corridors, fully actuated controls can also be configured to extend the green time to support progression and allow vehicle platoons to pass easily (see Figure 1.1)

1.2.2 *Adaptive Signal Control*

While the actuated signal control systems decribed previously provide some responsiveness to changing travel conditions, these responses are made in a reactive way, limiting their ability to account for some predictable traffic patterns and optimise the performance of the traffic system as a whole. As such, they have limited applicability in unusual circumstances (such as a sudden increase in traffic due to a sporting event ending nearby). Furthermore, as these plans are created based on field data collected at a particular point in time, their efficiency decreases as traffic patterns change. The limitations of this approach were recognised in early studies, which led to the development of adaptive traffic control systems. The first adaptive system was proposed by Miller in 1963 (Miller, 1963; Friedrich, 2002). Since then, a number of adaptive systems have been developed, including Split Cycle Offset Optimization Technique (SCOOT), Sydney Coordinated Adaptive Traffic System (SCATS), Real Time Hierarchical Optimized Distributed Effective System (RHODES), and InSync (Federal Highway Administration, 2011). These systems differ in the methods used to adjust signal timings, and are described in greater detail in Section 2.1. For example, SCOOT, one of the first successful systems, developed in the 1980's by the Transportation Research Laboratory in the UK (Stevanovic, Kergaye, and Martin, 2009), adjusts the green ratios (splits) assigned to each phase incrementally over time.

Although these systems use different methods to adjust the signal timing, most of them rely on traditional technologies like loop detectors to determine current conditions. The SCOOT system, for example, typically requires detectors to be placed upstream of the traffic signal (see Figure 1.2)(SCOOT Systems, 2014). This design means that these systems are also heavily dependant on the predictability of traffic patterns downstream, which may not be possible in situations with highly variable traffic, such as cases with major disruptions occurring mid-block.

Due to these limitations and high implementation costs, these systems have failed to gain traction in cities, especially in small and medium-sized cities. A recent study has shown that only 30 system deployments have been made in the United States, due primarily to concerns about the system's performance, benefits, deployment and maintenance costs, and the requirement of additional personnel (Zhao and Tian, 2012). These systems have seen even lower rates of adoption outside of North America and Europe, where traffic is less predictable and the infrastructure requirements exceed the benefits they could provide.

The limited adoption of traditional adaptive signal control suggests that improvements are still required before such systems can be seen as viable replacements to traditional technologies. Adoption of these systems has been promoted by many different governmental bodies. For example, FHWA's *Every Day Counts* program was cre-

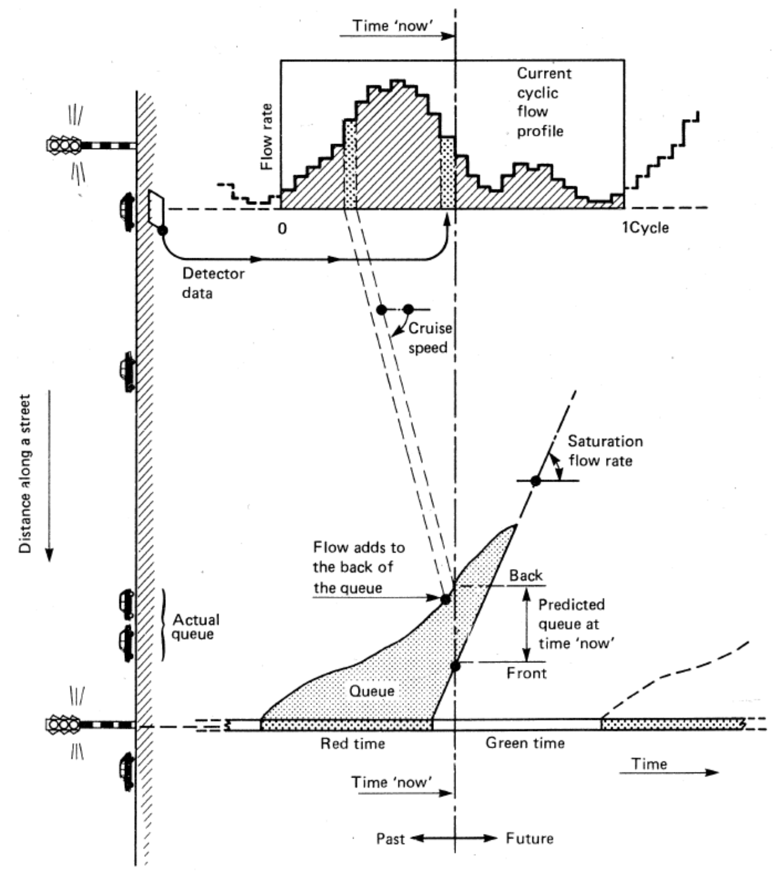


Figure 1.2: SCOOT Traffic Control System, Reproduced from Hunt, 1982 (Hunt et al., 1982)

ated to promote the use of innovative technologies in transportation, including the promotion of adaptive signal control in its first round (2011-2012). The results of these programs have already resulted in the development of newer and more advanced controls, such as InSync. Unlike traditional adaptive systems, InSync uses an Internet Protocol (IP) camera to collect data at intersections (but it can also use existing loop detectors). The camera's images are processed in real-time to generate estimates of demands, which are then used to prioritise which intersection approaches are served green. The system dynamically chooses from a library of user-defined sequences, but has the ability to change the order and length of phases as well as adjust the timings of other signals. While this facilitates customisation by deploying agencies, it limits its flexibility to adapt to new situations. This system has the advantage of being compatible with existing hardware, which has translated to many new deployments in recent years (Fontaine, Ma, and Hu, 2015; Rhythm Engineering, 2017). Municipalities have also reported positive experiences with the system, making it one of the most popular choices for adaptive control. As a proprietary system, agencies that have deployed the system have indicated that it is a *black box*, being overly dependant on the manufacturer for support (Centennial FDOT, 2016). Despite this shortcoming, the success InSync highlights the potential of such systems that take advantage of new technologies and available data.

Current methods in common use largely rely on limited data sources and employ heuristic and model-based approaches. While these approaches may deliver good performance in many situations, there is still abundant room for further improvements by considering alternative approaches, paradigms, and data sources. Machine learning presents an opportunity to capitalise on these changes and develop the next generation of traffic controllers. While some research and work has been done on such methods, research in this area is still very much in its infancy and many questions remain unsolved. In particular, existing research has focussed primarily on small problems and proofs-of-concept and few have looked at the complex cases involving a network of intersections with complex real-world road, traffic, and environmental variations.

1.3 BIG DATA AND MACHINE LEARNING

Since the development of the first adaptive traffic control systems (e.g. SCOOT), substantial technological changes have occurred. In particular, the emergence of the *Big Data* paradigm and advancements in artificial intelligence and machine learning have presented new opportunities for improved traffic control.

1.3.1 *Big Traffic Data*

One of the most prominent developments in recent years is an increase in the methods available to monitor traffic conditions. These methods use new data sources made available by the increasing presence of technology in every-day life. In popular media, these data sources are often called "Big Data". The most popular definition of this term has been described with *three V's*: Volume, Velocity and Variety (Laney, 2001). The ubiquity of portable devices, computers and the Internet has created an environment with a high volume of data infusing in real time (high velocity). For example, the rise of contact-less smart-card payment systems gives transit agencies a wealth of data when compared to traditional cash fares. With these systems, transit agencies can collect information on the boardings and alighting of each customer in real-time. A number of other sources also exist, and the following sections highlight some the ones most relevant to this research.

1.3.1.1 *Crowd-Sourced GPS Data*

In this context, it can be said that *Big Data* as a term describes a new paradigm that has manifested itself over the past decade. It has resulted in the development of new technologies that alter how society makes decisions. In transportation, perhaps one of the strongest examples of this is *Google Maps*, specifically *Google Traffic*. Developed in 2007, Google Traffic uses crowd-sourced data collected from people who use enable location services in their smartphone. Data are collected anonymously from such users in real time, including time-stamped position data and speed. These data are then analysed to estimate current roadway conditions and travel time (Barth, 2009). The technique used to arrive at these estimates is called *floating car data*, and is also used by other providers such as TomTom (TomTom, 2017).

Besides Google, a number of other companies also collect crowd-sourced data. One prominent example is INRIX, which uses a propriety system to aggregate traffic information from GPS-enabled sources, primarily vehicle fleets, such as delivery vans, taxicabs, trucks, mobile devices with road sensors, and other data sources (INRIX Traffic, 2017; Kim and Coifman, 2014; Kim et al., 2014). Through this data, INRIX provides information on historical and real-time speeds, travel times, and demand. These data are sold to private companies and municipalities alike, and have been used to replace traditional loop-detector based monitoring systems. One chief concern with these systems is their accuracy. While previous studies have found that accuracy levels are comparable to those provided by loop detectors (Haghani, Hamedi, and Sadabadi, 2009; Kim and Coifman, 2014), some studies have also demonstrated cases where these systems have failed to detect traffic jams (Kim et al., 2014) and have latencies associated with the real-time values they report, sometimes in excess of 10 minutes (Kim and Coifman, 2014).

1.3.1.2 *Bluetooth and WiFi Data*

In addition to GPS data, recent research has also focussed on using Bluetooth and WiFi data. The principles employed in these approaches is to capitalize on the fact that devices such as cellphones continuously broadcast a unique identifier as they search for networks. One of the first proposals to use this data to estimate travel times was made by Wasson et al (Wasson, Sturdevant, and Bullock, 2008), who proposed and tested a system to estimate travel times on a freeway. These devices work by tracking connected devices as they pass by various tracking devices in the network. Since their development, these devices have found applications in many cities and regions worldwide as a roadway performance indicator. For example, the City of Calgary currently uses Bluetooth detectors to provide real-time travel time estimations to commuters through its variable message signs and online through its website (City of Calgary, 2017). Despite the some use of these devices in such situations, their use is currently restricted to condition monitoring and performance measurements.

Besides travel time estimates, Bluetooth and WiFi detection has found applications in monitoring pedestrian and individual behaviour. These applications are commonly proposed for use in the retail sector, where the technology can be used to track shopper behaviour (e.g. how long they stay in the store, how often they return, etc. (Phua, Page, and Bogomolova, 2015; Yoshimura, Krebs, and Ratti, 2016)). Some of these approaches are notable in that their analysis is based on single devices. For example, Malinovskiy et al. (Malinovskiy, Saunier, and Wang, 2012) evaluate the use of Bluetooth detectors to study pedestrian behaviour. Besides using these devices to estimate volume, the study is notable in that the collected data was used to estimate dwell time around the site. This was estimated by tracking devices that were continuously present, defined as successive detections within 60 seconds. Their study used a high powered detector with an estimated range of 50 metres mounted on a pole, and evaluated travel behaviour on a busy section of a university campus. Their results indicated that in situations with low volume, some of the metrics were difficult to estimate, however the technology is suitable to obtain estimations of these quantities.

A significant limitation of this technology in traffic situations is its inability to distinguish signals from vehicles and those from other agents, such as pedestrians or cyclists. The accuracy of the estimates provided by this technology may also limit its applicability to certain scenarios. For example, a recent study by Moghaddam and Hellinga evaluated the accuracy of Bluetooth detectors in detecting travel times on a 1.5km arterial through a simulation approach. They found that in all the scenarios tested, the 95% confidence interval of the measurement error exceeded the 5% of the true mean travel time. Additionally, in situations with poor coordination, long cycle lengths, or high volume-to-capacity ratios this error approached 25% of the true mean (Moghaddam and Hellinga, 2013). Furthermore, when using these measurements for real-time travel time detection, a time lag exists in the data as vehicle times cannot be measured until they pass a downstream detector (Moghaddam and Hellinga, 2014).

1.3.1.3 *Connected Vehicle Technologies*

In addition to increased data availability, the relationship between infrastructure and vehicles themselves is changing. In particular, substantial research in the area of autonomous vehicles and connected vehicles is being done. These technologies can be divided into four categories (Federal Highway Administration, 2017a):

1. *Connected Drivers*, which are drivers who use technology to communicate and alert each other of traffic incidents and alternate routes. Smart-phone applications that use crowd-sourced data such as Waze, Google Traffic or INRIX are examples of these.

2. *Connected Vehicles* are vehicles that are able to communicate with each other or with infrastructure through embedded technology. These vehicles could broadcast information about their speed, location, and other actions the driver may be taking (such as braking).

3. *Automated Vehicles*, where computers manage driving tasks in place of a human driver. Basic implementations of this technology are already present in many modern vehicles, including adaptive cruise control which allows the vehicle to automatically adjust its speed based on its surroundings without human input.

4. *Autonomous Vehicles*, which are vehicles that are able to operate without any human assistance.

To support the development of these technologies, regulatory agencies are developing guidelines to mandate the inclusion of Vehicle to Infrastructure (V2I) and Vehicle to Vehicle (V2V) devices on new vehicles. Research is still needed to determine the best methods by which V2I technologies can be incorporated to improve traffic signal control.

1.3.2 *Machine Learning*

In addition to the emergence of a data-rich environment, substantial strides have also been made in the areas of artificial intelligence and machine learning. These techniques aim to replicate the decision-making capabilities of human beings, and many unsolved challenges remain. In particular, the development of Deep learning has shown exceptional promise, and has already been implemented in many high-profile applications, such as the AlphaGo algorithm that was the first computer program to defeat the best human players in the game of Go. AlphaGo uses a tree search algorithm to find the best move based on training obtained through its deep learning model (Google Deep Mind, 2017). Deep learning models are a special type of Neural Networks (NN) model that are larger (*deeper*) than traditional implementations. These models use a collection of simple interconnected elements called *neurons*. The structure is modelled after an abstraction of the human brain, where neurons activate and connect to each other depending on the action a person is doing. Likewise, in response to specific inputs, various neurons in the NN will activate. Neurons in NN are usually organised into layers, and neurons in successive layers are connected to those of previous layers and will activate based on a weighted relationship to them. By comparing the final output of the NN, given a specific stimuli, to the desired output, adjustments are made until the model consistently produces the expected results. This process of finding weights that produce desired behaviour is called *learning* or *credit assignment* (Schmidhuber, 2015). A fundamental limitation of NN models is that the training difficulty increases substantially as the network grows. *Deep Learning* therefore refers to the collection of techniques developed to deal with this problem.

Traditional NN learn by using a technique called *backpropagation*, which is based on determining the difference between the final output of the model and the *correct* output, and then propagating them backwards through the model. A chief limitation of this approach is that in larger networks, it becomes difficult to update the weights as the signal propagating backwards becomes weaker. This effect is often called *vanishing gradient*, which describes the phenomena where the contribution of each weight to the error becomes harder to determine as the model size increases. In addition, local minima values in the error function can lead to the identification of incorrect values. To address these issues, researchers have developed a number of techniques to improve the training process which has enabled deeper networks such as those used by AlphaGo to function. The first approaches, proposed by Hinton et al. (Hinton, Osindero, and Teh, 2006), were *Deep Belief Networks* that used dense networks trained one layer at a time in a bottom-up approach by constraining the weights of layers above. After the layer-by-layer training is complete, backpropagation is used to fine-tune the weights. Hinton et al. applied this scheme to demonstrate that their model had high accuracy on a version of the MNIST handwritten digit recognition problem commonly used to benchmark models (Hinton and Salakhutdinov, 2006). Other methods developed to address these problems include the use of Convolutional Neural Networks (CNN) to reduce the complexity of the data. In CNN, a smaller *filter* or *convolutional* layer containing a few weights is shifted step-by-step across the input data. The output from this filter layer is then used as input in subsequent layers. The advantage of this structure is a reduction in the total number of weights required.

In contrast to simpler models, the additional complexity of deep networks allow them to produce better and more accurate results and find patterns in noisier data. This feature makes them an attractive choice in the world of Big Data. One of the most prominent areas where they have been applied is that of image classification. For example, Google Photos allows users to search for a term and will find all photos in a user's library related to that term. The system employs an unsupervised feature learning technique called *DistBelieve*, which uses large-scale clusters of machines to distribute training and inference in a deep learning method (Dean et al., 2012; Rosenberg, 2013). While these techniques have seen extensive application in these types of situations, real-world applications of machine learning in traffic signal control has yet to be done, and significant challenges still remain un-addressed. These challenges and the work currently done in this area are explored in greater detail in Section 2.4.

1.4 PROBLEM STATEMENT

While the problem of optimal traffic signal control has been explored extensively in the past, much of the past work has focussed on traditional sources of data and optimization methods. To date, few traffic signal control systems have been developed that capitalise on the availability of crowd-sourced big data and the power of deep learning based optimization strategies. Specifically, as more high-granular traffic data becomes available, such as travel time from Bluetooth and WiFi detectors, queue length from traffic cameras, and trajectory and route information from connected and autonomous vehicles, so is the significant opportunity to develop new control strategies that address some of the shortcomings of the traditional adaptive control systems.. However, despite the rapidly growing popularity of Big Data and machine learning, research on their applications in traffic signal control is largely scarce with few significant progresses reported in literature. Many open questions are still remaining:

- How should various new data sources such as Bluetooth and WiFi probe data be formally characterized and what is their relation to traffic signal operations?
- How should the traffic signal control problem be formulated so that it can be solved in a most effective way using a deep reinforcement learning algorithm? What variables should be used for defining system states and what reward or penalty functions should be adopted in accordance to technological settings and data availability? Few methods using these data sources to create an adaptive signal timing system have been proposed in the literature. Furthermore, existing methods in the literature have assumptions that will only be valid in the distant

future or are too narrowly defined. For example, Feng et al. (Feng et al., 2015) evaluate a system that uses data from connected vehicles only, and do not consider common elements of signal operations. A need exists for a system that is able to capture and use data from multiple sources.

- How effective is a deep learning based signal control system as compared to the traditional traffic signal optimization methods? What is the capacity for a deep reinforcement learning based signal control model for addressing some typical traffic variation patterns in time (e.g., time of day and day of week variation), over space (e.g., directional, by movement, and network-wide), and by environment (e.g. accidents, road closures, and weather)?
- What are the limitations of and requirements for the various data sources to be effectively applied for signal optimization using machine learning? Specific issues could be related to the minimum market penetration rate, measurement errors of a particular sensor technology, and spatial and temporal coverage, under various traffic situations (e.g. uncongested, congested, or during a sudden increase of traffic).

1.5 RESEARCH OBJECTIVES

This research seeks to investigate the application of machine learning based strategies for traffic signal control with a specific focus on their ability to take advantage of new traffic data from various emerging technologies such as video cameras, Bluetooth/wifi detectors, and connected vehicles. The following specific objectives are expected to be achieved:

- To examine different ways in which various data sources can be used and to propose an approach to utilise these data in a real-time or near-real-time application.
- To develop deep learning based models and other adaptive models for optimising traffic signal operations of varying complexity, from single intersections to a network of multiple intersections along a corridor. The proposed models will assume a combination of multiple traffic data sources, including Bluetooth and Wifi detector data, video data, or data from connected vehicles.
- To investigate the performance of alternative deep learning models as compared to various traditional traffic signal control methods, including popular adaptive approaches, and to assess their relative performance with regard to state representation, reward functions, and training settings.
- To examine how deep learning models may be trained and applied to field settings, including questions of transferability, generality, and requirements.

1.6 THESIS ORGANISATION

This thesis is organised into three parts. The first part contains this introductory chapter and a second chapter that provides a detailed overview on relevant literature. The second part outlines the methodology and results of three major studies, each being covered in one chapter. The first study details an approach to use Bluetooth and WiFi probe data for signal optimisation while the second is a sensitivity analysis on a DRL model. The third study covers an application of the DRL model to a corridor and a comparison against adaptive control. Finally, the last part provides the conclusion, including the major findings, contributions, and limitations of the work.

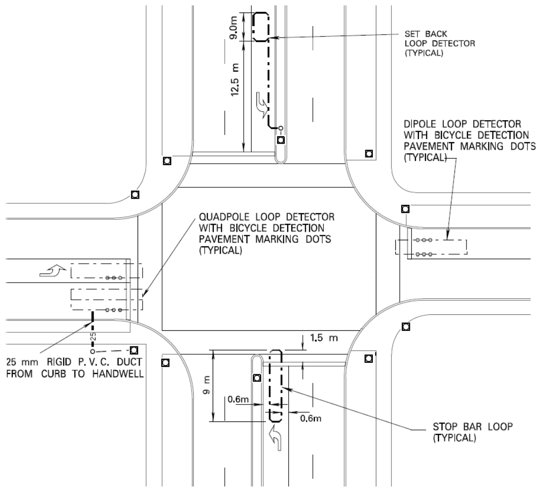
2 LITERATURE REVIEW

Modern traffic signals are the product of many years of research and development. This chapter briefly traces the development of different traffic signal control methods and strategies, starting with traditional ones commonly in use today. Substantial research has been done in this area, but as a continually evolving field there are always new developments. This chapter therefore also highlights research done on the use of more advanced control strategies, such as adaptive signal control, as well as those capitalising on emerging technologies.

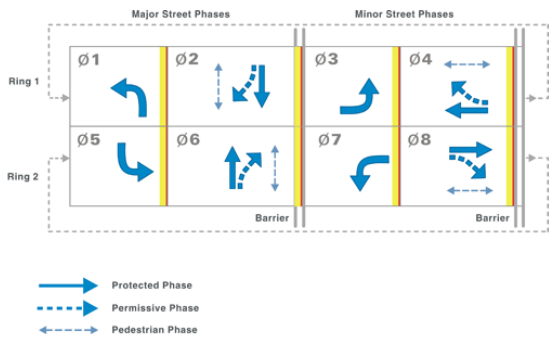
2.1 THE DEVELOPMENT OF MODERN SIGNAL CONTROLLERS

The development of traditional traffic signals can be traced back to railways. In the late 1800's, mechanical semaphore signals were created to control rail traffic using oil lamps placed behind coloured discs. In 1866, similar devices were adapted to create the first mechanical traffic signal to guide roadway traffic at a busy intersection near London's parliament buildings. In the period following this, the growth of automobile traffic led to the development of a number of similar devices across Europe and North America. In the 1920's, traffic signals with the design commonly seen today first appeared in Detroit and New York (Mueller, 1970). Since then, developments and improvements continued to be made, and by the 1930's the need to improve the responsiveness of traffic control to different situations was identified. Without vehicle detection, signals were timed with fixed schedules and intervals, or were manually controlled. This approach, however, is ineffective as green may be served to directions that have little or no demand. The earliest methods used to address these limitations were time-of-day based scheduling, including variable split times and cycle lengths. However, these methods still did not address the issues of green being shown to minor streets even when no vehicles were waiting (Tyack, 1938). To address these needs, vehicle detectors were developed for busy intersections, with early devices making use of microphones and pneumatic mats to detect vehicles and call the signal to serve a direction (Tyack, 1938). The advantages of signal progression in signal timing were recognised early, and actuated systems were designed such that the main road would be served until vehicle platoons were fully served (Tyack, 1938). By the late 1970's and early 1980's, induction loop detectors were developed for roadway use (e.g. (Koerner, 1976), and quickly became one of the most common vehicle detection methods in use. Today's typical intersections in North America and Europe use induction detectors often placed directly at the stop bar. Vehicle detectors may also be located upstream of the intersection to enable detection of vehicle platooning, to facilitate green time extension, or to address dilemma zones caused by the signal changing to amber.

In addition to advancements in vehicle detection and signal control technology, the methods employed in traffic control have also changed since the first traffic signals, and the current generation of traffic controllers are capable of employing a number of strategies to manage traffic demand. Modern practice mostly divides traffic control into continuous loops (called *rings*) which serve signal *phases* in sequence. Phases are signal aspects that indicate which movements (e.g. through, left, right) at the intersection are to be served. Signal plans are typically described using *ring barrier diagrams*, the sequence of phases that time in each ring. Phases that do not conflict and can time concurrently are placed in separate rings, while those that do conflict are placed in sequence. The diagram can then be read as a kind of film-strip progressing from left to right and looping back at to the start. Barriers indicate reference points in the cycle that must be served at the same time for all phases, and are used to define the relationships between compatible movements (Federal Highway Administration, 2017b). An example ring-and-barrier diagram and typical intersection layout is shown in figure 2.1.


9.0m
12.5 m
SET BACK LOOP DETECTOR (TYPICAL)
DIPOLE LOOP DETECTOR WITH BICYCLE DETECTION PAVEMENT MARKING DOTS (TYPICAL)
QUADPOLE LOOP DETECTOR WITH BICYCLE DETECTION PAVEMENT MARKING DOTS (TYPICAL)
1.5 m
25 mm RIGID P.V.C. DUCT FROM CURB TO HANDWELL
9 m
0.6m
0.6m
STOP BAR LOOP (TYPICAL)

(a) Sample Intersection Layout (City of Toronto, 2012)



(b) Sample Ring-and-Barrier Diagram (Federal Highway Administration, 2017b)

Figure 2.1: Sample Intersection Layout (City of Toronto) and Ring-and-Barrier Diagram

Modern signal controllers also have special considerations for left-turn phasing, which must cross a conflicting traffic movement. These include permissive left turn phasing, which requires drivers to yield to the oncoming movement, and protected phasing, which gives left turns a separate phase for exclusive movement. In many configurations, traffic signals are configured to provide a combination of protected left turns at the start of the phase, followed by permissive left turns during the through movement's green phase. Many signal controllers also serve these movements on-demand, using loop detectors to call the protected left turn phase when needed. These detectors may also be set back behind the stop bar, functioning as a form of queue detector that requests a protected phase only if a certain number of vehicles are waiting (see figure 2.1).

Vehicle detection at traditional intersections is most often facilitated through the use of loop-detectors. Traditional detectors use induction coils to detect metal objects passing over them, and can operate in two modes depending on their purpose. In *pulse mode*, the detector sends a short pulse after detecting the arrival of a vehicle. The pulse is usually timed to last 0.1 to 0.15 seconds regardless of the length of the vehicle. These detectors are usually used in traffic counters or in upstream detectors, which are used to extend the green phase and mitigate dillemma zone issues. In *presence mode* the detector is continuously activated as long as a vehicle is present and waiting over it. These detectors are usually used to detect if vehicles are waiting at a traffic light and call the appropriate signal. They are often placed at the stop-bar (Federal Highway Administration, 2017b).

Detectors at intersections are usually laid out in two configurations: semi-actuated control and fully-actuated control. In semi-actuated control, detectors are placed on the minor street approaches only. The detectors are used to request calls to the signal controller for green, which serves these movements only when needed. The controller attempts to sustain green for the maximum time possible on the main street. In contrast, fully actuated signals have detectors on all approaches, and serves movements as they are requested. The requirement for detectors on all approaches makes this configuration more expensive than the semi-actuated approach.

Over the years, a number of rules have been developed for signal controllers to follow when serving phases. When a signal phase is served, the signal must show green for a period long enough to allow all waiting vehicles to clear the signal and to give pedestrians sufficient time to cross. Modern controllers can also be configured to permit pedestrian movements only if requested by pedestrians pushing a button. Additionally, modern actuated controllers have the ability to extend or shorten the green time served to a particular approach, within the confines of pre-specified maximum and minimum greens. To accomplish this, *passage time* is normally used to find gaps in traffic to terminate phases. With this configuration, the time between successive vehicle actuations on the movement being served green is recorded, and if it exceeds a threshold value, the signal changes (i.e. *gaps out*) to red to serve the next phase. The configuration of these threshold values is important to the efficient operation of the signal, and substantial research has been done in this area. The values are ideally set such that the system serves the entire queue, but the values are dependant on the design of the detector (i.e. its length) (Gordon et al., 2010).

Efficient operation of actuated signal control depends heavily on regular evaluations of the intersection's performance and adjustments in signal timing as needed. Previous studies have shown that signal timings should be reviewed every 30 to 36 months, but many regions do not meet this standard due to budget constraints (Gordon et al., 2010). Regular re-timing has been shown to have significant benefits. For example a project in Texas demonstrated reductions of 9.1% in fuel consumption and 24% in delay after signals were re-timed (US Department of Transportation, 2007). A cost benefit analysis of signal re-timing done on New Jersey Route 23 examined the benefits of re-timing the signals and found that the cost benefit ratio is approximately 24 to 1. The study also provided some recommended re-timing intervals based on the analysis for different traffic growth scenarios. A recommendation of 3 years was given for 0.5% annual growth in traffic, and 1 year for 1% and 1.5% growth in traffic (Chien, Kim, and Daniel, 2006). Many other studies have shown significant benefits to regular re-timing in terms of reduced emissions (fuel consumption) and reduced delays. Despite these benefits, many agencies fail to regularly re-time their signals due to the significant costs of re-timing, which is generally estimated at between $2500 to $3100 (Federal Highway Administration, 2017b; Gordon et al., 2010). The total cost can quickly add up for cities with large networks. Additionally, intersections in rapidly changing neighbourhoods may require more frequent re-timing.

2.1.1 *Adaptive Signal Control*

To address the limitations of traditional signal control, a number of adaptive traffic control systems have been developed. The most popular systems include SCOOT, SCATS, RHODES and InSync. Of these, the first three rely solely on traditional loop detector technology. The following sections provide an overview of each system.

2.1.1.1 *SCOOT*

SCOOT is a centralised traffic control system developed in the 1980's by the Transport Research Laboratory in the UK (Stevanovic, Kergaye, and Martin, 2009). SCOOT is designed to work with Urban Traffic Control System (UTCS) controllers, which is the most common traffic control system deployed worldwide (Chiu and Chand, 1993; Robertson and Bretherton, 1991). The primary control objective of SCOOT is the minimization of the sum of the average queues in the area. In the ideal scenario, there are no queues and all vehicles see green when they reach the signal. To support this objective, SCOOT uses a traffic model to estimate the queue size based on its Cyclic Flow Profiles (CFP). The CFP is obtained from a point upstream of the signal to be optimised and represents the flow of vehicles passing that point over a predefined time increment (see figure 2.2). Typically, this time increment is set for 4s, and periods where platoons of vehicles pass-by can be visualised as periods with successive high flow.

The CFP is generated in real-time for all approaches by using flow data from detectors placed upstream of the stop-line; as a result, the method is dependant on the installation of additional detectors, often beyond those typically used for actuated control (Hunt et al., 1982; Chiu and Chand, 1993; Robertson and Bretherton, 1991; SCOOT Systems, 2014). Following the calculation of the CFP, the algorithm estimates when the vehicles will arrive at the downstream signals. Using deterministic equations, the size of the queue and the time it takes to clear can be calculated, and signal timing changes can be predicted. The method requires the assumption that platoons travel at a known cruising speed, and that discharging queues reach the saturation flow rate at the intersection (Robertson and Bretherton, 1991; SCOOT Systems, 2014; Hunt et al., 1982).

Optimisation in SCOOT is done incrementally, which results in a more predictable and non-erratic response. As per its name, this incremental optimisation is done by adjusting the split, cycle, and offset times. Each of these is optimised in a different way by a separate part of the algorithm. The split optimiser is run a few seconds before a phase change is set to occur, deciding whether the current split should be extended or shorted by 4 seconds, or if it should remain unaltered. The cycle optimiser and offset optimiser also make the same decision for their respective parameters. The cycle length optimisation is reactive in nature, and is usually configured to keep the busiest intersection at a predefined saturation level (usually 90%). If the saturation level increases beyond this value, the cycle length is increased, and conversely if it decreases below this level, the cycle length is decreased. The result is illustrated in Figure 2.2 where the decisions shift, extend, and shrink the phase (Robertson and Bretherton, 1991; SCOOT Systems, 2014; Hunt et al., 1982).

2.1.1.2 *SCATS*

Development of the SCATS system first began in the 1980's in Australia, and remains popular in its namesake city, Sydney. Since its development, it has seen applications in many major cities worldwide. Unlike SCOOT, SCATS is a hierarchical platform that organises intersections into groups (called subsystems) which guide the actions it takes. Like SCOOT, optimisation is done on the split, cycle, and offset times (SCATS, 2000; Stevanovic, Kergaye, and Martin, 2009; Wilson, Millar, and Tudge, 2006; Sims and Dobinson, 1980). Three types of controllers exist in the system, each defined by their responsibilities. At the lowest level, each intersection is controlled by an individual computer which is responsible for processing data collected from its intersection and to make tactical decisions on the signal's operation. At the next level, a regional controller is responsible for the real-time operation of up to 200 sets of signals (Sims and Dobinson, 1980). Within a region, individual intersections are grouped into *subsystems* that consist of a critical intersection whose timings can be adjusted by the regional controller directly (SCATS, 2000; Chiu and Chand, 1993). Subsystems do not need to contain multiple intersections, they can be defined from only a single intersection. Intersections in the subsystem are always coordinated together, including sharing a common cycle length and possessing inter-related phase splits and offsets.

Their phase splits are allocated such that they are compatible with those at the critical intersection (based on calculations and assumptions about the flow of traffic). Subsystems may also be temporarily linked to other subsystems, depending on prevailing conditions. When linked these systems will all share a common cycle length (SCATS, 2000; Chiu and Chand, 1993). Finally, at the highest level, a supervisory computer links all the regional controllers together and provides information on the system's current status (Sims and Dobinson, 1980)

Unlike SCOOT, optimisation in SCATS is conducted on the basis of *Degree of Saturation*, which is the ratio of the effectively used green time to the total available green time. In a typical installation, SCATS requires loop detectors located in each lane at the stop line to facilitate the estimation of the degree of saturation. These detectors are required at the critical intersections, but are optional at non-critical intersections in the subsystem. Within a subsystem, the cycle length is optimised such that busiest lane of the critical intersection maintains a degree of saturation of 0.9 (90%). To support coordination, large cycle lengths are sometimes used, as smaller cycle lengths do not always support progression. Offsets are selected to minimise stops in the direction with greater flow. Individual phases are adjusted with the goal of equalising the degree of saturation across all competing approaches.

2.1.1.3 *RHODES*

Like SCATS, RHODES approaches the problem of signal control as a hierarchical problem with decisions made at different levels. At a fundamental level, RHODES is modelled as a *dynamic network loading* model that captures the slowly varying nature of traffic. The system works by estimating the load (or demand) on constituent links in the network in vehicles per hour. At the local level, these demand estimates are then used to allocate green time to competing movements.

At the middle level, the system also makes decisions to support traffic flow through the network. Here, specific travel flow characteristics are estimated, such as the speeds of platoons of vehicles and their expected arrivals and the queues they will create. These aspects culminate in the final selection of phase change times at the intersection level, based on the predicted times that vehicles arrive at the intersection. In contrast to SCATS or SCOOT, the decision logic of RHODES is more complicated, and requires upstream detectors placed just after the previous traffic light for each intersection. Based on detections at the upstream detector, the system predicts when and at what rate vehicles will arrive at the subsequent downstream detector. Because of its design, the system's prediction model is dependant on estimations of the link free-flow speed, queue discharge rates, and turning probabilities. Based on the accuracy of its predictions, the system is able to adjust its estimation of these parameters, though initial parameters are determined based on default functions. The result of this model is the ability to predict queues at all approaches. Unlike traditional implementations, the RHODES model was designed to function without pre-specified timing plans and phase sequences (though these can be provided), and instead it responds pro-actively by setting phases and phase durations based on predicted traffic conditions (Mirchandani and Head, 2001). Despite its ambitious design, the system has not seen widespread deployment on the same scale as SCOOT and SCATS.

2.1.1.4 *InSync*

InSync is a proprietary adaptive control system developed by Rhythm Industries, and has seen strong popularity in the United States in recent years. As of 2017, the company reported deployments in over 30 states and 1 Canadian province. While systems like SCATS and SCOOT can be deployed with other forms of vehicle detection outside of traditional loop detectors, existing implementations have largely been confined to traditional technologies. In contrast, InSync is designed to use IP cameras positioned at intersections. Video detection can also be augmented with existing loop detectors. The cameras process real-time images of the traffic, and proprietary software is used to process the images and extract information on demand levels. These demand levels are in turn used to choose which movements will be served green and for how long. The system is customizable to some extent, and allows agencies to choose preferences surrounding phasing and timing (Fontaine, Ma, and Hu, 2015). Despite its popularity, the system is sometimes criticised due to its *black-box* nature, and agencies using the systems must go liaise with system's manufacturer when changes are necessary to the system (Centennial FDOT, 2016). Another limitation the system is its reliance on high-quality communications, which however come with the benefit of being able to view camera feeds at all controlled intersections (Fontaine, Ma, and Hu, 2015; Centennial FDOT, 2016)

2.1.1.5 *Evaluations of Adaptive Traffic Control Systems*

The performance of SCATS and SCOOT on networks have been studied extensively. For example Kergaye et al. conducted an extensive set of simulation-based tests comparing SCATS, SCOOT and traditional actuated signals (Kergaye, Stevanovic, and Martin, 2008). Their research demonstrated that both SCATS and SCOOT reduced delays, travel time, and total stops by at least 10%. When compared to each other SCATS and SCOOT performed similarly across a high range of test scenarios, and no substantial differences were observed in their effectiveness.

Field testing of a RHODES system was done at an intersection of highway US60 and a rural road (Mirchandani and Lucas, 2001) and compared to a traditional actuated controller. The system tested was modified to accommodate placement at a parclo-interchange with two traffic lights. The results of the field tests showed that the RHODES system performed as well as the actuated controller. The field tests were limited by restrictions on the RHODES system, which did not allow phase skipping or adjustment of maximum and minimum green times. Despite this, the system demonstrated the ability to provide signal phasing equal to that of an optimised actuated controller "on-the-fly".

A field evaluation was conducted of the SCATS system on a deployment in Park City, Utah. The deployment consisted of 14 signals, and evaluation was conducted by collecting data while the SCATS system was running and comparing it to previously deployed actuated signals. The results showed that SCATS consistently reduced travel times and delays, and generally lowered the number of stops and stopped delay for both major and minor movements in the system (Stevanovic, Kergaye, and Martin, 2008). Similarly, a field evaluation of the SCOOT system was conducted in Anaheim. The results of this study highlighted the limitations of these systems, and the expected benefits were not realised when compared to the previously deployed adaptive control system (Moore et al., 2005). The results of the study highlighted the need for proper training and configuration of these systems, as institutional challenges were reported in the study that limited the expected performance of the system. The evaluation was done as part of a large project that received funding from the US Department of Transportation, but unanticipated costs caused a breakdown in the project management used to deliver the SCOOT contract.

To promote its product, the manufacturer of InSync has conducted a number of case studies evaluating the effectiveness of its system. An independent study conducted by the Virginia Centre for Transportation Innovation and Research to evaluate the effectiveness of the system across a wide range of deployments (Fontaine, Ma, and Hu, 2015). The results of the study showed that the system generally improved the performance of the main street if it was not operating over capacity, did not have unusual geometric designs, and was not already operating at a good level of service. In oversaturated conditions, no benefits were observed as green time cannot be effectively allocated. The study also reported that high volume unsignalised accesses, high truck volumes, and sparsely separated signals all reduce the benefits of the system when compared to traditional control methods. Of note is that the study found that side street delays were generally increased after deployment of these systems, but were offset by delay decreases on the main streets. This study was extensive, using data from floating cars, Bluetooth detectors, and INRIX GPS. The study highlighted that the largest reductions in travel time were observed during the midday and afternoon periods on weekdays, suggesting that the system is most effective when demand is highly variable. An average reduction of 4.6% was observed in the 95th percentile travel time across all testing periods and sites after the system was deployed.

2.2 TRAFFIC MODELLING AND PREDICTION FOR SIGNAL CONTROL

A critical component of modern signal design processes is traffic modelling. The Highway Capacity Manual (HCM)'s method is commonly applied when re-timing signals and included in software packages such as Synchro (HCM2000 and HCM2010). Its basis includes analytical and experimentally generated models. Beyond the HCM, many models have been formulated to replicate different aspects of the traffic system, depending on the level of detail required. This section highlights some critical modelling approaches commonly used by researchers: queueing models, dispersion models, cell transmission models, and micro-simulation approaches. Simulation itself is a collection of individual models that replicate specific aspects of driver behaviour, such as car-following and gap acceptance.

2.2.1 *Queueing Models*

Queueing is one of the most elementary processes that influence the performance of an intersection. Queueing processes, however, exist in many other applications such as computing, telecommunication, and industrial settings. Many general purpose mathematical models have been developed to predict queue behaviour based on concepts such as the arrival pattern, departure pattern, and number of servers. In traffic settings, all queues are fundamentally First-in-First-Out (FIFO) queues. In terms of arrival and departure patterns, queueing models can be deterministic or stochastic. Typical notation uses an abbreviation for the arrival, departure, and server count, such as D/D/1 which refers to deterministic arrivals, deterministic departures, and one server. The following section gives a basic overview of the differences between deterministic approaches and stochastic approaches.

2.2.1.1 *Deterministic Queueing*

In deterministic queueing systems vehicle arrivals and departures are assumed to be known exactly. The system states and inputs can be specified by four key functions, including the arrival rate $\lambda(t)$, departure rate $\mu(t)$, cumulative arrivals $A(t)$, and cumulative departures $D(t)$. The cumulative arrivals and departures represent the number of vehicles that have entered or left the queueing system since the start of the model ($t = 0$) and so the following relationships are also true:

$$\int_0^t \lambda(t) = A(t) \tag{2.1}$$

$$\int_0^t \mu(t) = D(t) \tag{2.2}$$

$$Q(t) = Q(0) + A(t) - D(t) \tag{2.3}$$

where:

t is the time examined (usually in seconds)

$\lambda(t)$ is a function describing the arrival rate (usually in vehicles per second)

$Q(t)$ is a function describing the number of vehicles queued (entered but have not yet departed) in the system at time t

$A(t)$ is a function describing the total number of vehicles that have arrived by time t (in vehicles)

$D(t)$ is a function describing the total number of vehicles that have departed by time t (in vehicles)

If vehicles are assumed to instantaneously stop and accelerate when entering or leaving the queue, then the total delay T_d for vehicles can also be calculated as follows:

$$\int_0^N A^{-1}(n) - \int_0^N D^{-1}(n) = T_d \tag{2.4}$$

where:

T_d is the total delay (in vehicle-seconds)

N is total number of vehicles that have arrived during the analysis period

$A^{-1}(n)$ is the inverse of the arrival function $A(t)$ that provides the time it takes for n vehicles to arrive (usually in seconds)

$D^{-1}(n)$ is the inverse of the arrival function $D(t)$ that provides the time it takes for n vehicles to depart (usually in seconds)

The maximum queue is the largest difference between $A(t)$ and $D(t)$, which occurs just when the light changes. The maximum queue reach occurs at the moment the queue dissipates (when $A(t)$ and $D(t)$ meet). The degree of saturation (or volume-to-capacity ratio) of an intersection is called X in the HCM and can easily be observed when looking at queueing diagrams. In under-saturated conditions, the $Q(t)$ will always be 0 before the end of the green period. If $Q(t)$ becomes 0 as the green ends, the intersection is saturated, and if $Q(t)$ is not 0 before the end of the green period then the intersection is over-saturated. In over-saturated conditions, a residual queue will be present at the start of the next cycle, and if over-saturated conditions persist this residual queue will continually grow cycle after cycle.

2.2.1.2 *Stochastic Queueing*

While deterministic systems can be useful in understanding some patterns and trends, there may be uncertainty in both the arrival pattern and departure pattern (Tong et al., 2015). Stochastic processes can be modelled using the same principles discussed previously, but by replacing the $\lambda(t)$ and $\mu(t)$ functions with random functions. The key processes modelled in stochastic approaches are the arrival and departure headways. In some situations only arrival rates are modelled stochastically while a deterministic approach is used for the departures (Tong et al., 2015). Distributions for arrival and departure headways are normally obtained empirically. For example, some studies have shown that departure headways follow log-normal distributions (Jin et al., 2009) and arrivals follow Poisson processes (Adams, 1937).

Stochastic models can also be used to derive estimations of average waiting time and average queue length through steady-state analysis. Various methods have been studied by researchers in both transportation and other disciplines. Steady-state analyses can use both numerical methods and approximations to characterise the queueing system (Van Houdt and Blondia, 2005) (Bekker et al., 2011).

2.2.2 *Platoon Dispersion*

When a platoon of vehicles travel depart from a signalised intersection, there is a tendency for them to separate and spread out as some vehicles move faster than others (Shen et al., 2018). This is commonly referred to as platoon dispersion. The Robertson model, or variants of it, is one of the most common and simplest models employed to model platoon dispersion dynamics. This model was first included in TRANSYT and was developed in the late 1960s (Robertson, 1969). Fundamentally, the Robertson model is a geometric model that applies smoothing factors to upstream volumes and takes the following form:

$$q'(n+t) = F \times q(n) + (1-F) \times q'(n+t-1) \tag{2.5}$$

$$F = \frac{1}{1+\alpha t}, t = \beta T \tag{2.6}$$

where:

n is the time interval bin being examined (e.g. ten seconds)

$q(n)$ is the arrival flow at interval n (in vehicles)

$q'(n+t)$ is the downstream volume (in vehicles) after t seconds since the start of interval n

F is a smoothing factor (dimensionless)

t is typically set to the fastest expected travel time (typically in seconds)

T is the average travel time between the upstream and downstream intersections (typically in seconds)

α is a calibration parameter for the platoon dispersion rate with respect to time (dimensionless)

β is a calibration parameter representing how much faster than the average the front of the platoon will travel (dimensionless)

α and β typically take on values between 0 and 1, and are provided separately so that individual effects can be isolated as both parameters are ultimately used to scale the travel time between intersections in the basic form of the model. These parameters are usually determined empirically in reference to the natural dispersion of the platoon and ratio of the leading vehicle's travel time to the average travel time of the platoon. Research has also been done to recommend values for these parameters or different functional forms for F. Typically, the value of β is left at 0.8 and α is calibrated for local conditions. Typical values include 0.5 in very urban and busy environments and 0.25 in less urban settings (Manar and Baass, 1996). As part of the development of the HCM, the calculation for the smoothing factor F uses the following (Bonneson, Pratt, and Vanderhey, 2008):

$$F = \frac{1}{1 + 0.138T'_R + \frac{0.315}{d_t}} \tag{2.7}$$

where T'_R is the segment running time and d_t is the step duration.

In recent years, many researchers have looked to improve on the Robertson model, though most of the current state-of-the art still relies on the construct. Many modifications to the calculation of the smoothing factor and have been proposed recently, and many researchers have focussed on using it to capture the dynamism present in traffic. For example, Beess et al. (Beess, 1988) propose a quadratic form to model the volumes at coordinated intersections while Shen et al. (Shen et al., 2018) develop a dynamic speed-based model to model changes in the t parameter and F.

2.2.3 *The Highway Capacity Manual and Canadian Capacity Guide*

The HCM (Manual, 2010) provides the most common set of tools and approaches used when designing and characterising intersections and roadways based on their volumes. In Canada, the Canadian Capacity Guide (CCG) (Teply et al., 2008) contains a similar set of approaches as the HCM but contextualised for Canada.

2.2.3.1 *Traffic Signal Timing*

The HCM and CCG provide a comprehensive and multi-step process to determine optimal signal timings based on known volume inputs. At any given time, a signal can be timing green to a phase and red to others, timing amber, or timing red to all phases. Amber and all-red times are frequently referred to as *inter-green time*, and the remaining green time g_t must be allocated to each movement j requiring service. In the HCM green time is allocated on the basis of the movement's flow ratio y_j in relation to the total sum of all flow ratios Y for critical movements. Critical movements are the movements with the highest flow ratio for each phase. The calculation proceeds according to the equation below:

$$g_j = g_t \times \frac{y_j}{Y} \tag{2.8}$$

where:

g_j is the green assigned to movement j (in seconds)
g_t is the total green available to be allocated (in seconds), which is equal to the cycle time less the total lost time.
y_j is the flow ratio calculated for phase j according to Equation 2.9 (unit-less)
Y is the sum of all flow ratios for critical movements (unit-less)

The flow ratio y_j of a particular movement j can be calculated as shown below:

$$y_j = \frac{q_{j,adj}}{S_j} \tag{2.9}$$

where:

$q_{j,adj}$ is the flow rate through phase group j adjusted for any effects (in vehicles per hour), such as removing right turns on red or turns during the inter-green
S_j is the saturation flow rate for movement j (in vehicles per hour)

The saturation flow is typically calculated in reference to some base saturation rate value but adjusted for the specific conditions of that lane and intersection. The HCM has a comprehensive guide covering most major configurations and a calculation process to adjust these values (Manual, 2010).

The total sum of all greens and inter-greens is the *cycle* time and can also be calculated on the basis of the total flow ratio according to the following equation:

$$c_{opt} = \frac{1.5L + 5}{1 - Y} \tag{2.10}$$

$$c_{min} = \frac{L}{1 - Y} \tag{2.11}$$

where L is the total lost time and Y is the sum of all critical flow ratios. Equation 2.10 is commonly referred to as *Webster's formula* for the optimal cycle length and would minimise the total intersection delay while Equation 2.11 ensures that the intersection remains in an under-saturated state.

2.2.3.2 *Intersection Performance*

For any given signal timing plan and intersection layout, the HCM has a process that allows the delay at the intersection to be estimated. The calculation is divided into two components, the uniform delay d_1 which is based on deterministic queueing theory and the overflow delay d_2 which captures both random effects and over-saturation effects. These equations are based on extensive reasearch done in this area but all rely on some key constants and assumptions. The basic formulation of these values is shown below as given in the CCG as shown below (Teply et al., 2008):

$$d_1 = c \times \frac{(1 - \frac{g_e}{c})^2}{2 \times (1 - X_1 \times \frac{g_e}{c})} \tag{2.12}$$

$$d_2 = \left((X-1) + \sqrt{\left(((X-1)^2) + \frac{240 \times X}{C \times t_e} \right)} \right) \times 15 \times t_e \tag{2.13}$$

$$d_t = k_f \times d_1 + d_2 \tag{2.14}$$

where:

d_1 is the deterministic delay (in seconds)
c is the cycle time (in seconds)
g_e is the effective green time (in seconds), which is expected amount of time that vehicles can discharge
X_1 is the lesser of the actual degree of saturation, X, or 1 (unit-less)
d_2 is the overflow delay (in seconds)
X is the degree of saturation (unit-less)
C is the capacity (in vehicles per hour)
t_e is the evaluation time (in minutes)
k_f is the progression factor (unit-less, between 0 and 1)

The capacity of an approach is calculated as the sum of the individual capacities of all the lanes in the approach. For each lane, the capacity can be determined in reference to the saturation flow, cycle time, and the amount of green time that can actually be used to discharge vehicles, according to the equation below:

$$C = \frac{S g_e}{c} \tag{2.15}$$

where S is the saturation flow rate and g_e is the effective green. g_e is present in both the delay and capacity equations and is derived from the actual green time assigned to the approach. Past research has shown that there is a delay when vehicles respond to the light's change to green and that vehicles are also able to use a portion of the inter-green time to discharge. The HCM provides a process to calculate the effective green time, but a common value recommended by the CCG is to add one second to the actual displayed green time and take that as the effective green.

Given the capacity and flow rate, the degree of saturation X can be calculated as the ratio between the flow rate q and the capacity C. The equations from deterministic queueing do not permit the volume-to-capacity ratio to exceed one, so d_1 caps the value of X to 1. Further, in practise uneven arrivals can result in temporary flows above the capacity of the approach for a few cycles, causing some vehicles to experience additional delay (as they must wait multiple cycles). The d_2 equation provides a mechanism to capture these effects. The observation time is used to limit the scope of the delay and is particularly relevant when X exceeds 1 as queues will grow indefinitely in such cases.

2.2.3.3 *Software Packages Implementing the HCM*

Off-line generation of signal timing plans based on the HCM is usually facilitated with the use of software, such as TRANSYT or Synchro. The latest version of TRANSYT, 7F, has been updated substantially since its first proposal to include new techniques and advances in traffic signal optimisation and uses a macroscopic model to simulate traffic. The model simulates platoons of vehicles, rather than individual vehicles, and uses a platoon dispersion model simulate their behaviour as they leave intersections and travel through the network. Simulation is done in a step-wise manner for smaller time increments (typically one second in length). By simulating the platoon, the arrival time of the constituent vehicles can be determined at downstream intersections. A critical limitation of this approach is its assumption that traffic behaves according to the platoon dispersion model, which may not be the case if disruptions occur mid-block.

TRANSYT 7F is defined to optimise an objective function (called the *performance index*), which can be customised according to the user's needs. Typical performance indexes include optimisation on the basis of delay, progression, stops or fuel consumption. An algorithm is used to find the best cycle length and green split configuration (Ratrout and Reza, 2014; Wallace et al., 1984). The traditional optimisation technique available is the *hill-climbing* technique, which is an iterative gradient search technique. This method optimises the offset and split times separately. It incrementally modifies these parameters at each intersection and calculates the value of the performance index, then chooses subsequent modifications with the goal of finding the optimum value of this function. To avoid being trapped in a local minimum, the algorithm tries larger step-sizes after finding a local-minimum, but there is no guarantee that the final solution will be optimal. Newer versions of TRANSYT also include other optimisation algorithms, such as genetic algorithms, but the fundamental assumptions used to obtain the performance index values do not change.

Like TRANSYT, Synchro functions similarly in the sense that it use macro-scale deterministic traffic modelling and does an exhaustive search to find an optimal cycle length. It differs from TRANSYT in that it is delay-based and calculates its cycle lengths and green splits based on the method prescribed by the HCM (Ratrout and Reza, 2014).

2.2.4 *Cell Transmission Models*

The Cell Transmission Model (CTM) is a macroscopic traffic simulation model proposed by Daganzo et al. (Daganzo, 1994) to operationalise the Lighthill, Whitham, and Richards (LWR)'s macroscopic traffic models while still maintaining the simplicity present in those models. In CTM the roadway is divided into smaller equally sized homogeneous sections (called cells). Typically cells are defined so that vehicles are able traverse the cell in one model calculation cycle (tick) under ordinary conditions but can queue in the cell if they are unable to leave it. Thus, each cell can be described by three functions, the number of vehicles in the cell at time t ($n_i(t)$), the capacity flow into the cell i during time t ($Q_i(t)$), and the maximum number of vehicles that can be housed in a cell at time t ($N_i(t)$). The traffic state of each cell can be described using basic traffic flow conservation laws and fundamental relationships, and the output from each cell is propagated to the next connected cell for each time stamp. The conditions of this model are normally stated using the following two equations (Lighthill and Whitham, 1955; Richards, 1956; Lo, 1999):

$$\frac{\partial q}{\partial x} + \frac{\partial k}{\partial t} = 0 \tag{2.16}$$

$$q = F(k, x, t) \tag{2.17}$$

where:

q is the traffic flow (in vehicles per hour)
k is the density (e.g. in vehicles per km)
x is the position in the cell
t is the time
F is a function relating q and k.

The partial derivative ensures conservation of traffic flow, namely that if the density changes with respect to time, the flow must also increase with respect to space. This model is often called the *kinematic wave model*, and a common solution to the model is given as follows (Lo, 1999):

$$q = \min(Vk, Q, W(k_{jam} - k)) \tag{2.18}$$

where:

k_{jam} is the jam density
Q is the capacity

V is the free-flow speed

W is the speed at which a backwards shock wave would propagate in congested traffic.

This equation is a piece-wise linear function that models flow as equal to the linear product of free-flow speed and density in uncongested conditions, and then decreases the capacity linearly according to the propagation speed of the shock wave W as density approaches the jam density.

Like traditional traffic theory, CTM models can also be used to understand and model traffic behaviour including shockwaves and flow-density-speed relationships.The cell-based approach allows manageable calculation of the effects occurring in each cell, and by relating the effects of surrounding cells to each other, the overall effect on a network can be examined. A representation of the cell structure for a simple network is shown in Figure 2.3.

2.2.5 *Micro-Simulation Approaches*

The HCM and its principles (such as queueing theory) focus on modelling the behaviour of traffic in aggregate. An alternative approach that has gained popularity in recent decades is to model individual driver behaviours and replicate their interactions at a large scale. The underlying approach is commonly called *microsimulation* and has been implemented in many platforms, including VISSIM, Paramics, SUMO, and Aimsun. The following sections focus specifically on car-following models, which are significant contributors to the overall behaviour of microsimulation.

2.2.5.1 *Wiedemann Car-Following Model*

The Wiedemann model is the fundamental model applied in the PTV VISSIM microsimulation platform. The Wiedemann model uses varying thresholds to govern drivers responses to the separation distance and velocity difference between them and the lead vehicle (Higgs, Abbas, and Medina, 2011; Group et al., 2014). This response is shown in Figure 2.4.

A vehicle pair with a positive velocity difference will reduce the distance between them until the distance and velocity difference combination crosses the SDV threshold. This threshold triggers a response in the modelled vehicle to gradually reduce speed (akin to lifting a foot off the pedal) and represents a *perception* threshold where the driver would recognise that a conflict may happen if they do not slow down. If the distance continues to decrease and passes the CLDV threshold, the modelled vehicle slows down even more *reacting* to the lead vehicle. The BX and AX thresholds are typically only crossed in collision states, and crossing BX means the vehicle would be applying maximum braking to avoid a collision.

2.2.5.2 *Krauß Model*

The Krauß model is a gap-based model used in the SUMO platform (Krajzewicz et al., 2012). This model was first formulated by Stefan Krauß (Krauß, 1998) and is based on the gap, acceleration/braking and reaction time of drivers. The functional form of the model is shown below:

$$v_n(t+\Delta t) = \min \begin{cases} v_{n,des} \\ v_n(t) + a_n(v)\Delta t \\ v_{n,safe}(t) = v_l(t) + \frac{g(t) - v_l(t)\Delta t}{\frac{v_l(t)+v_n(t)}{2b(t)} + \Delta t} \end{cases} \tag{2.19}$$

where:

Δt is the simulation resolution (typically 1s)

t is the time of the calculation (in seconds)

$v_n(t)$ is the speed of vehicle n at time t (typically in metres per second)

$v_{n,des}$ is the desired speed of vehicle n (typically in metres per second)

$a_n(v)$ is a function relating the speed of vehicle n to its acceleration (typically in $\frac{m}{s^2}$)

$v_{n,safe}(t)$ is a function returning the safe speed of a vehicle (typically in metres per second)

$v_l(t)$ is a function for the speed of the vehicle l in front of vehicle n (typically in metres per second)

$g(t)$ is a function providing the gap between vehicle l and vehicle n (typically in metres)

$b(t)$ is a function relating the speed of vehicle n to its maximum acceptable braking (typically in $\frac{m}{s^2}$)

In simple terms, this function selects the minimum of the desired speed of a vehicle, the safe speed considering the lead vehicle in front of it, or the maximum speed it could accelerate to in the next time step.

2.2.5.3 *Model Calibration*

Calibration can be an important but time consuming step when working with microsimulation data. Car-following models are often calibrated using trajectory data, such as data from naturalistic driving studies, and many researchers have also developed modifications or alternative models to better fit specific data. For example, Higgs et al. (Higgs, Abbas, and Medina, 2011) use naturalistic driving data to recommend values for the Wiedemann model's thresholds. Their result found that varying the thresholds with the speeds produced better results than using a single constant. Hoogendoorn et al. (Hoogendoorn, Hoogendoorn, and Daamen, 2011) proposed a stochastic car-following model based on the principles used in the Wiedemann model and calibrated it using trajectory data collected from a Dutch freeway.

2.3 BIG DATA TECHNOLOGIES

Although the term Big Data can be applied to a number of different data sources, this research will focus on a few specific sources. In particular, this research will focus on emerging technologies, such as connected vehicles, crowd-sourced data sources, Bluetooth and WiFi detectors, and new detection methods, such as smart traffic cameras. In the near future, modern vehicles will come equipped with devices that broadcast basic information about the vehicle, including its operating parameters. The data provided by these Infrastructure to Vehicle (I2V) communications will open new possibilities to signal control systems, and leveraging their deployment is an important part of the next generation of traffic controllers. In addition to connected vehicle technologies, passive tracking of vehicles using crowd-sourced data and video detectors is already possible, but research is still needed to determine the best ways to use these data sources. The following sections highlight work already done in this area. Development in this area is ongoing, and many companies have developed products that use some of these data sources to produce traffic analytics. One example is Miovision, which provides real-time traffic analytics using existing loop detector data, video data, and WiFi detection (Miovision, 2017).

2.3.1 *Data from Connected Vehicles*

The use of Connected Vehicle technologies in signal control is an emerging area of research in transportation. A number of testbeds are currently in development to evaluate these technologies in field settings. One such project is the ACTIVE-AURORA project, led by teams at the University of Alberta and University of British Columbia (University of Alberta, 2016; Transport Canada, 2016). The system is designed to test connected vehicle technologies, including safety notifications such as collision risks, as well as speed recommendations based on prevailing traffic conditions. The test bed was activated in 2014, and includes infrastructure instalments in both Edmonton and Vancouver. Many other testbeds are in progress in other regions of the world, and the field has seen a lot of interest from researchers.

In addition to field testbeds, many researchers have conducted simulation-based studies to test the use of I2V and V2V communications for the purposes of signal optimisation. One such study was by Goodall et al. (Goodall, Smith, and Park, 2013). Researchers in the study developed a decentralised adaptive traffic control algorithm optimising the signals based on a rolling-horizon approach. This approach involved predicting the value of the algorithm's objective function a short distance into the future and selecting actions that minimised it. The researchers tried this approach using a single variable formulation based on cumulative vehicle delay and a multi-variable approach based on a weighted sum of the delay per second per vehicle, their deceleration rate, and the number of stops each vehicle is expected to make. The study was tested at varying penetration rates on a four signal corridor based on US-50 in Chantilly, Virginia. The researchers found that the system is able to improve the performance when compared to coordinated actuated systems in a penetration rate of at least 50%. Despite its limited performance at low penetration rates, the system was better able to accommodate unexpected demands.

Another study by Yang et al. (Yang, Feng, and Liu, 2021) examined how Connected and Autonomous Vehicles (CAV) technologies could be used to implement signal control. Their study considered mixed penetration rates and proposed a hierarchical framework to allow ordinary vehicles, Connected Vehicles (CV), and CAVs to cooperate together. Their study examined these interactions on a corridor level and used a system whereby the intersection controller estimates and predicts the traffic state and then makes adjustments to the signal timing. Their approach found potential reductions of up to 14% for the delay, a 6.8% reduction in fuel use when connected vehicle technology was used to implement an adaptive systems. With CAV, their study found that delays could be reduced by 33% and fuel consumption by 7.4%.

Similarly, a study by Guler et al. (Guler, Menendez, and Meier, 2014) conducted a limited simulation-based test on two one-way streets. The researchers tested a number of penetration rates, as well as scenarios where traditional signal control parameters (such as minimum green and amber time) were relaxed. The researchers reported that the highest delay reductions were observed for scenarios with low and medium volumes, where the position information gained can be better used to react to unexpected demand changes. Another study by Feng et al. (Feng et al., 2015) developed and tested an adaptive control system based on connected vehicle technologies. The system solves a two-level optimisation problem using two objective functions to select the best phase allocation. At the first level, a dynamic programming approach is used to calculate a performance measure with forward and backward recursion. For each ring-barrier group, in the forward stage the algorithm determines the optimal length of that phase within a certain range for each phase group in the ring, and then passes its results to the lower level optimisation. In the backward stage, the algorithm then retrieves the final optimised allocations for each phase group. In the lower level, optimisation for the specific phase is conducted by minimising an objective function. The researchers evaluate the use of vehicle delay and minimisation of the queue length based on a table of predicted arrivals. Although the positions of connected vehicles are known, the algorithm overcomes the lack of information from unequipped vehicles in lower penetration scenarios by estimating their status using the data available. Their method divided the approach to a signal into three regions: a queueing region, slow-down region, and free-flow region. In the queueing region, the length of the queue for all vehicle types is estimated based on the locations and stopping times of the first and last stopped connected vehicles. They calculate the queue progression speed, v_q, according to the following equation:

$$v_q = \frac{D_1 - D_2}{T_1 - T_2} \tag{2.20}$$

Where D_i and T_i are the times and distances of the first and and second connected vehicles. The queue length, l, is then estimated using the following equation:

$$l = D_1 + v_q(T_c - T_1) \tag{2.21}$$

Where T_c is the current time. The count of vehicles in the queue can then be obtained by dividing this length by an assumed average vehicle length. In the slow-down region, the researchers characterise the system on the basis of the Wiedemann car-following model and use the behaviours of the connected vehicles to infer the locations of the unconnected vehicles. Finally, in the free-flow region, the system is unable to make an accurate estimation on the locations of unequipped vehicles, and simply divided the number of connected vehicles by an assumed penetration rate. The results of their proposed algorithm demonstrated delay reductions of up to 16.55% in high penetration cases when compared to actuated control, and similar performance to actuated control in lower penetration situations. The objective functions each function differently, with optimisation based on total delay producing lower overall delays, and optimisation based on queue lengths serving approaches in a more balanced way.

CV technologies have also shown promise in estimating key parameters required for the calibration of signal timing. A study by Bagheri et al. (**Bagheri2015**) focussed on estimating the time-varying saturation flow rates for use in adaptive signal control. In their work data from CVs was used to estimate saturation flow rate with reasonable accuracy even in cases where market penetration rate was limited to 20%. In addition to saturation flow, their method also covered queue estimation and turning movement ratios.

As an emerging technology, connected vehicles will increasingly play a role in the modern traffic system. While much research has been done on situations with high market penetration rate, low penetration rates can still be leveraged for adaptive signal control. One application of CV technology is estimation of vehicle delay. Feng et al. (Feng, Zheng, and Liu, 2018) demonstrated a method by which delay can be estimated using penetration rates as low as 10%. Their method relied on the identification of critical CVs such as the last CV that stopped at a traffic light and the first CV that did not stop to generate vehicle arrival and departure times. This approach is then used to develop an adaptive signal control algorithm. Their results showed that even when the penetration rate is 10% the delay is reduced by 16%.

2.3.2 *Bluetooth and WiFi Data*

Although connected vehicles are expected to eventually provide position and speed information, in the interim researchers have also studied the use of other data sources to obtain this information. One such method is based on the detection of Bluetooth and WiFi devices. Smartphones are one of the most common devices that can be detected with these methods and have seen increasing use in recent decades. For example, in 2015 it was estimated that 81% of Americans owned a smartphone, more than double the usage rate of 35% in 2011 (Pew Research Center, 2015). Similarly, in Canada in 2017 it was estimated that 76% of people owned a smartphone (Statistics Canada, 2017), and in 2015 was estimated at 58% in China, and 41% in Brazil (Pew Research Center, 2016).

A cellphone or other device equipped with a Bluetooth or WiFi module has a unique 48-bit electronic address called a Media Access Control address that identifies it and allows it to communicate with other devices. Media Access Control (MAC) addresses are usually represented with a hexadecmial string divided into six groups (octets), such as 00:15:5D:15:9E:00, and by tracking the movement of these addresses through a network of detectors, information about the performance of the traffic system can be obtained. This approach is made possible due to the fact that equipped devices constantly send out requests to connect to other devices or networks; requests include the address of the device (Haghani et al., 2010). The range of communication depends on the power rating of the device and the detector, but previous studies have shown ranges of between 150 metres to 200 metres (e.g (Abbott-Jard, Shah, and Bhaskar, 2013; Haghani et al., 2010)). This range can be increased or decreased by increasing the power of the scanner or by filtering on the basis of signal strength (Tong et al., 2017).

Each protocol has different methods that can be used to find devices to connect to, but the scanner does not need to connect to any other device to obtain its MAC address. For example, the Bluetooth protocol provides two stages that can be used to detect devices, an *inquiry* stage (which is typically the only stage used) and *paging* stage (Chakraborty et al., 2010). The method and protocol used affects how often a device is detected, which in turn will have implications on the accuracy of any measurements derived. For example, for Bluetooth the time to discover a device may be as high as 10s, but for WiFi the time may be as low as 1s (Abedi, Bhaskar, and Chung, 2013; Wang, Zhu, and Miao, 2017). However, the frequency of detection will depend on the device's current connectivity and power state, and past research has also shown that some smartphones broadcast probes identifying the device less than one time a minute over all WiFi channels (Freudiger, 2015)

As devices can be captured multiple times by a detector, a method needs to be used to select a detection time when attempting to estimate travel time between successive detectors. Common approaches include using first-to-first detection times or last-to-last detection times (Abbott-Jard, Shah, and Bhaskar, 2013).

Recent studies have also begun exploring the use of travel-times and dwell-times obtained for these devices to improve signal timing. For example, a preliminary study by Hart-Bishop et al. (Hart-Bishop, Hellinga, and Zarinbal, 2016) developed metrics that can be used evaluate traffic conditions on an arterial corridor using Bluetooth and WiFi data. The study proposes the use of Bluetooth detected travel times and dwell times within a single detector to identify when a signal timing plan change would be beneficial. Limited work has been done in using this technology for signal timing control, and more research is still needed to identify ways in which these technologies can be incorporated into the next generation of signal control methods.

2.3.3 *Data from Smart Traffic Cameras*

A chief limitation identified in existing adaptive methods is the reliance on loop detector data. Although other detection methods can be used, the benefits of systems such as SCOOT and SCATS depend on accurate data. One alternative to loop detectors is the use of video-based detection systems. While such systems can be used to provide the locations of vehicles at intersections, it is often simpler to provide metrics such as queue length. This approach has been proposed by researchers such as Zanin et al., who evaluated an image-recognition approach to estimating queue lengths (Zanin, Messelodi, and Modena, 2003). The system was tested over a 6-month period in a suburban area in Italy, and used *presence* based detection (in terms of pixels on the image) to count vehicles. The system was also able to detect situations where traffic movements slowed beyond a certain speed (e.g. 25 kph). The system performed quickly, and was able to identify queue lengths in real time for each monitored lane, and performed well under a variety of conditions (e.g. severe weather and night-time).

In addition to queue length data, researchers have also proposed algorithms capable of tracking vehicle movements at intersections using video data. For example, Bas et al. (Bas, Tekalp, and Salman, 2007) propose and evaluate a video detection system that uses background subtraction to detect and track vehicles as they travel on a roadway. Their proposed system also functions at night using the same principle. During evaluation of their system it was found that counts are generally under-estimated due to the fact that vehicles are sometimes occluded from the camera (e.g. if a large vehicle blocks a smaller one) and fail to be counted.

2.3.4 *Other Analytical Signal Control Strategies*

Wey (Wey, 2000) proposed an analytical approach incorporating platoon dispersion without assuming fixed cycle lengths or phase sequences. The model's objective is to minimise total delay on the network, which is framed as the sum of delays on all phases. These delays are calculated based on the queues at each intersection in the network, and their discharge pattern. A key component that limits the model is the requirement of a model that accurately captures the movement of vehicles through the network. The author applies a platoon dispersion model to replicate the behaviour of traffic after it discharges from a green signal. This creates a limitation similar to that present in other systems such as TRANSYT or SCOOT which requires the assumption of constant travel time and may be unable to capture unusual disruptions in traffic mid-block. The model also requires full information on external inputs, but can be configured to function using a rolling horizon with flow detectors similar to other systems such as SCOOT. The author formulated the problem as a mixed integer linear programming problem and solved it using a modified network simplex algorithm. The idea framed the signal timing problem as a transportation problem with the goal of minimising the total cost (i.e. *delay* of all movements) through a time-expanded network according to the platoon dispersion model. Rather than goods flowing through a network, arcs to represent the vehicles that discharge at a particular time. Vehicle flows are then assigned a "time" to flow through to the next node, and the amount of flow to assign at each node is calculated using the platoon dispersion model. This approach allows quick convergence of the model to an optimal solution.

By creating a model of the traffic network, the effect of signal timing on the network can be understood in terms of the macro behaviour of traffic. Since queues propagate back through the cell structure, the effect of the signal timing on traffic is known for a particular volume input (even if it changes over time), and does not depend on steady-state assumptions. Optimal solutions to the problem can then be obtained by framing the problem of signal control using an objective function. For example, Lo defines delay in a cell as the time beyond what would normally be incurred from free-flow conditions, and frames the objective function as the minimisation of total delay (Lo, 1999).

A key limitation of this approach is the computational effort required to arrive at optimal solutions, even for small intersections. While such approaches can find optimal solutions even in grid-lock conditions, the computational time required to obtain these solutions grows as more networks are added (Lo, 1999). Assumptions are often made to reduce the complexity of the problem (e.g. fixed cycle times), though proposals for situations with dynamic cycle lengths exist (Lin and Wang, 2004). The validity of the solutions obtained is also dependant on the model's ability to capture real-world traffic behaviour, and the model must still assume an input O-D flow that is deterministic.

2.4 MACHINE LEARNING

In recent decades, machine learning applications have demonstrated exceptional performance in solving many tasks that were once too complex for computers to solve efficiently. A variety of techniques fall under the umbrella of *Machine Learning*, including data analysis techniques such as clustering, reinforcement learning, and neural networks.

2.4.1 *Data Clustering*

One of the most significant challenges that arises when working with large amounts of data is the need to efficiently group it. *Data Clustering* is a set of unsupervised machine learning techniques that can be applied to group data together based on their innate characteristics. There are many techniques in common use, such as k-means clustering, density clustering, hierarchical clustering, and probabilistic model-based clustering (Kotu and Deshpande, 2019; Hartigan, 1975).

2.4.1.1 *K-means*

K-means is an old technique that was first proposed by Lloyd in the 1950's (Kotu and Deshpande, 2019) and was further refined and popularised by MacQueen et al. in the 1960's (MacQueen et al., 1967). The fundamental basis of the algorithm is to divide data into k clusters by first seeding them with some initial value and then placing each data point in the cluster that is closest to it. After adding all data points, the average of the k groups is then re-calculated and used as the next value to match data. The process continues iteratively until the cluster values do not change significantly or some maximum iteration count is reached. Importantly, while k means will find a solution, the solution is not guaranteed to be the global optimum even if the algorithm converges and will only be optimal relative to the initial seeding. Various techniques have been devised in recent years to improve this performance: optimal seedings, dynamic estimation of the optimal k value, and other algorithmic changes (MacQueen et al., 1967; Hartigan and Wong, 1979; Hartigan, 1975; Kotu and Deshpande, 2019).

Selection of an appropriate seeding point is the simplest way to improve the optimality of solutions generated from k-means. Optimal seedings will depend on the dataset being used, but traditional approaches include selecting the first x sequential points or evenly spacing them throughout the data (Hartigan and Wong, 1979). Other approaches select groups randomly (Hartigan and Wong, 1979; Kotu and Deshpande, 2019) and repeat them selecting the best performing seeding (Kotu and Deshpande, 2019).

Selection of a value for k is also critical to the usefulness of the clustering results. Early suggestions focussed on ad-hoc rules of thumb such as using $k = \sqrt{n/2}$ (Mardia, 1979). More recently, some researchers have proposed heuristic methods for this process such as the *elbow method* (Ketchen and Shook, 1996). In this approach an indicator of performance is used and is plotted across a range of values for k. The point of inflection where higher values of k do not produce meaningful improvements is then selected. Similarly Pelleg et al. (Pelleg, Moore, et al., 2000) proposed the use of the Bayesian Information Criterion (BIC) as a scoring method and select the value of k that maximises this criteria. Hamerly et al. (Hamerly and Elkan, 2003) proposed a dynamic process beginning with some smaller value of k and splitting clusters with data that do not follow the Gaussian distribution.

2.4.2 *Artificial Neural Networks*

Artificial Neural Networks (ANN) are designed to replicate biological processing units in an algorithmic setting. They operate using a network of individual nodes (called neurons) that have weights $w_0, w_1, ...w_n$ that take input from other sources and generate output based on an activation function. These nodes are typically grouped together into layers that share similar sources of output and input. For a particular neuron j's output o_j, the output can be calculated based on an aggregation of the inputs v_i according to the following process for a given activation function: $o_j = f(\sum_i w_{ji} v_i)$. One of the most common activation functions is the Rectified Linear Unit (ReLU) function, which is a truncated linear function that outputs only positive values. Functionally, it can be represented as below:

$$g(z) = max(0, z) \tag{2.22}$$

where z is the weighted aggregate sum of the neuron's input. Other functions are also commonly used, depending on the output desired and training process. For example, the softmax function gives a normalised probability for each individual output in the layer and is commonly used as the activation function in the final layer of a neural network(Agarap, 2018; Basheer and Hajmeer, 2000).

The process of tuning the weights in a neural network is called training. A number of algorithms can be used for this process, but the most common approach is called backpropagation. Backpropagation relies on a stochastic gradient descent process to update the weights by estimating the contribution of each weight to the observed error according to the following equation:

$$w_i \leftarrow w_i - \frac{\eta(\delta E(E, w)}{\delta w_i} \tag{2.23}$$

where:

w_i is a particular weight

η is the learning rate

E is the error

The learning rate η controls how large the change in weight value will be at the update step. Although high learning rates can allow faster weight updates, if the learning rate is too large the model may oscillate and fail to converge. Similarly, if the learning rate is too small the model may also fail to converge as weight updates may be too small, especially on deeper layers. One strategy to address this is the use of variable rates where a larger learning rate is used initially and then a smaller one is used to fine-tune the model (Krizhevsky, Sutskever, and Hinton, 2012).

2.4.2.1 *Deep Neural Networks*

Due to the difficulty in training, early ANNs typically had 2 or fewer layers. While backpropagation allows training of deep models in theory, in practise the algorithm suffered from a *vanishing gradient* problem as adjustments become exponentially smaller on layers further away from the output. Various strategies were developed to combat these problems and improve the performance of deep networks, including progressively pre-training lower layers as feature detectors and the use of Convolutional Neural Networks (CNN) (LeCun, Bengio, and Hinton, 2015; Schmidhuber, 2015). Convolutional neural networks use smaller groups of weights than would be required in traditional settings. These smaller groups of weights are scanned over the input progressively to generate output in successive layers and have shown excellent ability in learning feature identification in unsupervised settings (LeCun, Bengio, and Hinton, 2015). Another technique commonly applied in CNNs is pooling, which down-samples an input and reduces the number of inputs into the next layer. Pooling layers work similarly to convolutional layers in that they are smaller than the input and progressively scan it to generate the output. Pooling layers apply a function, such as outputting the maximum, minimum, or average value.

2.4.3 *Reinforcement Learning*

Reinforcement learning is a technique used to train an agent experientially to interact with an environment with reference to some rewarding system. The environment and problem the agent must solve are framed in the wider context of Markov Decision Process (MDP). All MDPs consist of four key aspects that an agent can interact with over a time period: a set of definable states S_t, a set of possible actions A_t, a transition function that defines how states change between successive time periods t and $t+1$ based on actions taken by the agent ($P(S_{t+1}|S_t, A_T)$), and rewards that can be measured based on these ($R(S_t, A_t, S_{t+1})$) (Szepesvari, 2010; Watkins and Dayan, 1992). In many applications, a further extension can be taken that limits the agent's ability to observe the true state S_t and the agent instead has to act based on its observations O_t. This extension is called a POMDP (Astrom, 1965; Hoey and Poupart, 2005).

Using the reward function and states, the agent's goal is to then learn a policy $\pi(S)$ that maximises the accumulated reward. In indefinite problems, a time horizon is defined by discounting the rewards by some factor $\gamma \in [0, 1]$. The value of a particular state V can then be calculated in reference to this factor and the future rewards possible from it according to Equation 2.24:

$$V = R_1 + \gamma R_2 + \gamma^2 R_3 ... = \sum_{j=0}^{T} y^j r_{j+1} \tag{2.24}$$

Where V is the value of the state, γ is the discount factor, and R_1, R_2, etc. are the rewards from state $1, 2, 3$, etc. Setting γ to 0 will result in the states only being valued on their immediate reward while setting γ to 1 results in an unbounded value for V since the summation is calculated over an infinite time period. A function V^π can be defined that quantifies the value of rewards expected if the agent acts according to policy π. For each choice that an agent has at time t, an additional quantity can be defined, $Q^\pi(S, A)$, which is the expected value of choosing any specific action A in state S that leads to state S' and then following policy π afterwards. Q can be calculated according to Equation 2.26 (Mnih et al., 2015; Szepesvari, 2010; Watkins and Dayan, 1992):

$$Q^\pi(S, A) = \sum_{S'} P(S'|A, S)(R(S, A, S') + \gamma V^\pi(S')) \tag{2.25}$$

$$V^\pi(S) = Q^\pi(S, \pi(S)) \tag{2.26}$$

When the policy π learned is the optimal policy $\pi*$, the agent maximises these functions and chooses the action that yields the highest Q value. Traditionally, such agents have trained through a process called Q-learning that uses a value-iteration approach to estimate the Q values for each state-action pair according to the process in Equation 2.27 (Szepesvari, 2010; Yau et al., 2017).

$$Q(S,A) \leftarrow \underbrace{Q(S,A)}_{\text{previous Q}} + \underbrace{\alpha\,(R+\gamma \max_{A'\in A} Q(S',A') - Q(S,A))}_{\text{error}} \tag{2.27}$$

where:

S is a particular state
A is a particular action
α is the learning rate
γ is the discount factor
S' is the successor state from S
A' is the optimal action in state S'

In this approach, the estimated Q value for a state-action combination is progressively refined by computing the difference between the Q value and reward R obtained in the subsequent state S' if the optimal action A' is chosen next compared to the Q value predicted in state S'. As with neural networks, a learning rate α can be specified to control how large these updates are.

If the agent is able to visit every state-action pair enough times, traditional Q-learning is guaranteed to converge to an optimal policy. The critical issue is ensuring the agent is sufficiently able to try all states and avoids the trap of a local minima.

One key approach used to mitigate this is called ϵ-greedy which has the agent select a random action with probability ϵ but allows it exploit its learned knowledge and choose optimal actions otherwise. In this approach the value of ϵ is typically annealed to some final low value with the agent transitioning from a high number of random actions to exploiting the learned policy.

Another key approach used is to replace the function $Q(S,A)$ with an approximation. This is particularly relevant in large problems where the combinations of states and actions is very large. One common model used to approximate Q is through neural network that predicts the Q value and has its weights updated based on the difference calculated in Equation 2.27. However, since the approach is an approximation, convergence issues can arise and the agent can make erroneous decisions when states and actions produce very similar outcomes. To regularise training, Experience Replay is often used. Experience Replay maintains a batch of past experiences and samples them randomly for the agent to re-experience during training (Mnih et al., 2015).

2.4.4 *Transfer Learning*

In ordinary applications, machine learning models are trained to solve a particular problem using a training dataset and then applied by handling different but related data. However, in many cases it may not be feasible to collect enough data to fully train a model, or the time to fully train a model makes it undesirable. One strategy that can be used to overcome this is called *transfer learning*. Transfer learning leverages the existing knowledge of a particular agent to reduce the amount of training required for a new agent by jumpstarting the agent's initial performance (Da Silva and Costa, 2019; Pan and Yang, 2009).

Transfer learning is a broad term that encompasses many techniques. Examples include teacher-student style methods where another agent (or human) demonstrates to or advises the agent on optimal choices, and reward shaping where modifications are made to the reward function to guide learning (Da Silva and Costa, 2019). Other approaches include pre-training the model on a specific dataset and then fine tuning it by freezing lower layers or reducing the learning rate (Chronopoulou, Baziotis, and Potamianos, 2019).

Transfer learning can also be used to apply a trained model on a related but different problem. A simple example of this comes from image classification problems. A model pre-trained on a broad set of images can be further fine-tuned to classify images into more categories by extending the training or to a domain-specific set of image categories (Pan et al., 2018)

2.4.5 *Traffic Control Applications*

All current traditional adaptive control systems share one common element: a model of the traffic system is generally required for the system to function. For example, the SCATS, SCOOT, and RHODES models discussed earlier all contain models that prescribe a relationship between the observations made by the vehicle detectors and traffic parameters such as expected queue length or arrival time. This approach necessitates the use of assumptions that may not be universally valid, and are often highly generalised, such as platoon dispersion models.

In their study, Arel et al. (Arel et al., 2010) evaluate their proposed model on a network with 5 intersections. In their network the central intersection is controlled by a Q-learning based Reinforcement Learning (RL) controller, and the four surrounding intersections are governed by an algorithm that serves the longest queue first. The system state in their algorithm was represented as an 8 dimensional feature vector where each element represents the relative flow in the system's lanes (two lanes on each approach were used). The action set was defined with a two-ring structure, but is free to choose any phase order. The reward function divides time into steps, and penalises or rewards the controller based on a comparison between the delay in the current time step and the previous time step. The proposed controller performed better than the controller based on longest queue first, but the study is limited in that it does not compare its system to various existing controllers.

Abdulhai et al. also evaluated the use of a Q-learning model, and compared its effectiveness over time (Abdulhai, Pringle, and Karakoulas, 2003). Test scenarios were conducted using three different traffic profiles, and were compared to a pre-timed signal plan. In their study, the Q-learning controller performed on par with, or slightly better than the pretimed controller. The Q-learning agent was able to produce significantly lower delays (between 38 to 44%) in cases with highly variable traffic flows. This work was extended by Samah El-Tantawy et al. (El-Tantawy, Abdulhai, and Abdelgawad, 2013) who proposed a multi-agent reinforcement learning strategy (called MARLIN). Their approach proposes two possible modes of operation: an independent mode where each controller works without cooperating with its neighbours, and an integrated mode where controllers coordinate their actions with surrounding neighbours. The proposal addresses the issue with dimensionality by introducing a coordination mechanism that considers the actions, states, and rewards of neighbouring intersections. In their system, the state s of the system is characterised in terms of queue length, with a vector of $2 + P$ components, with P being the number of phases. The first two components are the index of the phase currently showing green and the time since the phase started, respectively. The remaining P components are the maximum queue lengths associated with each phase. The controller can choose between two actions, extending the current phase or switch to any other phase based on traffic fluctuations (skipping phases is permitted). The system is rewarded based on the difference in delay between two successive decision points, with positive rewards corresponding to reductions in delay, and negative rewards to increases in delay. The system is notable in that it was compared to an extensive simulation of the City of Toronto's network for the morning rush hour. The results of their simulation showed intersection delay reductions on the busiest routes in Downtown Toronto of 27% in the independant mode and 39% in the integrated mode, corresponding to a travel-time savings of 15% and 26% respectively.

In addition, recent research has also sought to take advantage of the possibilities enabled by autonomous vehicles. For example, a study by Dresner et al. (Dresner and Stone, 2008) proposes a system where each vehicle is an autonomous agent and traditional traffic lights are not used to time signals. In their system, approaching agents contact a signal controller and attempt to reserve a block of space-time to pass through the intersection. The intersection manager then decides if the request is granted or not based on a control policy. The system is compared to a hotel reservation system, where an agent attempts to book rooms in a hotel, occupying a certain amount of space (number of rooms) for a certain length of time.

Recent developments in Machine Learning (especially deep learning) have led to extensions to the basic Q-learning framework. Typically called *deep learning*, it differentiates itself from ordinary learning models by inserting a number of hidden layers with weights between the input and output layers. Hidden layers use non-linear transformations to generate the values of successive hidden layers based on the original input. Typical functions employed are logistic, tanh, or ReLU, which are applied to the input of each previous layer to obtain the current layer's representation of the input. In this sense, deep learning can be thought of as a filtering process, whereby important information from each layer is gleaned until finally arriving at the output. After reaching the output, the error is estimated, and backpropagation is used to adjust the weights of the hidden layers.

Research in the area of using deep learning to learn the Q-function of RL systems is a new field, and research continues to be done in this area. Applications of these techniques to traffic signal control have been limited, but some work has already been done in this area. A recent study by Li et al. in 2016 used deep Q-learning to train a signal controller at an isolated intersection (Li, Lv, and Wang, 2016). The authors compared the training speed of their deep Q model to that of an ordinary Q-learning implementation and found that deep Q model performed better, reducing average delays by about 14%. Improvements were also found in queue lengths and stopped vehicles. Another study by Genders and Razavi developed a deep reinforcement learning model and evaluated it on a single intersection using SUMO (Genders and Razavi, 2016). In their study, the state space was represented as three vectors discretising traffic approaching the intersection. Their method divides the approach into cells, with the first two vectors indicating if a vehicle is present in the cell and its speed, and the last vector indicating the current signal state. The controller could then choose from four actions, giving 2 seconds of green to the North-South Through, North-South Left, East-West Through, or East-West Left. If the controller's choice results in a phase change, then 5 seconds of transition time was added before the controller could make a new choice. Their study compared their agent to a one-layer neural network controller, and found that it reduced delays by 82%, queue lengths by 66% and travel times by 20%. Despite this, these studies have only considered one type of restricted DRL model as applied to single isolated intersections with simplified traffic patterns. Furthermore, no studies have been conducted on developing and evaluating alternative DRL models for traffic signal control at various levels of network scale. Research on this subject has also not taken into account the increasing availability of various new big data sources, as described in the previous section.

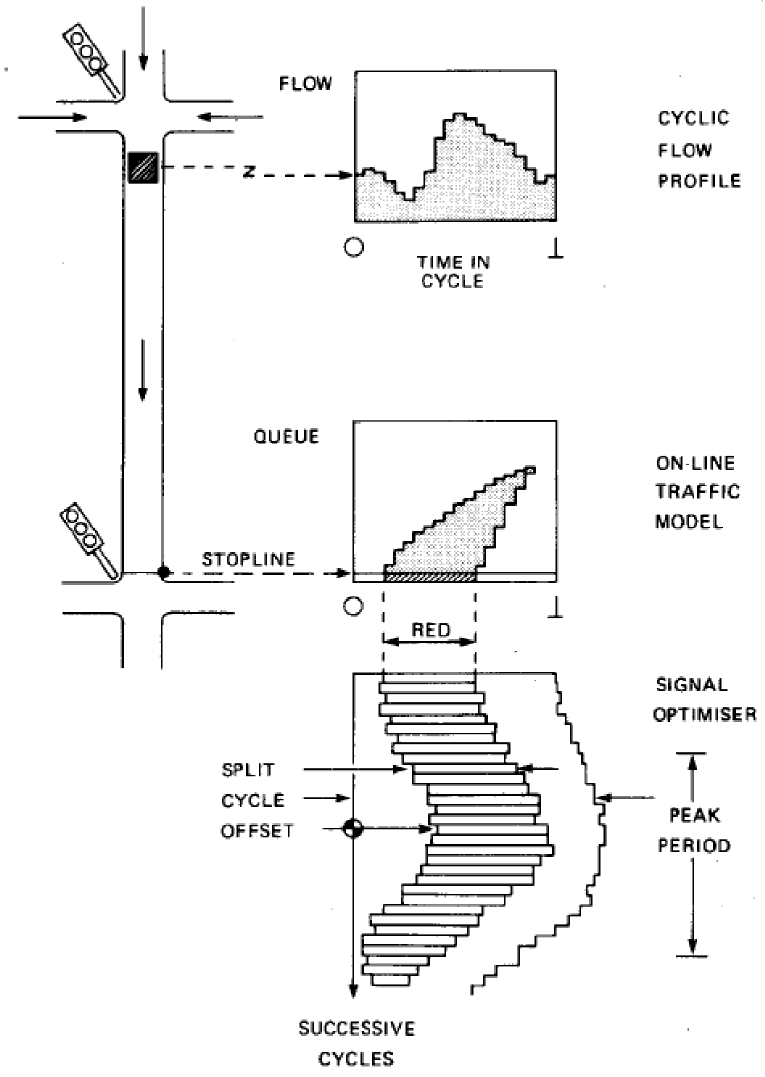


Figure 2.2: SCOOT Traffic Control System, Reproduced from Robertson, 1991 (Robertson and Bretherton, 1991)

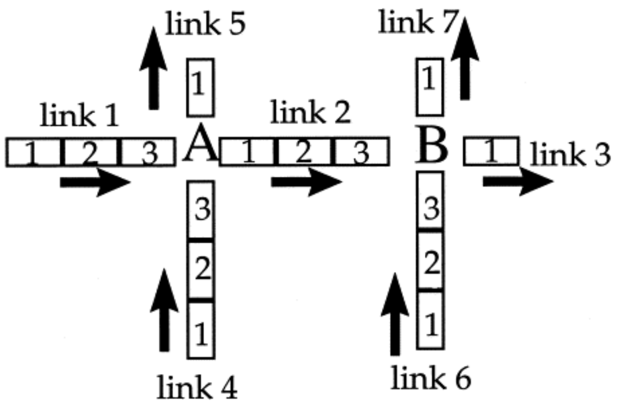


Figure 2.3: Cell Model of a Simple Network, Reproduced from (Lo, 1999)

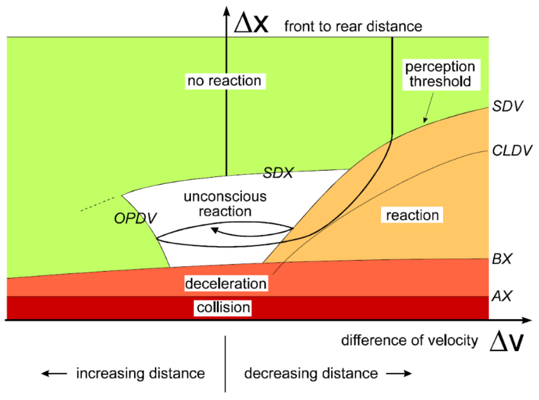


Figure 2.4: Car-Following in the Wiedemann 74 Model, Reproduced from (Fellendorf and Vortisch, 2000)

Part II

ALTERNATIVE TRAFFIC SIGNAL CONTROL METHODOLOGIES

This part is divided into three chapters. Each chapter covers a major component of the research done. The first focusses on a Big Data application outlines a method to re-time signals with travel time data. The second and third chapters both detail the development of a deep reinforcement learning algorithm for traffic signal control, with the one chapter focussing on the case of isolated intersections and the other on multiple intersections.

3 USING TRAVEL TIME TO RE-TIME SIGNALS

The Big Data environment created by recent technological advances gives new opportunities for traffic engineers and system operators to improve their operations. As discussed in Section 2.3.2, one of these Big Data sources is travel time data collected through Bluetooth and WiFi monitoring. Presently, these data are often collected as a metric to evaluate the performance of intersections. This section proposes a method by which travel time data can be used to re-time signals. The method draws on the HCM process for re-timing signals and could be applied in near real-time or on a periodic re-timing schedule. Two scenarios are then explored to consider the possibility of a travel-time based near-real-time adaptive signal controller. In the first scenario errors are applied to hypothetical measurements and a Monte Carlo simulation approach is used to assess the impact. In the second scenario virtual detectors are simulated in a microsimulation platform and travel times and delays are estimated directly.

3.1 SYSTEM SETTINGS AND PROBLEM DESCRIPTION

Consider the operation of a typical signalized intersection controlled by a fixed timing plan, as shown in Figure 3.1. Without loss of generality, it is assumed that the current signal plan consists of J phases (Φ_j for Phase j, $j \in J$) with a cycle time of c. For each phase Φ_j, the total green duration, amber and all-red are also known, denoted by g_j, a_j, ar_j respectively. Each phase is assigned to a given number of movements, with (i, j) representing the movement i in Phase j. Movements include through, right, and left and a phase may have any combination of movements assigned to it.

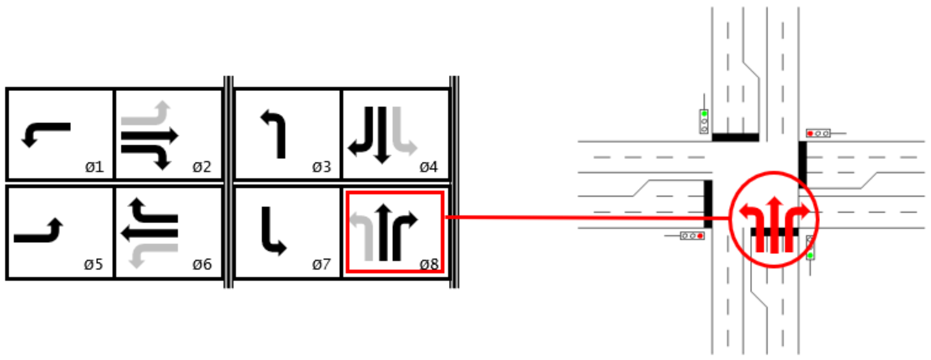


Figure 3.1: Signalised Intersection: Layout, Movements, and Signal Timing

In the traditional approach, signal timing plans are generated and updated on the basis of intersection layout and Turning Movement Counts (TMC), following the procedure specified in HCM (Manual, 2010) and CCG (Teply et al., 2008). TMCs are often obtained periodically through field studies, which could also include delay study for the purpose of evaluating the performance of the existing signal timing plan. In the typical workflow, information on intersection layout (e.g., approach, lane grouping and movements) and TMCs are first used to determine the optimal phasing plan, cycle time and green split. Note that the phasing plan may include elements such as protected left turn phases where only left turning traffic discharges or mixed through and right movements where multiple movements and lanes will discharge. As discussed previously, this process of signal timing or re-timing is time-consuming and the resulting plans may become suboptimal quickly due to constant changes in traffic demand and the road network (e.g., accidents and construction).

In this research, the problem of signal re-timing using a new data source is considered, specifically, data from Bluetooth and WiFi detectors, which can be deployed at multiple locations for measuring the times that individual vehicles with Bluetooth and WiFi devices pass each location. These detected times can then be used to estimate travel time and delay between specific locations. For the purpose of signal optimization at an isolated intersection, it is assumed that Bluetooth and WiFi detectors are deployed at the entry of each intersection leg so that travel time data and then delay can be collected for all individual movements at the intersection. The arrangement assumed is for detectors at mid-blocks, but any arrangement that allows per-movement travel times to be collected would be suitable. Let $d_{t,i,j}$ be the sample mean and $s_{t,i,j}$ be the sample standard deviation of the measured delay for vehicles of movement i in phase j, in seconds, estimated from Bluetooth/WiFi data collected over the current time interval t (note that t could cover multiple cycles). With these data, the first problem of interest is to determine the optimal adjustments on the current signal timing plan on a rolling horizon basis, that is, determining the optimal cycle length and green split that can be implemented in the next time interval, based on the observed travel time and delay of individual movements over the past periods.

In addition to signal timing for isolated intersections, this research also considers the problem of how to make use of travel time data for improving signal coordination of multiple intersections. In a multi-intersection configuration, detectors are placed at the mid-block points between each intersection. It is assumed that each approach leg must have an upstream detector before it can be considered for optimisation. Similarly to the case with the isolated intersection, the signal timing can be optimised at each intersection k for each phase j ($\Phi_{i,j,k}$). An additional decision variable is considered in multi-intersection configurations, the offset o_k at each intersection k, which is the time difference between the start of green at intersection k and a reference clock. Detectors upstream on each approach leg can then also be used to obtain $d_{t,i,j,k}$, the sample mean of the delay over time interval t for movement i of phase j at intersection k.

3.2 METHODOLOGY

As delay is typically a performance measure and may be subject to error, a process was designed to mitigate the effects of random variation. This process relies on incremental changes to the signal timing where the measured delay values are first used to estimate a new signal timing plan. Then, following the estimation of this plan an incremental change is made towards this plan. The following sections provide an overview of the critical steps that are used to implement this approach.

Section 2.2.3 gives an overview of the HCM process used to time signals. One of the key metrics discussed is Equation 2.14 that allows estimation of the total delay from the signal timing parameters and Degree of Saturation, $X_{i,j}$ for each movement i in phase j. $X_{i,j}$ is the volume-to-capacity ratio and is calculated using the demand for the approach and the parameters of the signal timing and design. The Degree of Saturation is a critical component of the HCM process to signal timing and is an intermediary value that is part of several key steps of the signal timing process. The proposed method adopts the approach shown in Figure 3.2 to use estimates of $X_{i,j}$ obtained by using the HCM's delay equations. The following subsections provide additional details on each step of the process for each approach. Psuedocode for the model developed is provided in the appendix in Section A.3.

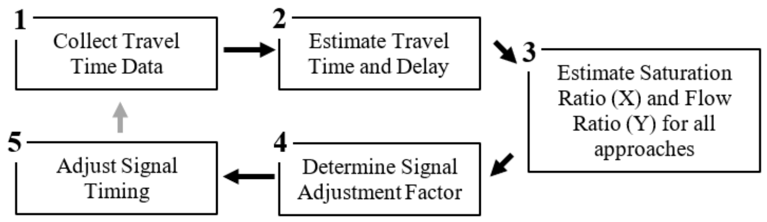

1
Collect Travel Time Data
2
Estimate Travel Time and Delay
3
Estimate Saturation Ratio (X) and Flow Ratio (Y) for all approaches
4
Determine Signal Adjustment Factor
5
Adjust Signal Timing

Figure 3.2: Overview of Split and Cycle Adjustment Procedure

3.2.1 *Estimation of Saturation Flow and Flow Ratio*

The HCM delay equations are used to estimate the value of $X_{i,j}$ for any given delay value. These equations are discussed in Section 2.2.3.1. Equations 2.14 and the constituent Equations 2.12 and 2.13 are primarily used in this analysis. These equations are also listed below, contextualised for a particular movement i in phase j, as Equations 3.1, 3.2, and 3.3 respectively.

$$d_{1,i,j} = c \times \frac{(1 - \frac{g_{e,j}}{c})^2}{2 \times (1 - X_{1,i,j} \times \frac{g_{e,j}}{c})} \tag{3.1}$$

$$d_{2,i,j} = \left((X_{i,j} - 1) + \sqrt{\left(((X_{i,j} - 1)^2) + \frac{240 \times X_{i,j}}{C_{i,j} \times t_e} \right)} \right) \times 15 \times t_e \tag{3.2}$$

$$d_{t,i,j} = k_{f,j} \times d_{1,i,j} + d_{2,i,j} \tag{3.3}$$

In this work, the equation inputs are obtained for each movement and phase and include the total delay ($d_{t,i,j}$), cycle time (c), and effective green time ($g_{e,j}$), which is the amount of time actually used by vehicles when discharging. Estimation of the capacity ($C_{i,j}$) of a particular movement is also required when calculating overflow delay ($d_{2,i,j}$) using the HCM process. In under-saturated conditions $d_{1,i,j}$ as calculated in Equation 3.1 dominates and is the most important quantity. Importantly, for this equation, estimation of $X_{i,j}$ only requires a further assumption on the relationship between the effective green $g_{e,j}$ and the displayed green g_j. The effective green is the actual amount of green time used, and in the CCG is taken as $g_j + 1$ (Teply et al., 2008). This is the assumption used in this research as well. $d_{2,i,j}$ becomes important as the calculated value of $X_{i,j}$ approaches 1 and dominates for higher values of $X_{i,j}$. For this equation, a further two assumptions are required: an evaluation time t_e must be chosen and a saturation flow rate, $S_{i,j}$, must be assumed to calculate the capacity $C_{i,j}$. In this research assume a value of $t_e = 15$ and use a base saturation of 1800 to calculate the adjusted saturation according to the HCM process for each approach. These assumptions are used in the calculation of the capacity of each movement, which is an input parameter when calculating the overflow delay ($d_{2,i,j}$), but not required to estimate the uniform delay ($d_{1,i,j}$). Consequently, the assumption on the base saturation flow rate is generally not significant for $X_{i,j} < 0.7$, but care should be taken to ensure that the evaluation time and base saturation are appropriate for higher capacity situations..

Due to the non-linearity of $d_{1,i,j}$ and $d_{2,i,j}$, a golden section method is used to back-calculate the value of $X_{i,j}$ from Equation 3.3. This method is suited for this approach since $d_{t,i,j}$ always increases as $X_{i,j}$ increases if all other values remain constant. The search is conducted with an initial bound of $X_{i,j} \in (0, 2)$ and if the delay value exceeds these bounds the corresponding bound is assigned as the $X_{i,j}$ value. In all other cases, the search proceeds by iteratively decreasing the search window based on the equations below.

$$X_{i,j,upper,new} = X_{i,j,lower} + \frac{X_{i,j,upper} - X_{i,j,lower}}{G} \tag{3.4}$$

$$X_{i,j,lower,new} = X_{i,j,lower} - \frac{X_{i,j,lower} - X_{i,j,upper}}{G} \tag{3.5}$$

where G is the golden ratio (≈ 0.618). Iterations continue until $X_{i,j}$ is within 0.0001 seconds of the measured delay.

With the estimated value of $X_{i,j}$, the flow ratio $y_{i,j}$ can be estimated. Using the equations in the HCM, it is possible to convert between the degree of saturation $X_{i,j}$ and flow ratio $Y_{i,j}$ using the following equation:

$$y_{i,j} = X_{i,j} \times \frac{g_{e,j}}{c} \tag{3.6}$$

where:

$y_{i,j}$	is the flow ratio for a particular lane (unit-less)
Xi, j	is the degree of saturation (unit-less)
$g_{e,j}$	is effective green time (in seconds)
c	is the cycle time (in seconds)

In this process an assumption is made that the flow ratio of all lanes in the movement have the same delay and degree of saturation, and therefore the flow ratio value calculated for a particular lane is representative of all lanes in that movement (so long as they are not shared). In situations with shared lanes an assumption on the proportion of flow going in each direction is required. Where relevant in this research prior volumes are used to inform this allocation. $y_{i,j}$ can then be used to re-time signals according to the HCM process described in Section 2.2.3.1. For situations where $d_{2,i,j}$ is insignificant this means that signals can be re-timed without requiring knowledge of or an assumption on the saturation flow-rate of the intersection.

3.2.2 *Determination of the Optimal Signal Timing Adjustment*

Using the HCM process, signals can be re-timed based on the estimated $Y_{i,j}$ values. Equations 2.8, 2.10, and 2.11 discussed previously in Section 2.2.3 can be used to determine a new cycle time (c_{opt}) and green splits once $Y_{i,j}$ has been calculated.

Although the new timing plan could be readily adopted, this can cause an overreaction to a high-burst of traffic on one approach that is not sustained. A step-wise approach similar to methods employed in SCOOT is therefore proposed. In this approach signal timing is adjusted subject to some maximum adjustment γ for each cycle. In the proposed method a sigmoid function is used to restrict the adjustment proportionally, as given in the following equations:

$$c_{new} = c_{old} + \Delta c \tag{3.7}$$

$$\Delta c = \text{ROUND}\left(\frac{F_c}{\sqrt{\beta_c + F_c^2}} \times \gamma_c\right) \tag{3.8}$$

$$F_c = \frac{\max(c_{min}, c_{opt}) - c_{old}}{\gamma_c} \tag{3.9}$$

$$\tag{3.10}$$

$$g_{j,new} = g_{j,old} + \Delta g_j \tag{3.11}$$

$$\Delta g_j = \text{ROUND}\left(\frac{F_{g,j}}{\sqrt{\beta_g + F_{g,j}^2}} \times \gamma_g\right) \tag{3.12}$$

$$F_{g,j} = \frac{g_{j,target} - g_j}{\gamma_g} \tag{3.13}$$

where:

ROUND rounds the value to the nearest whole integer seconds

c_{new} is the new cycle time (in seconds)

Δc is the increment to adjust the cycle by (in seconds)

γ_c is absolute value of the maximum adjustment possible (in seconds)

β_c is a calibration parameter that controls how sensitive the the sigmoid function is (unit-less)

c_{min} is the minimum cycle length (in seconds) as outlined in the HCM

c_{opt} is the optimal cycle identified by using the estimate $X_{i,j}$ values (in seconds)

$g_{j.new}$ is the new green time (in seconds) for a particular phase j

$g_{j,old}$ is the old green time (in seconds) for a particular phase j

Δg_j is the increment to adjust the green by (in seconds)
βg_j is a calibration parameter that controls how sensitive the sigmoid function is (unit-less)
γ_g is the absolute value of the maximum amount to adjust the green by (in seconds)
$g_{j,target}$ is the green value to adjust to computed based on the delay value (in seconds)

Fundamentally these equations are used to scale the adjustment to some maximum value. A sigmoid function is used to dampen the changes and can be used to ensure that the maximum change is only applied when the gap between the target plan calculated based on the delay is drastically different than the current plan. Due to the error that may be associated with the delay values, this allows high adjustments to only be made when the calculated ideal plan differs from the current plan by a significant amount. β_c and β_g can be used to control how linear-like the equation's behaviour is and different thresholds between the green split adjustment and cycle adjustment can be used for all the β and γ values. In situations where the γ value is small, the β be set to a higher value to create a threshold by which the target plan must differ by before changes will be applied.

3.2.3 *Determination of the Optimal Offset Adjustment*

A multi-step process is used to identify the optimal offset as some parameter values depend on the results of other intersections. For the cycle length, each intersection can compute its optimal cycle c_k that is used to select the optimal cycle c for the corridor. With the optimal cycle selected for the corridor, green times can be optimised at each intersection. Then, following this, the offset is optimised. Optimisation of the offset requires prediction of how vehicles will move through the corridor. The model applied in this research is the Robertson model, which depends on the flowrate at the upstream intersection q_k, the flowrate at the downstream intersection q'_k, the travel time T between the two intersections, and calibration parameters α and β.

The offset controls the lag between the start of green at all intersections. The proposed procedure for offset adjustment is shown in Figure 3.3. In coordinated configurations with multiple intersections, the degree of saturation $X_{i,j}$ (also called the volume to capacity ratio) calculated can be used to directly estimate the demand for a particular approach. This requires specification of an appropriate adjusted saturation flow value and calculation of the capacity of all relevant movements. Downstream volumes are computed in reference to upstream movements that would provide volume, and volume between intersections is assumed to be insignificant. With the demand, predictions can be made on downstream arrivals and the offset can be adjusted accordingly. In this research the Robertson model as discussed in Section 2.2.2 is applied to predict downstream flows using Equations 2.5 and 2.6, which are shown below as Equations 3.14 and 3.15.

$$q'(n+t) = F \times q(n) + (1-F) \times q'(n+t-1) \tag{3.14}$$

$$F = \frac{1}{1+\alpha t}, t = \beta T \tag{3.15}$$

Where:

n is the interval bin being examined (e.g. ten seconds)

$q(n)$ is the arrival flow at interval n (in vehicles)

$q'(n+t)$ is the downstream volume (in vehicles) after t seconds since the start of interval n

F is a smoothing factor (unit-less)

t is typically set to the fastest expected travel time (typically in seconds)

T is the average travel time between the upstream and downstream intersections (typically in seconds)

α is a calibration parameter for the platoon dispersion rate with respect to time (unit-less)

β is a calibration parameter representing how much faster than the average the front of the platoon will travel (unit-less)

A deterministic discharge model is used to model departure rates at intersections and the resulting delay. Computation of the arrivals requires estimation of the capacity of the intersection $C_{i,j}$, which is then multiplied by the degree of saturation $X_{i,j}$. Computation of the capacity requires an estimation of the saturation flow rate $S_{i,j}$. This is calculated based on a pre-specified base saturation flow rate and pre-specified adjustment factors for each intersection's movement. The computation proceeds as outlined in the HCM and CCG for each movement and is done in conjunction with the signal timing parameters. For permissive left turns an iterative approach is used where through movement volumes are first estimated for use in the capacity calculation. With the volume estimate obtained, the expected queue accumulation at each movement is modelled at each intersection using deterministic $D/D/x$ queuing. When a particular phase j is green, signals are assumed to discharge vehicles for each movement i at their respective saturation rates S_i. These discharge rates are then used in the Robertson platoon dispersion model specified in Equations 3.14 and 3.15 to predict the arrival rates at each downstream intersection. Using the predictions from the Robertson model, each intersection is optimised sequentially by selecting the offset adjustments from a range of $[-\gamma_o, +\gamma_o]$ that provides the highest total count of vehicles arriving on green.

3.3 CASE STUDIES

To evaluate the proposed method, two approaches were used. In the first approach, a simple simulator was created to generate delays based on the CCG equations, and errors were then applied and a Monte Carlo approach was used to gauge the sensitivity

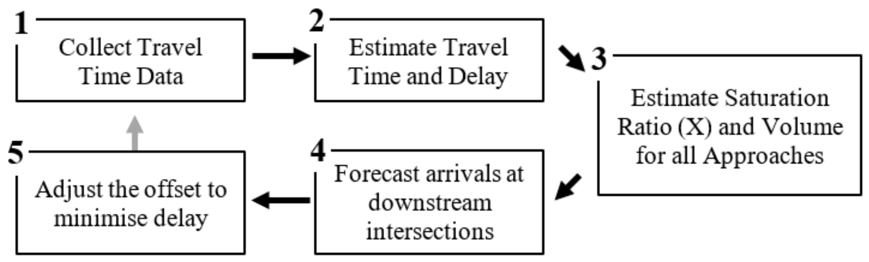

1
Collect Travel Time Data
2
Estimate Travel Time and Delay
3
Estimate Saturation Ratio (X) and Volume for all Approaches
4
Forecast arrivals at downstream intersections
5
Adjust the offset to minimise delay

Figure 3.3: Overview of Offset Adjustment Procedure

of the model to those errors. In the second approach, micro-simulation was used and Bluetooth and WiFi detections were replicated in the simulation platform.

3.3.1 *Monte Carlo Simulation*

Preliminary evaluation was conducted using the Monte Carlo Simulator and a variety of settings were tested to gauge the performance of the proposed method. The following section highlights the effects studied. In all preliminary studies the values of γ and β were set to 1 and adjustments are made after every cycle.

3.3.1.1 *Scenario Settings*

In the Monte Carlo Simulation a single isolated intersection with one lane for each movement is modelled and the effect of error, γ, and β were studied in a sensitivity analysis. As an isolated intersection, this study only examines the split and cycle adjustment methods. Multiple volume scenarios are examined, including constant volumes, a volume scenario that steadily increases from zero to a high value, and a real-world volume scenario. The real-world volume is shown below in Figure **??**. For each volume scenario, delay values are generated based on the volumes at specific intervals and errors are applied to these delay values based on assumed standard deviations. Two error scenarios were evaluated, in the first an error of $\sigma(\epsilon) = 1s$ was applied and in the second an error of $\sigma(\epsilon) = 5s$ was applied.

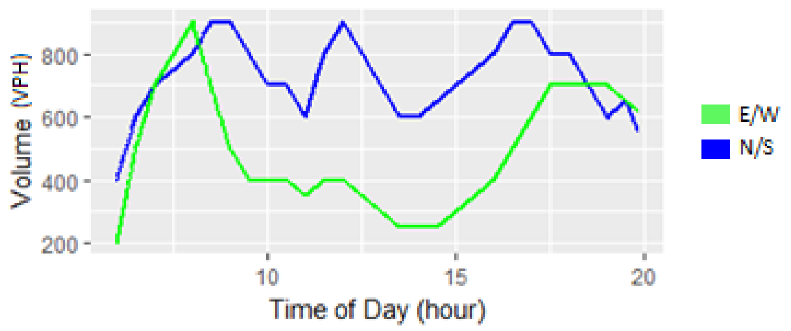


Figure 3.4: Monte Carlo Simulator Volume

3.3.1.2 *Travel Time and Delay*

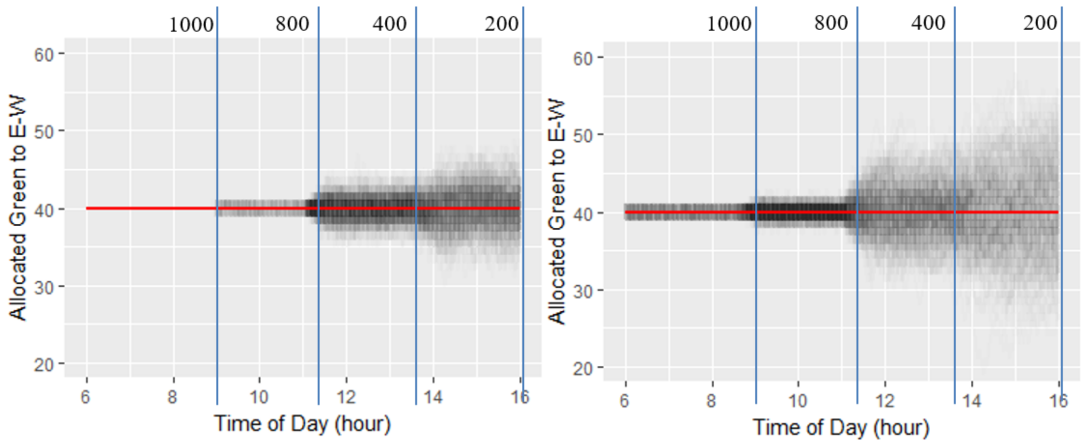
In the Monte Carlo simulation predefined volumes are first generated and a simple deterministic simulator was developed. The deterministic simulator is a period-by-period simulator that uses the HCM equations to predict the delay experienced by vehicles per cycle and averages it over each optimisation period. These delay values are treated as the ground truth, and the best decision the controller could make would be to use this value to make a signal timing adjustment. A key limitation of this simulator is that it is not designed to simulate over-saturated conditions as, for simplicity, it is assumed that the queue clears after cycle change and that queue accumulation does not affect subsequent delays. An error is applied to the ground truth delays for each iteration of the Monte Carlo Simulator to obtain an overview of the impact of measurement error on the algorithm. Errors are applied additively to delay values according to the normal distribution with multiple predefined standard deviations.

3.3.1.3 *Effect of Volume*

Volume was expected to have a strong effect on the performance of the algorithm. There are two primary reasons. The first is that lower volumes mean less accurate delay measurements as there are fewer samples. The second is that lower volumes typically produce lower delay values which amplify the effect of the error as even errors of one second will become significant. Figure **??** shows the performance of the algorithm as the volume varies through 1000vph, 800vph, 400vph, and 200vph for each approach. In this design, the cycle length of the intersection is held constant, so the expected delay predicted by the algorithm becomes lower as the volume decreases. Since both approaches have the same volume, the optimal decision is always to maintain the allocated green to the seeded value of 40s for each approach. This decreasing delay magnifies the effect of the applied error such that at low volumes the error dominates the decision making process and significant degradation is observed when volume drops to 200vph.

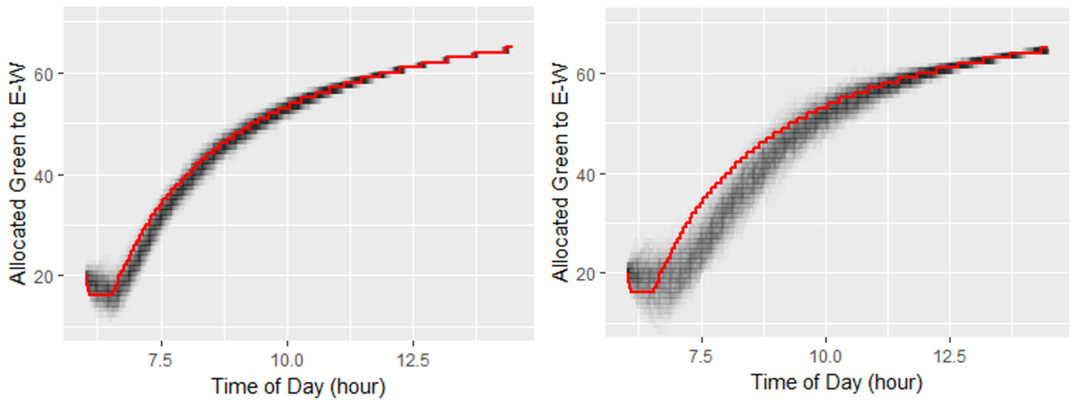
3.3.1.4 *Responsiveness to Demand Change*

A further test was conducted to gauge the responsiveness of the algorithm to dynamic traffic variations. In this test the volume on the E/W approach varies linearly from 100vph to 1700vph across an 8 hour range, with a 30 minute warmup period of constant volume. The opposite approach's volume is held constant at 400vph. The results are shown in Figure 3.6. As before, cycle length is not changed. Errors applied to the delay measurements produce a noticeable lag in the decisions made by the controller at low volumes. This effect diminishes as the volume increases regardless of the error applied, but in the high error scenario it can be observed even at volumes of 800 vph.



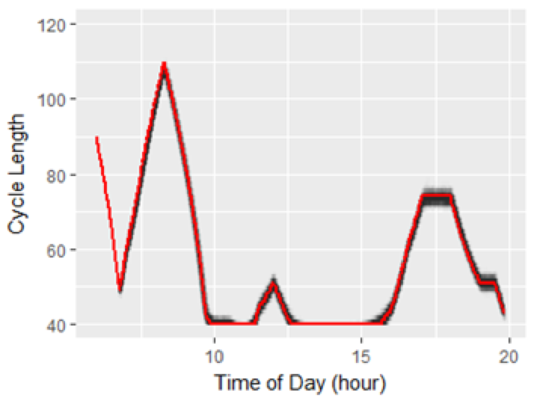
(a) $\sigma(\epsilon) = 1s$ (b) $\sigma(\epsilon) = 5s$

Figure 3.5: Volume Impacts on Split Decisions across 10,000 runs (decisions with no error shown in red)

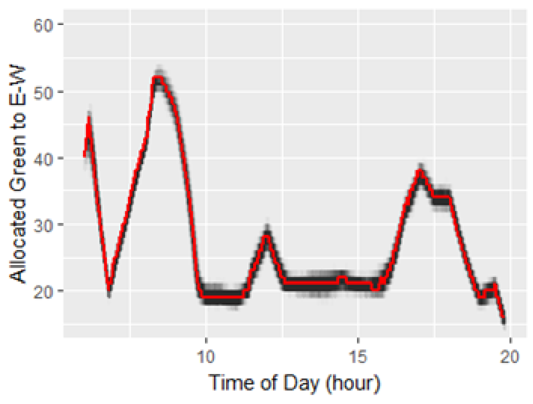


(a) $\sigma(\epsilon) = 1s$ (b) $\sigma(\epsilon) = 5s$

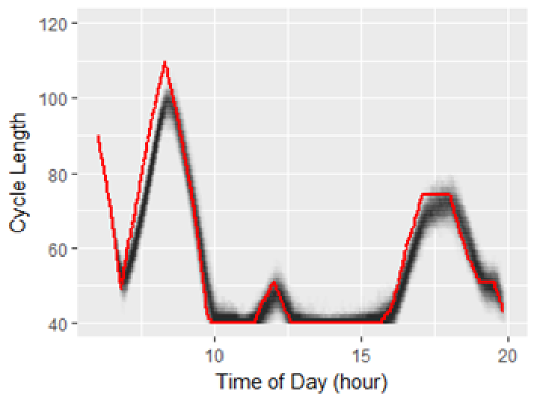
Figure 3.6: Reaction Speed across 10,000 runs (decisions with no error shown in red)



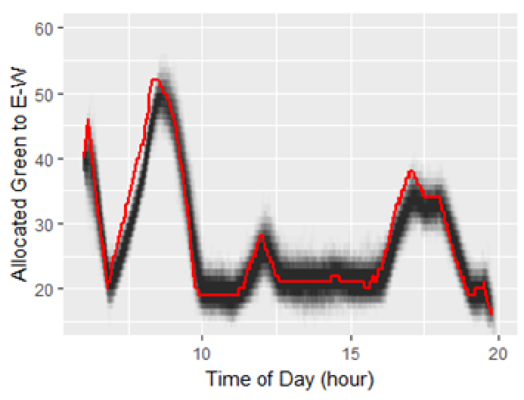
(a) Cycle Adjustments, $\sigma(\epsilon) = 1s$



(b) Split Adjustments, $\sigma(\epsilon) = 1s$



(c) Cycle Adjustments, $\sigma(\epsilon) = 5s$



(d) Split Adjustments, $\sigma(\epsilon) = 5s$

Figure 3.7: Cycle and Split Decisions across 10,000 runs (decisions with no error shown in red)

3.3.1.5 *Cycle and Split Adjustment*

A key limitation of the algorithm is clearly visible in situations with low volumes as these will also generally lead to lower delays. However, when the cycle time is not adjusted, low delays also produce low values of $X_{i,j}$ which reduce the effect of timing changes on the delay due to the under-saturated nature of the intersection. The real-world volume scenario shown in Figure 3.4 is applied to evaluate the performance of the split and cycle adjustment across a wide combination of volumes. Cycle times are bounded by a maximum of 120 seconds and a minimum of 40 seconds. The results are shown in Figure 3.7. As in previous sections, a noticeable lag is observable in decisions, especially for the high error scenario. However, the range of differences in timing plans by absolute value is not as significant as in previous sections as the lower cycle time means that even at low volumes the effect of a few seconds of green are more pronounced.

3.3.1.6 *Impacts of Other Control Parameters (γ and β)*

The γ parameter controls the maximum step size and β is a smoothing factor that controls the linearity of the response. Changes to either affect the algorithm's ability to move to the optimal plan in rapidly changing situations. This also limits the effect of bad decisions. This effect is shown in Figure 3.8 for the high error ($\sigma(\epsilon) = 5s$) scenario.While selection of a low γ results in more stable performance, the setting can cause higher delays in rapidly changing scenarios or in situations where decisions are made at longer intervals.

Setting β to high values results in a muted response and the controller will only make changes when the current plan differs substantially from the optimal plan. The effect is illustrated in Figure 3.8c. The muted response reduces large erroneous steps from being taken, however, altering this parameter also means that the adjustments themselves are smaller steps toward the optimal plan. For example, the timing plan may only change when the difference between the optimal plan is greater than 5 seconds but the controller will only make a 1 second adjustment if that threshold is met. Higher β values therefore result in sub-optimal decisions when data is error-free, as can be seen by comparing the *optimal* decisions between Figures 3.8c and 3.8b. In contrast, when errors are introduced, simulation runs with higher β values better track the *optimal* decisions that more aggressive parameters would make in error-free configurations.

Setting $\beta = \gamma^2$ produces a near linear response for smaller changes but limits the situations where large adjustments are done. This was found to offer the best compromise between allowing the model to quickly re-correct bad decisions while also preventing steps from being too large in the presence of errors.

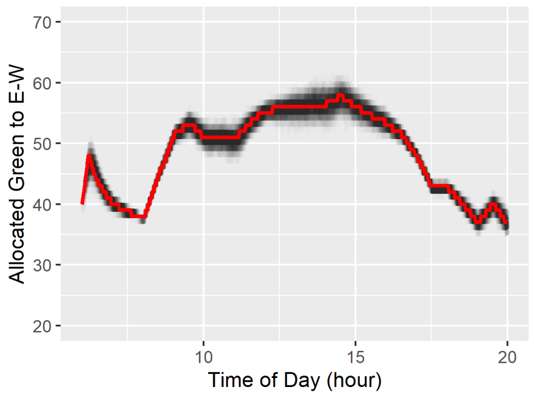
3.3.2 *Simulation Study*

Simulation was conducted in VISSIM for the control scenario first and it serves as a baseline to compare. Since all scenarios are simulated with the same seeds, the arrival time of vehicles between different simulation runs is identical. Comparisons are therefore provided on a relative basis compared to the control scenario for all the sensitivity analyses conducted.

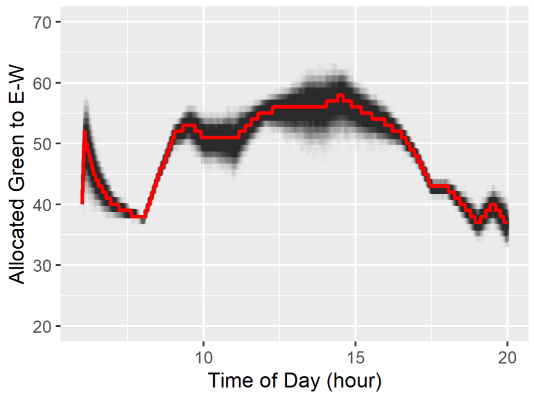
3.3.2.1 *Scenario Settings*

PTV Vissim was used to model a corridor on University Avenue in Waterloo, Ontario from the intersection of Westmount to King Street as shown in Figure 3.9. The modelled corridor uses volume data from 2018 obtained from the Region of Waterloo for a three hour peak period. Intersections #1 and #6 are major intersections while intersection #4 has significant northbound traffic during the peak hour. Aggregated volumes for all scenarios are shown in Figure 3.10.

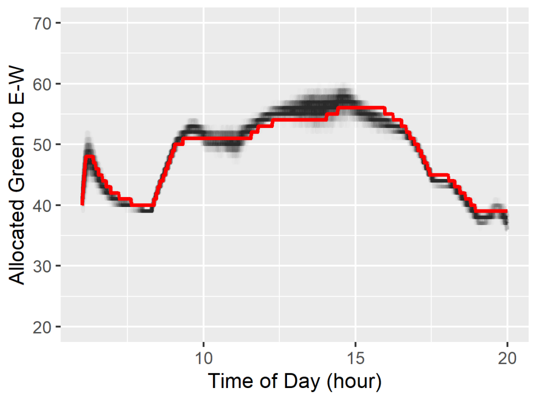
Simulated detectors are shown on Figure 3.9 and are placed at mid-block locations and cover all approaches. Detectors are all modelled with fixed detection ranges of



(a) $\gamma = 1\text{s}$



(b) $\gamma = 5\text{s}$



(c) $\gamma = 5\text{s}, \beta = 25$

Figure 3.8: Effect of γ and β in High Error Situations

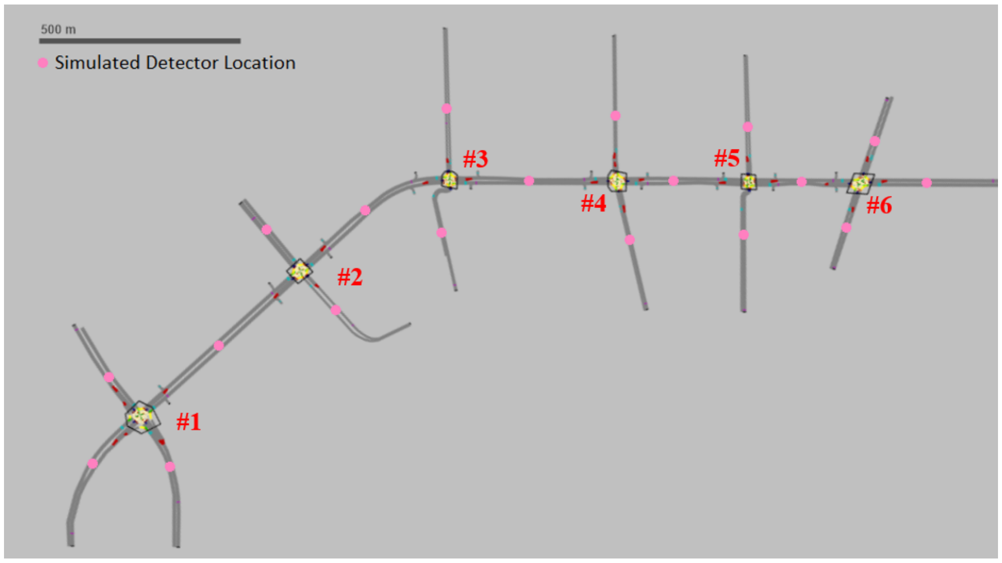


Figure 3.9: Microsimulation Network

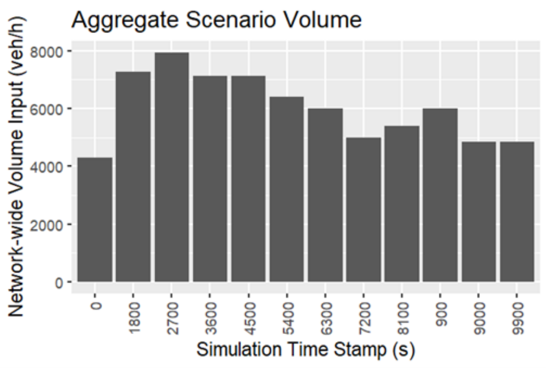


Figure 3.10: Aggregated volumes by time stamp

100m and poll vehicles every 2s. These values were selected after a review of the literature and from field data collected by a probe vehicle portable Bluetooth and WiFi scanners. For this scenario, γ and β are set to 5 and 25 respectively for both cycle, split, and offset adjustments. For simplicity, the progression factor $k_{f,j}$ in equation 2.14 was set to 1 for all intersections with no upstream intersection and 0.7 otherwise. Green splits were limited to a maximum 100s for major phases and 5 - 15s for protected left turn phases (depending on storage length) for intersections with protected lefts in the field plan. A minimum of 20s green for major movements and 0s of green for protected left movements was also enforced. Cycle lengths were limited to a maximum of 120s and a minimum of 60s.

All controllers implement a fixed signal timing schedule with adjustments made by the algorithm. A three hour period was simulated in VISSIM and repeated across 10 runs, with runs having different seeds. The same set of 10 runs was applied to test multiple scenarios including a control scenario with current field timings and actuated signal control, penetration rates varying from 5% to 100%, optimisation intervals of every 2, 3, or 4 cycles, and a 1.5x volume scenario. A high response scenario is also tested that sets γ and β to 10 and 100 respectively for all adjustments. During each optimisation interval, delay data is collected for the duration and used to estimate X and Y. Optimisation decisions are made at the end of the interval and take effect immediately. This means that shorter cycle lengths have more frequent adjustments and that longer cycles will have higher confidence in their delay estimates.

3.3.2.2 *Travel Time and Delay*

In the microsimulation approach individual Bluetooth and WiFi detectors are modelled directly. A platform was developed to interface with PTV Vissim using the Component Object Model (COM) programming interface. Detectors are defined with a detection type, measurement centroid, detection rage, and detection frequency. On entry, a random check is done on each vehicle to determine which detectors can detect them based on a predefined market penetration rate. At each detection interval, the ID of detectable vehicles within range of a detector are recorded. Key simplifications introduced by this approach is that vehicles are detected only once (in field settings vehicles may have more than one Bluetooth or WiFi device), non-vehicles such as pedestrians or cyclists are not modelled, detectors have static detection rates and ranges, and that detectors always detect vehicles that are in range of the detector if they are flagged as detectable.

As with real Bluetooth or WiFi detectors, however, only the travel-time is measured and the delay must be estimated from the collected data. Lloyd's implementation of the k-means approach, as outlined in Section 2.4.1.1 was used to group vehicles into *delayed* and *non-delayed* clusters. k was predefined to 2 and the algorithm was seeded using the fastest and slowest times. The differences between the average across each optimisation period is then used as the delay when estimating X using Equation3.6 and the process in Section 3.2.

3.3.2.3 *Rate and Optimisation Interval*

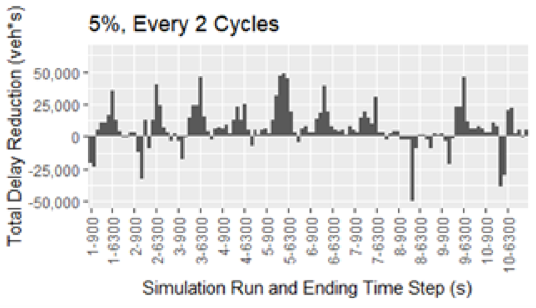
A total of 15 scenarios for combinations of penetration rate (5%, 10%, 25%, 50%, and 100%) and optimisation interval (2, 3, and 4 cycles) were considered. Table 3.1 highlights the results for all runs relative to the control scenario. Scenarios are listed by their penetration rate and optimisation interval. The control scenario is an actuated plan developed from the same field volumes. The adaptive algorithm is seeded with a plan that provides equal green to all approaches and a cycle of 90 seconds.

Better results were generally observed for more frequent updates, and in most scenarios the results from updates every 2 cycles produced the best improvements. This is likely due to the fact that the algorithm can correct any poor decisions made sooner and is better able to adapt to the sudden spikes in volume that occurs after the first hour. Since the algorithm was seeded with a sub-optimal plan, the longer update intervals had particularly poor performance in the first run. Penetration rate was not

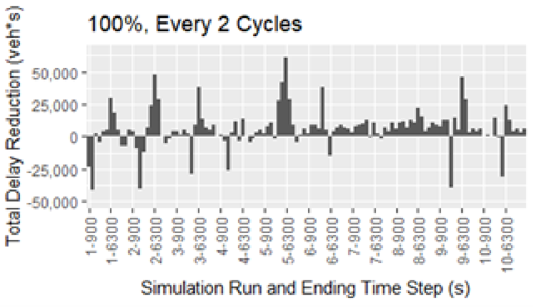
Table 3.1: Average Delay Reductions, listed by (Penetration Rate, Optimisation Interval)

Scenario	**1**	**2**	**3**	**4**	**5**	**6**	**7**	8	**9**	**10**	**Avg**
(100, 2)	−2	7	9	0	25	14	11	20	15	6	11
(100, 3)	15	14	17	18	19	11	8	−3	6	14	12
(100, 4)	−41	10	14	4	12	−3	3	7	−3	7	1
(50, 2)	−4	−9	22	11	22	8	0	4	22	7	9
(50, 3)	17	28	22	19	23	19	7	3	13	9	16
(50, 4)	−37	−16	4	22	20	−1	−14	3	7	19	1
(25, 2)	5	20	12	12	24	6	4	3	2	14	10
(25, 3)	3	−3	2	3	8	14	−3	3	1	5	3
(25, 4)	−40	−5	1	−8	10	−13	14	5	2	2	−3
(10, 2)	3	−4	35	23	39	27	30	24	19	24	22
(10, 3)	12	8	18	7	23	21	15	8	10	−3	12
(10, 4)	−22	−17	15	−2	34	11	6	16	23	15	9
(5, 2)	8	7	16	17	30	20	16	−10	15	2	12
(5, 3)	−25	7	2	−10	26	26	7	9	7	−7	4
(5, 4)	−11	13	3	−9	6	7	2	−21	21	20	4

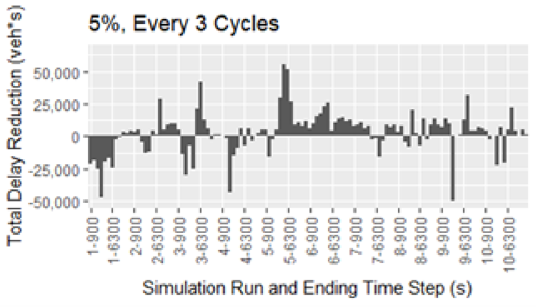
found to have a strong impact on the performance, however more stable performance was observed for the higher penetration rates. This is more clearly visible in Figure 3.11 which shows the performance in 15 minute intervals. The difference is particu-



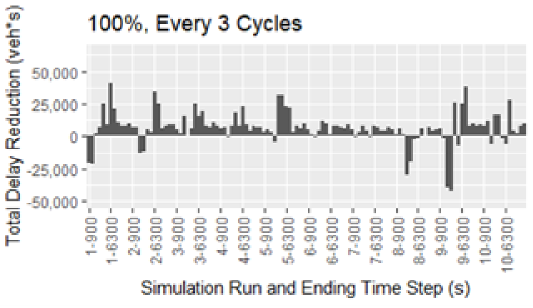
(a) 5% Penetration, Every 2 Cycles



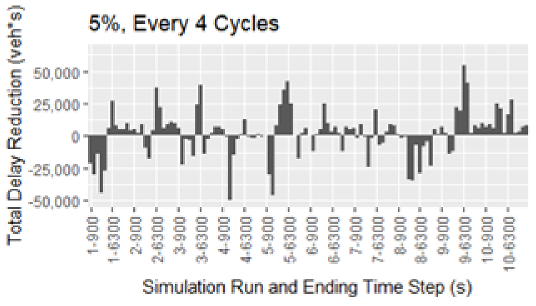
(b) 100% Penetration, Every 2 Cycles



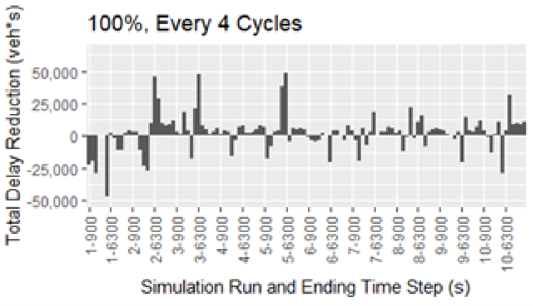
(c) 5% Penetration, Every 3 Cycles



(d) 100% Penetration, Every 3 Cycles



(e) 5% Penetration, Every 4 Cycles



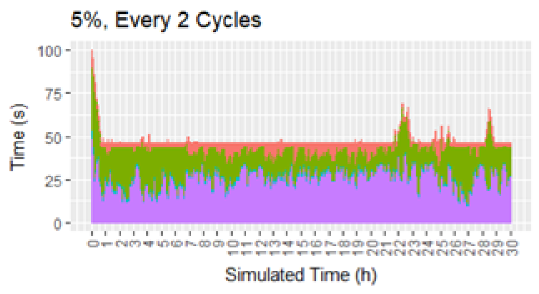
(f) 100% Penetration, Every 4 Cycles

Figure 3.11: Delay Reduction per 15 minute Compared to Control

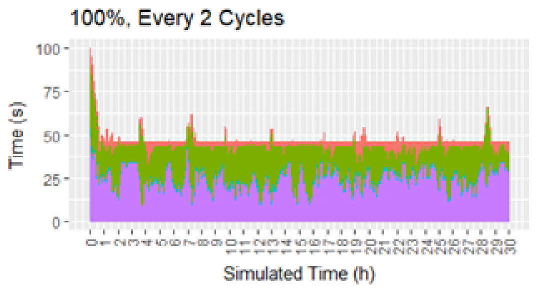
larly visible during the first run where the controller is better able to reach an optimal plan from the seed plan. Since the control scenario is timed to peak flows (between 1800 seconds and 2700 seconds on each run) the algorithm's performance is better during other periods, particularly between 5400 seconds and 7200 seconds when volume patterns shift more substantially away from peak flows.

3.3.2.4 *Cycle and Green Splits*

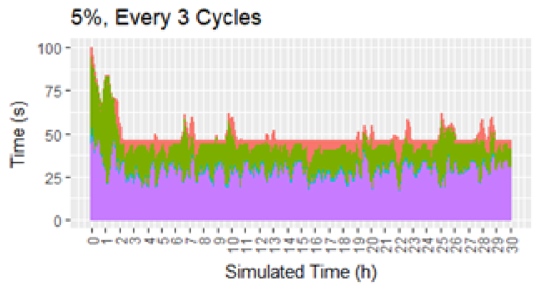
The signal timing decisions made by the algorithm were also examined for consistency. Figure 3.12 plots the decisions made by the controller at intersection #1 for the 5% and 100% penetration scenarios. It should be noted that these figures plot the planned time, but the actual time may differ as the signal may extend the green to accommodate changes in the offset.



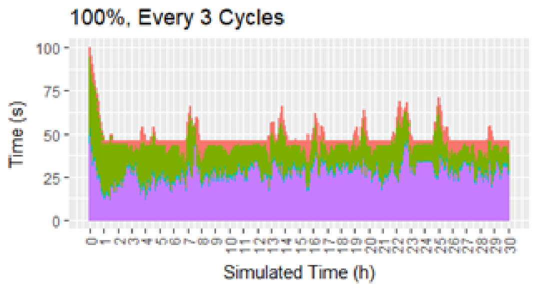
(a) 5% Penetration, Every 2 Cycles



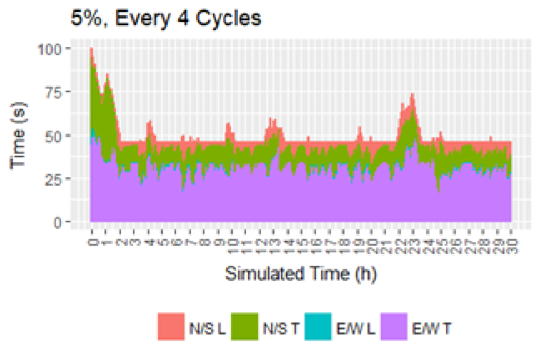
(b) 100% Penetration, Every 2 Cycles



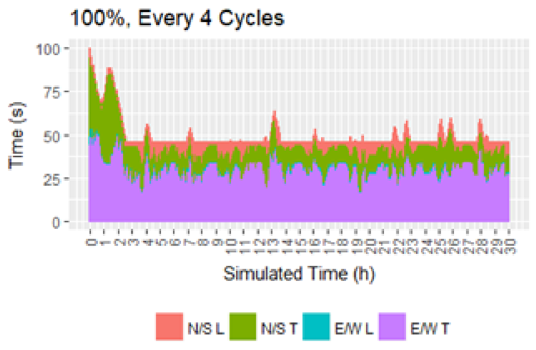
(c) 5% Penetration, Every 3 Cycles



(d) 100% Penetration, Every 3 Cycles



(e) 5% Penetration, Every 4 Cycles



(f) 100% Penetration, Every 4 Cycles

Figure 3.12: Split Allocations and Cycle Lengths (per Cycle)

At this intersection, the field plan provides a maximum of 15s of green to the protected left turns, 45s to the East/West movements and 35s to the North/South movements. Protected left phases are actuated and serve a minimum of 5s of protected green when called. In Figure 3.12 it can be seen that higher optimisation intervals provide split ratios that adequately serve the demand most of the time, and the timing plan is more stable. However, more frequent optimisations produced signal timings that better track the cyclic changes in volume at the intersection. The controller has a tendency to lower the cycle length and at this intersection much of the delay is caused by the N/S left turn movement.

3.3.2.5 *Offsets*

Offset adjustments made by the algorithm in the above 15 scenarios were tracked and recorded. These decisions are summarised in Figure 3.13 for the 5% and 100% scenarios and are presented as a heatmap. In this heatmap no-change is marked as orange while red areas denote decreases of the offset and green areas increases. As can be seen from the figure, the 5% penetration scenario generally had more changes made to the offset but the overall pattern was similar regardless of the penetration rate. Intersections #4, #5, and #6 generally saw more increases to the offset while intersections #1, #2, and #3 generally had more stable offsets.

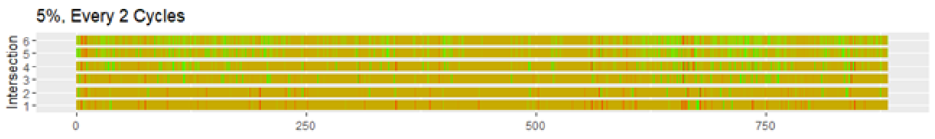
3.3.2.6 *Additional Demand*

This scenario is designed to demonstrate the potential benefits of the proposed method to allow signals to adapt as volumes change. Volumes are increased on all approaches linearly by 1.25x and comparisons are re-run for the control scenario without changing the signal timing plan and with a new signal timing plan. The algorithm's performance with updates every 3 cycles is compared to both control scenarios. The results of this comparison are shown in Table 3.2.

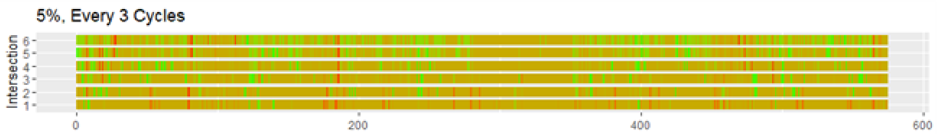
Table 3.2: Performance improvement of the Re-timing Method Compared to Actuated Controller

Penetration Rate	**Retimed Signals (%)**	**Base Timings (%)**
100%	10.9	18.7
50%	10.2	18.2
25%	8.8	16.9
10%	5.2	13.8
5%	5.1	10.1

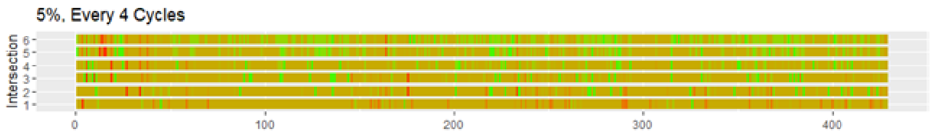
When compared to the base scenario's timings, the algorithm reduces delays by 10% to 18%, depending on the penetration rate. When compared to re-timed signals, this improvement is reduced to 5% to 11%. These results show that under changing volume conditions, the proposed algorithm can provide performance that is close to or better than actuated controllers timed with the volume data.



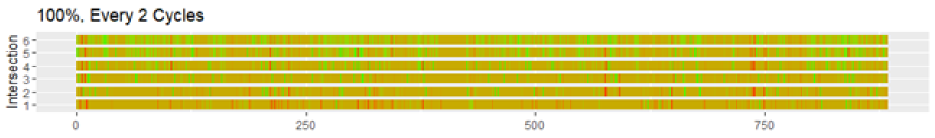
(a) 5% Penetration, Every 2 Cycles



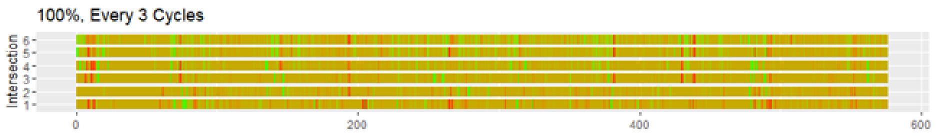
(b) 5% Penetration, Every 3 Cycles



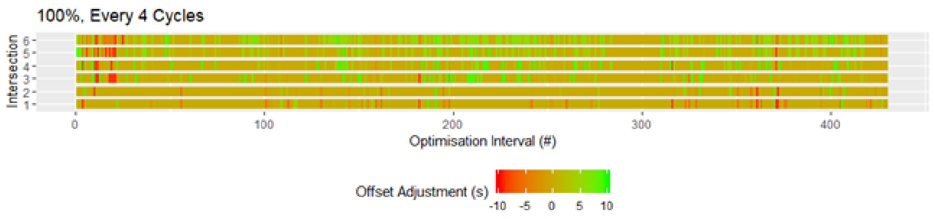
(c) 5% Penetration, Every 4 Cycles



(d) 100% Penetration, Every 2 Cycles



(e) 100% Penetration, Every 3 Cycles



(f) 100% Penetration, Every 4 Cycles

Figure 3.13: Offset Changes per Cycle

3.4 CONCLUSIONS

This chapter investigates the potential of using travel time data to improve the signal timings of signalised intersections. A model-based fine-tuning method is proposed to re-time signals solely using travel-time data. This method could be implemented in both on-line or off-line configurations at locations without vehicle detection or as an alternative to install costly vehicle detectors. Although the algorithm is evaluated using data from WiFi or Bluetooth detectors, any technology that provides travel time estimates could be suitable.

The system's reliability depends on a number of key factors. In particular, the sensitivity analysis and simulation study highlighted that under some circumstances timing plans can be unstable. In the simulation study, it was observed that the algorithm struggled to reliably re-time low-volume approaches, such as non-busy periods for protected left turns. The results also showed that while the proposed system has some elements of adaptive control, the decisions lag behind optimal ones, especially in situations with measurement error.

Adjustments of the timing plan every other cycle help mitigate these impacts and generally saw higher performance in the microsimulation study, however frequent adjustments also lower the accuracy of measurements, particularly for low volume approaches. Penetration rates were found to have limited impact, though generally lower penetration rates produced less stable timing plans.

Under changing volume conditions, the algorithm proposed provides an alternative method to constant signal re-timing through the use of travel time data. When volumes were increased the algorithm reduced delays by up to 18% when compared to unmodified actuated signal timings and by 11% when compared to re-timed actuated signals.

4 DEEP REINFORCEMENT LEARNING: SIGNAL CONTROL OF ISOLATED INTERSECTIONS

One of the key gaps in the state-of-the-art identified in Chapters 1 and 2 is the inability of traditional systems to capitalise on advances brought by new data sources and Artificial Intelligence (AI). Further, while traditional systems can work well when optimised for specific conditions, it is difficult and costly to keep signal timings up-to-date. Bluetooth and WiFi-based travel time data were introduced in Chapter 3 as one possible way to solve these problems, but technological developments also allow other sources of data to be used. In particular, advancements in video-based detection, LIDAR, infra-red, and other similar systems have brought the possibility for high-resolution data to be available in real-time.

The scale of data available also opens the door to more complex ways of optimising traffic signals. Recent advancements in AI has demonstrated algorithms that are capable of human-like decision-making, such as AlphaGO that used Deep Reinforcement Learning (DRL) to beat the world's best players of the complex strategy game of Go (Mnih et al., 2015). Within the literature, AI techniques have seen some interest in recent years, but systematic studies exploring optimal settings and design are limited. Successful application of DRL to the traffic domain requires specification of a number of key parameters, and configuration of them. These include the state space, action space, model structure, and rewarding system.

This chapter outlines a comprehensive study to evaluate and design an optimal DRL signal controller that functions on isolated intersections. A platform was developed and the sensitivity of key parameters was explored. The results of this analysis form the basis of the corridor or area-wide signal control study conducted in the subsequent chapter. In-depth details on the developed platform, including pseudo-code representations are provided in Appendix A in Section A.2.

4.1 SYSTEM SETTINGS AND PROBLEM DESCRIPTION

Consider the problem of traffic signal control of a typical intersection whereby the signal controller needs to decide at what time it should change the signal and give green to a different group of movements. The underlying problem can be framed as a Markov Decision Process (MDP) in a rolling-horizon form with discrete time decision points, t (now), $t+1$, $t+2$, ... In an MDP, the system state at the current time t, as described by a state space S_t, can be altered by an control agent's decision or action, denoted by A_t and subsequently transit to another state in the next time interval $t+1$ (S_{t+1}). The transition from a particular state S_t to the next state S_{t+1} can be described by a transition probability denoted by $P(S_{t+1}|S_t, A_t)$. An immediate reward

$R_t = R(S_t; A_t; S_{t+1})$ can be defined based on how the system transitioned from state S_t to state S_{t+1} after action A_t is taken. These three components form the basis of all MDPs. It should be noted that, in many real-world problems, it may not be possible to fully observe the state at a particular time, and often only part of the state is visible to the decision-maker. These problems are typically called Partially Observable Markov Decision Process (POMDP) and in these problems the decision-maker makes some observation O_t based on S_t and then chooses its action based on that observation. The relationship between all these variables is illustrated in Figure 4.1.

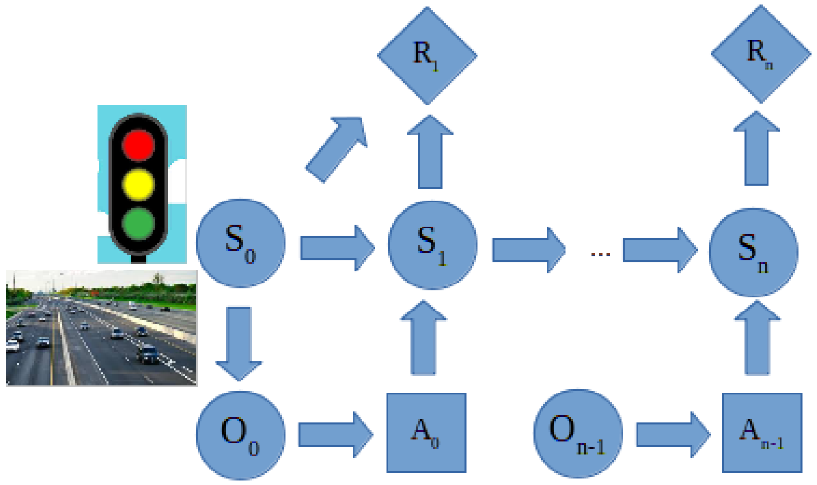


Figure 4.1: The Relationship Between States, Observations, and Rewards in an MDP

Under this framework, the traffic signal controller acts as an agent that continuously makes decisions (or take actions) at each second on whether to advance the signal to the next phase or to keep it at the current phase. It is assumed that the agent would be constantly making observations about the system state which could be traffic volume, queue length or movements of individual vehicles depending on the traffic detectors available to the agent. The agent would use the observations to assess the immediate and future rewards of alternative actions and decide at each decision point the best action to take. This is further illustrated below in Figure 4.2. The environment considered is an isolated intersection, including the time, place, lane structure, and positions of vehicles. It is assumed that the signal controller acts as an agent in the environment and can observe the time, queue lengths, discharges from the stop bar, and phasing of the signals. The fundamental elements that must then be determined before this framing can be used includes: the exact representation for the state and observations to give the model; the exact actions the agent can take and the expected consequences of those actions, and the rewarding method to use; which are the key issues being addressed in the proposed methodology described in the following sections.

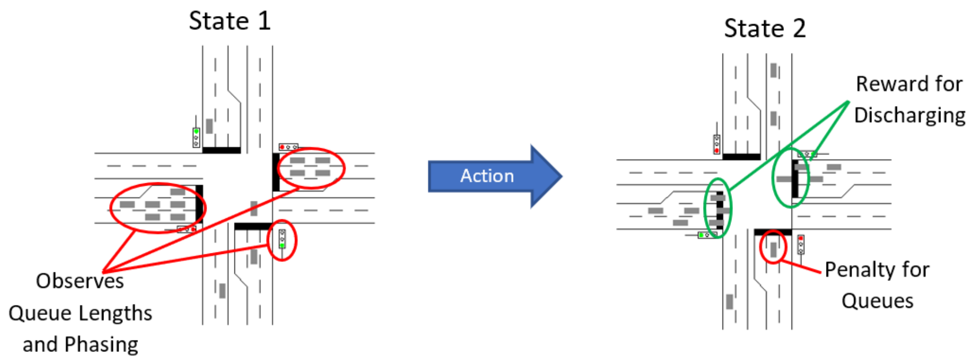


Figure 4.2: Framing the Traffic Problem as a POMDP

4.2 METHODOLOGY

As discussed in Chapter 2, one of the most successful solution methods to MDP is Reinforcement Learning (RL). In RL, the agent uses the observations, states, actions, and rewards to learn an optimal policy π^* by which it can operate in the environment. One of the most common approaches to reinforcement learning is Q-learning, where the agent assigns and learns a value for each state based on the rewards it receives and the rewards it expects to receive in the future discounted by some factor γ. Deep Reinforcement Learning (DRL), specifically, Deep Q Networks (DQN), is an approach to reinforcement learning that relies on deep neural networks to learn the optimal policy, or Q-function. This research explores and formulates a method by which DQN can be used to control traffic signals.

Deep Q Networks (DQN) models have several components that govern its model structure. These include many different weights w that are organised into groups called layers. Input are provided in matrices with dimensions $M \times N$ and fed sequentially starting from the raw sensing of the environment and passing through each layer until reaching the output from which the actions are chosen. The output of each layer is calculated using many different approaches. In this research Convolutional Neural Networks (CNN) are used in many of the layers. These layers work by using a smaller batch of weights to progressively scan the input values and generate the output. The step size by which this scanning proceeds is determined by the stride of the layer, and the resulting output will be smaller for higher stride values as a distinct weight is not used for each individual input value. In addition to CNN layers, this research also uses fully-connected layers and max-pool layers. In fully-connected layers each input is mapped to an output through a weight, whereas in max-pool layers adjacent input values in the matrix are combined and the maximum value is used as input for the output. All layers in the model use the Rectified Linear Unit (ReLU) function to determine the output for a given input. More broadly, this function is called an Activation Function and controls how the weight is applied to the input to generate the output. The ReLU function outputs the product of the input and the weight if the result is positive, and zero otherwise.

In order to be used successfully, DQN models must first be trained to operate on the environment they are working in. Training adjusts the weight values based on the outcomes observed using a process called backpropagation. A number of different approaches are used to stabilise the training process and improve convergence. This research uses an approach called ϵ-greedy training which operates by having the agent choose actions randomly by some probability ϵ and exploiting its training by some probability $1 - \epsilon$. Training is divided into three phases, the first is observation where the agent fills its experience buffer by observing the environment, including states, transitions, actions, and rewards. The second is training where the agent anneals ϵ by reducing it from its initial value to some final value. The third is fine-tuning or application of the model. At some point during this stage, if the model is to be applied to operate on a different environment, the learning process is stopped and the agent's experience is transferred to the new problem. The agent is then fine-tuned further on the new problem, but is able to exploit the knowledge previously gained.

4.3 MODEL DESIGN

In this research, a modular API platform was first developed to facilitate communication with the deep learning agent and sensors providing information on the environment as well as the intersection controller. An extensible design framework was used to allow drop-in replacement of components such as the simulation model or sensor API, and to allow multiple intersections to be trained from a single simulation network in parallel. The only requirement for any replacement component is that its input and output conform to the API's specification. A high level overview of these components is shown in Figure 4.3. In this chapter only isolated intersections are evaluated, but the architecture was developed to enable multiple intersections to be linked together through the API layer. In isolated configurations, the architecture functions identically except there is only 1 model training thread.

To operate properly, the agent must be provided with some input at pre-defined intervals from the environment. The agent then passes its actions out through the API for the controller to act on. The following sections provide detailed discussions on the key components of this architecture.

4.3.1 *Deep Reinforcement Learning Model*

A deep reinforcement learning model was developed in Python using Tensorflow (Abadi et al., 2016). Tensorflow is developed by Google and is commonly used for the purposes of large-scale machine learning without requiring the development of complex neural networks from a syntactical point of view. A deep convolutional neural network was developed based on the model structure proposed by Mnih et al. (Mnih et al., 2015). This model structure showed strong performance in complex pattern recognition tasks when applied to reinforcement learning problems, including the ability to play computer games such as early Atari games and Pong, as well as

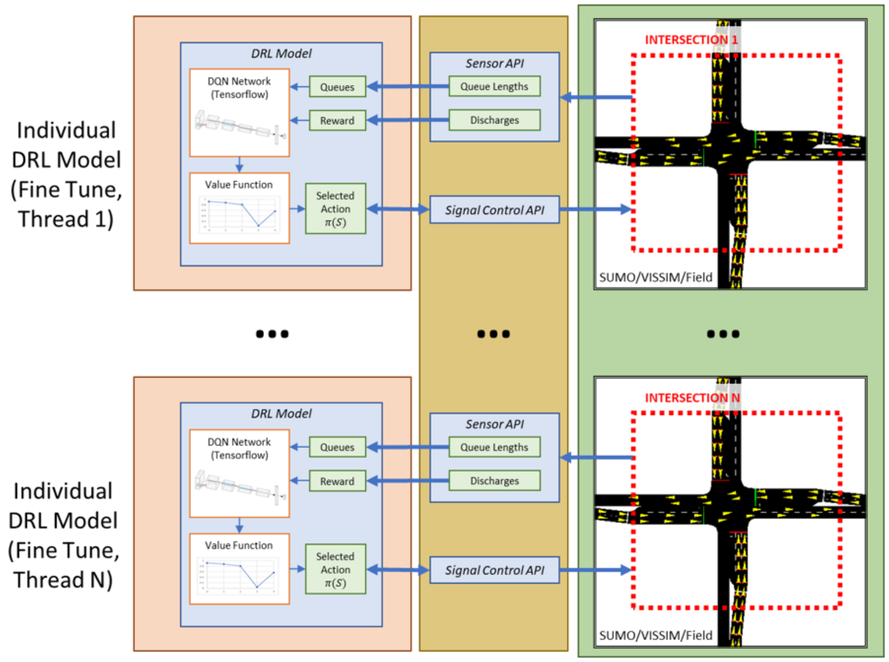


Figure 4.3: Overall Design Architecture, extensible intersection count from 1 to N

different applications demonstrated by other researchers including robotics (Tai and Liu, 2016; Rodriguez-Ramos et al., 2019), finance (Jiang, Xu, and Liang, 2017), and image recognition tasks (Rao, Lu, and Zhou, 2017).

The model structure adapted for this platform is shown in Figure 4.4. This structure accepts input as a matrix with dimensions $80 \times 80 \times 4$. Three convolutional layers are included as well as two max-pooling layers. The final output provides the estimated Q value of selecting each action given the particular input. Further details and pseudo-code implementations of this algorithm are provided in Appendix A in Section A.4.1. The following sections provide overviews of the critical parameters and inputs of the model.

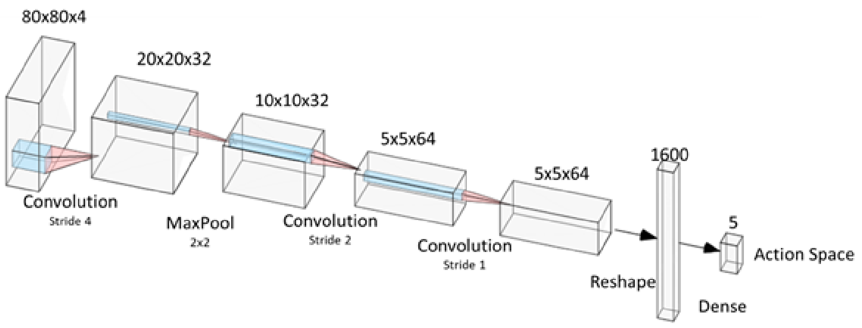
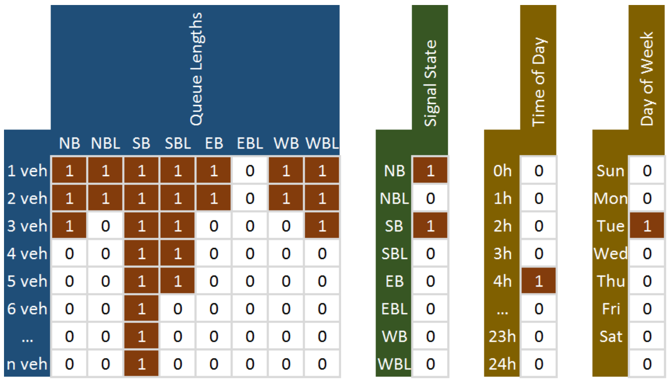


Figure 4.4: Deep Learning Model Structure



Queue Lengths

	NB	NBL	SB	SBL	EB	EBL	WB	WBL
1 veh	1	1	1	1	1	0	1	1
2 veh	1	1	1	1	1	0	1	1
3 veh	1	0	1	1	0	0	0	1
4 veh	0	0	1	1	0	0	0	0
5 veh	0	0	1	1	0	0	0	0
6 veh	0	0	1	0	0	0	0	0
...	0	0	1	0	0	0	0	0
n veh	0	0	1	0	0	0	0	0

	Signal State
NB	1
NBL	0
SB	1
SBL	0
EB	0
EBL	0
WB	0
WBL	0

	Time of Day
0h	0
1h	0
2h	0
3h	0
4h	1
...	0
23h	0
24h	0

	Day of Week
Sun	0
Mon	0
Tue	1
Wed	0
Thu	0
Fri	0
Sat	0
	0

Figure 4.5: State Space Observation Matrix

4.3.1.1 *State Space and Observation Matrix*

Representations of the states of the control environment are a critical component of any reinforcement learning problem. In traffic signal control contexts, this will depend on the information available. In this research it is assumed that real-time information on the number of vehicles waiting to be discharged in front of the stop line in each lane group of all approaches, i.e. queue length, is available and observable by the agent. There are many technologies in development that are able to provide some of these data, including some commercial products that automate traffic counts using video, Radar, or Lidar data. Loop detectors can also be used to provide some of these inputs. In addition to the availability of the queue length, it is also assumed that within its detection range the technology would provide accurate per-second queue counts. With this assumption, a framework for the observations made on the state space by the agent was developed to represent queue information in a vector format usable in the model. An overview of the state space observation matrix is shown in Figure 4.5. The model's input space is $84 \times 84 \times 4$ individual values, and so the matrix must be shaped to fit this size. Figure 4.5 shows the 2D abstraction used to represent the key data, including the queues for each movement, the signal state, and time of day information and has dimensions 84×84. In this figure, the abstraction highlights generally what data is included in the observation matrix, and the model is provided the current and previous 3 states of the matrix as input.

Each vector is a binary vector of N quantities that are 0 or 1 depending on the data. For queues, each queue vector q_i is composed of N sets of binary values $q_i = c_0, c_1 ... c_N$ which are 1 if there are at least i cars in the queue and 0 otherwise. N can be any arbitrary value, and each column is scaled such that all 84 values are used (even if N is less than or greater than 80). Multiple vectors are generated for each movement group. Any combination of lane groups can be included, but once trained the structure cannot be altered significantly without further fine-tuning. In this research separate vectors are coded for each of the cardinal directions' through and left turn movements. Similarly, signal timings are coded into eight possible groups, $P_i = p_1, p_2, ... p_8$ for protected-permissive operation corresponding to the respective lane groups. Time of day information is also encoded by hour and day of the week as two separate vectors $T_d = h_0, h_1, ...h_{23}$ and $T_w = d_1, d_2, ...d_7$ for Monday to Sunday. As columns are scaled to fit the height of the matrix, the width is also scaled by duplicating column values to increase their width. Each assembled matrix represents the state at a particular time slice, but the final input to the model also includes the matrices of the proceeding 3 states $(O_{t-1}, O_{t-2}, O_{t-3})$.

4.3.1.2 *Action Space*

In modern signals, timing plans are normally understood in terms of *ring-and-barrier* diagrams. In these diagrams, phases that govern what signal phase a particular lane group sees are organised into groups called *rings*. Rings in the diagram depict the phase as a sequence similar to a film-strip. Barriers are special points on the diagram where all rings must cross over and change their timing at the same time. Between barriers, phases in the same ring are *incompatible* and cannot time together, while phases in other rings are *compatible* with each other. This is illustrated in Figure 4.6 for a typical protected-permissive operation along with a proposed action scheme. Five actions are defined to allow the controller to follow basic ring-and-barrier structures while also implementing some features common to actuated signals. At every second, the agent chooses a legal action from each of the five available actions. Action 0 is the *do nothing* action and allows the controller to move to the next decision frame without changing the signal. Actions 1 and 2 advance phases in only rings 1 and 2, respectively.

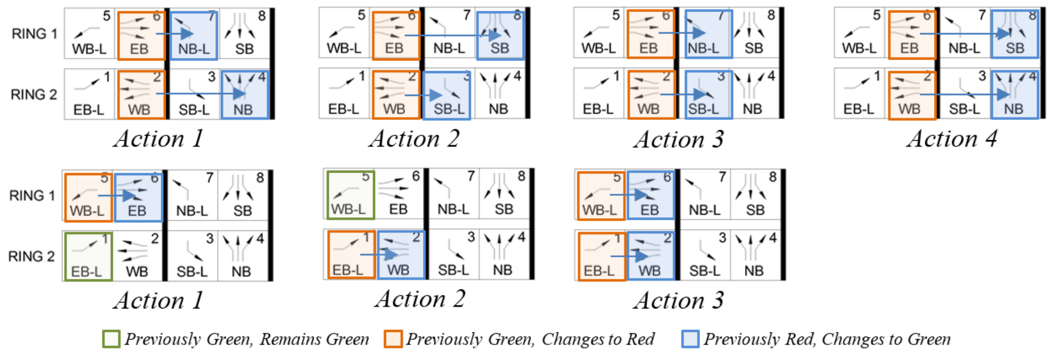

RING 1
RING 2
5 WB-L
6 EB
7 NB-L
8 SB
1 EB-L
2 WB
3 SB-L
4 NB
Action 1
Action 2
Action 3
Action 4
Previously Green, Remains Green
Previously Green, Changes to Red
Previously Red, Changes to Green

Figure 4.6: Actions Available to the Controller

When crossing a barrier, actions 1 and 2 also allow the agent to implement protected-permissive operation by advancing ring 1 or 2 to the left turn phase (e.g. $\phi1$ to $\phi3$ for action 1) and the other ring to the last compatible phase (e.g. $\phi6$ to $\phi8$ for action 1). When crossing a barrier, these actions are only permitted if both rings have served the main through phase (e.g. if $\phi5$ and $\phi2$ are timing, Action 1 is illegal). Action 3 advances both rings 1 and 2 to the next phase and, like actions 1 and 2, when crossing a barrier is only valid if all rings are timing the phase immediately prior. In protected-permissive settings, when crossing the barrier, action 3 will provide a protected left to both directions. Action 4 is only valid when both rings are timing the phase before the barrier (e.g. $\phi 2$ and $\phi 6$). When selected it causes the controller to skip to the phase just before the next barrier. In protected-permissive configurations this means advancing to the through movements for both directions.

Action filtering is employed to prevent the agent from selecting illegal actions. Where an action is illegal and has the highest Q value, the agent cannot select it and must instead select the action with the next highest Q. This design is suitable for most typical applications, but some complex situations may require additional considerations. However, adding additional rings would increase the number of actions significantly resulting in more state-action pairs for the model to learn.

4.3.2 *Reward Function*

Reward functions are the primary method by which a model developer guides the strategy learned by a reinforcement learning agent. The design of a rewarding function will influence how fast the agent learns and how effective the final policy is. For traffic signal control, several options are available depending on what goals are to be achieved and the design of the model. Since the input is framed in terms of queue lengths it naturally makes sense to consider a reward function that considers these inputs. Further, since decisions are made frequently, it is necessary to consider a metric that can give instantaneous feedback to the model.

A number of different rewarding structures based on the queueing mechanism were therefore evaluated and a sensitivity analysis was conducted. In the first, a reward of x for each vehicle discharged and a penalty of y for each queued vehicle were assigned. An additional penalty factor u can also be applied to penalise the agent for creating split failures. Different candidate values for x, y, and u are evaluated. Since rewards and penalties are applied every second, the long term effect of of the reward is similar to the total delay. This rewarding system requires that agent be able to observe the discharges from the stop bar for any time period where the model is being trained.

Reward clipping has been shown to have positive effects on learning by stabilising the performance (Mnih et al., 2013). Reward clipping modifies the original reward value by restricting its output to some specific range, commonly $(-1, 1)$. Clipping is done using a wide variety of approaches, including truncating rewards outside the range or scaling the rewards to fit within the range. In this work a scaling approach is evaluated where the rewards are clipped between $(-1, 1)$ by a sigmoid function shown in Equations 4.1 and 4.2 below. The function applied will diminish the penalty for excessive queues but threshold values can be configured.

$$R_c = \frac{\frac{R_t}{250}}{\sqrt{1 + (\frac{R_t}{250})^2}} \tag{4.1}$$

$$R_t = (x_t - y_t) \times u \tag{4.2}$$

Where:

R_c is the clipped reward
R_t is the reward from actions in time t given the state S_t, successor state S_{t+1}, and action A_t taken
x_t is the positive component of the reward calculation
y_t is the negative component of the reward calculation, or the penalty to be subtracted
u is a scalar parameter that can be configured

In addition to this approach, a second approach to the rewarding system was also evaluated. This approach attempts to minimise the longest queue at the intersection based on work proposed by Li et al (Li, Lv, and Wang, 2016). In this approach the agent is rewarded for each vehicle g_i behind a green light and penalised for each vehicle r_i behind a red light and rewards are clipped using Equation 4.1.

Finally, a third approach to rewarding was also evaluated. In this third approach the agent is only rewarded for discharging vehicles and not penalised for any other actions.

4.4 TRANSFER LEARNING

In Section 2.4.4 the idea of transfer learning is introduced. Transfer learning is a critical part of any machine learning model that would be applied to real-world intersections. In practical situations, it would be expected that a model would be first pre-trained in a simulation environment and then applied in the real-world. This could reveal a number of problems as the model's performance may be tied to the driver behaviours of the simulator, which could differ from local real-world driver behaviour.

A transfer learning approach is therefore evaluated by first pre-training the model on one simulator (e.g. SUMO) and then applying it on another (e.g. VISSIM) as a way to simulate the transfer between simulated and real-world environments. These simulators have different driver behaviour models, VISSIM's is based on the Wiedemann model outlined in Section 2.2.5.1 and SUMO is based on Krauß's model as outlined in Section 2.2.5.2.

4.5 SCENARIO SETTINGS

A simulation environment was developed in SUMO as shown in Figure 4.7. This environment is a single four-legged intersection with two lanes for the through and right movements as well as additional turning lanes for the left turn movement. In the default scenario, volumes are generated using a stochastic Poisson arrival process, with the average arrival rates given in Figure 4.8 and Table 4.1. This arrival rate starts with one direction at a low volume and the other direction at a high volume and steadily switches which direction has the higher flow.

In addition to the volume settings, the other settings listed in Table 4.1 were established as the reference scenario. Subsequent comparisons were made by modifying some of these settings to gauge their impact while holding others at their defaults.

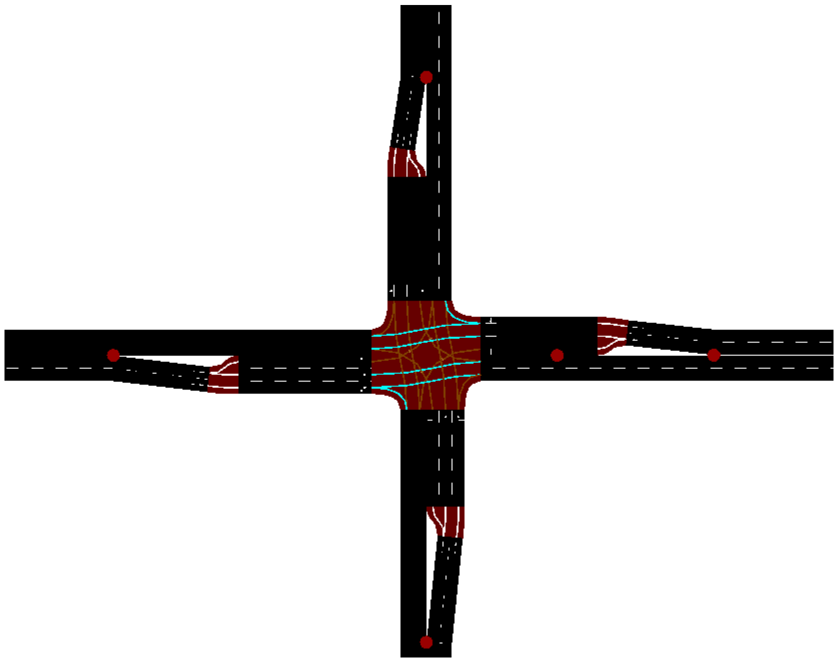


Figure 4.7: SUMO Network

For the initial sensitivity analysis, turning movements and protected left turn phasing is not considered to reduce the training time as the agent only has to choose between two actions, changing the signal or doing nothing. A test with the best values identified is then run with volumes that contain all turning movements and protected-permissive phasing. Based on the initial analysis work, the simplifications were found to not have a significant impact on the overall patterns of the sensitivity analysis beyond increasing the training time.

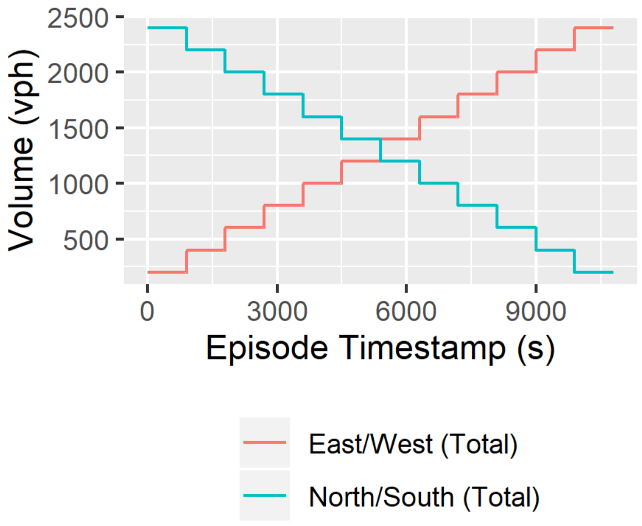


Figure 4.8: Default Volume Scenario

Table 4.1: Default Settings

Setting	**Value**	**Setting**	**Value**
Volumes		*Model*	
Northbound and	linear increase to	Initial Epsilon	1
Southbound	2400 every 900s	Final Epsilon	0.005
Eastbound	linear decrease to	Annealing Steps	50000
Westbound	0 every 900s	Observation Steps	10000
Turning Ratios (R,L,T)	0%,0%,100%	Learning Rate	10^{-6}
		Discount Rate γ	0.99
Simulator		Mini-batch Size	32
Speed Limit	60 kph	Detectable Queue	80
Driving Model	Krauss (Krauß, 1998)	Reward Function	$x = 5; y = 0.4$
Terminal State	10800 seconds or	-	$u = 10$ times
	total queue $\geqslant 150$	-	queue $\geqslant 2$

4.6 SENSITIVITY ANALYSIS

A sensitivity analysis was conducted on the isolated intersection by varying some of the critical parameters and retraining the model. The sensitivity analysis covers the detectable queue length, reward function parameters, other model settings, and transfer learning. A comparison is also provided to traditional actuated control with a ring-and-barrier controller.

Results in these sections are collected as averages of a complete scenario run (maximum 3 hours). Terminal states occur either when the full simulation has run *or* when the total queue length in the system exceeds 150 vehicles. Excessive queues represent unrealistic situations where the agent has lost control over the situation; therefore reducing their prevalence could lead to less experience required before convergence.

4.6.1 *Detectable Queue Length*

The model's input depends on the availability of queueing information, but in many practical situations this information may be limited in its availability, including limits on the resolution and range of detectable vehicles. A sensitivity analysis was conducted to evaluate the performance of the model as the detection limits ranged from 2 to 80 for Queue length. Queue detection is modelled through a per-lane system, and the Queue vectors are scaled to accommodate the tested limits. The reward function similarly is not scaled past the detection limit, though discharges are still recorded as usual. The results of this analysis are highlighted in Table 4.2 and Figure 4.9 for all models that converge to a stable policy that ceases to trigger terminal states.

Table 4.2: Comparative Results based on Limited Queue Length[a]

Detectable Queue	**Average Queue**	**Average Delay**	**Convergence Step[b] (steps)**
2[c]	-	-	
10	23.7	36.6	1030000
20	23.0	35.9	1020000
40	23.0	35.8	1300000
80[d]	23.0	35.8	1050000

[a] Results for 1 million time steps after initial convergence of the baseline model (1×10^6 to 2×10^6)

[b] Convergence Step defined as point where accumulated reward does not improve by more than 5% over the next 100 complete simulations.

[c] Does not converge to a stable policy.

[d] Baseline settings

The reward accumulation plot in Figure 4.9c shows the total rewards collected in an episode for each model as it gains experience. Total rewards will peak and plateau once the model's training has stabilised, and generally a model that collects higher rewards is better for the same reward function. However, in this test, since the reward function's penalty is affected by the queue detection, this comparison is limited. In most cases, rewards plateaued after around 1 million frames of experience.

When only 2 vehicles can be detected the model is not able to converge to a stable policy. With such limited resolution, queues are almost always detected at all approaches and the agent is not able to discern a proper strategy. Further, the agent sometimes ignores long queue lengths, especially when volumes are low on the other approach, in favour of collecting rewards from the main street. All other settings examined eventually converge to policies that produce stable timings.

Training beyond the onset of plateauing in the collected rewards resulted in a degradation of performance for all models, likely due to over-fitting. Average delays increased by at least 10% beyond 2 million frames for all models except for the case in which the detectable queue length was set to 10.

Although the model with the detectable queue length set to 10 did converge to a stable policy, its performance was noticeably worse than all other models. An Analysis of Variance (ANOVA) and Tukey analysis were conducted to compare the performance results of the models and determine which differences were statistically valid. For a period of 1 million steps after the convergence of the baseline model, only the model with detection less than 10 had a statistically significant difference. However, after 2 million frames, all models have similar performance with no statistically significant differences. However, this is likely due to over-fitting to this demand pattern.

4.6.2 *Effect of Reward Function Parameters*

Since the model's primary goal is to maximise rewards, the reward function's design is a critical component of the model's architecture. A variety of values were tried to evaluate the effect of the penalty and reward factors (x and y) and the scale factor (u) that define the final reward specified in Equation 4.1 and 4.2 have on the model's learning rate and final performance. A total of four settings for x, y, and u plus the baseline setting were examined as outlined in Table 4.3. Mathematically these can be represented using the following basic forms:

$$x_t = b_x \times Q(t) \tag{4.3}$$

$$y_t = b_y \times E(t) \tag{4.4}$$

Where:

$Q(t)$ is the queue length at the end of time interval t

b_x is a weight parameter applied to the queue length $Q(t)$ at time t

$E(t)$ is the count of discharged vehicles during time interval t

b_y is a weight parameter applied to the count of discharged vehicles $E(t)$ at time t

A further 3 settings were also examined for the alternative reward framing discussed in Section 4.3.2. These results are all presented in Table 4.3. The results are highlighted in Table 4.3 and Figure 4.10. Changes to the reward function resulted in substantial impacts to the convergence of the model. In particular, only the Baseline, R2 and R3 settings produced stable performance while the alternative reward framing was not able to converge to a stable policy at all. The baseline setting performed the best, acheiving a 60% higher level of performance than the next best performer - R3. The additional guidance provided by penalising the agent for split failures through the u parameter was found to be useful, as the performance of the baseline with $u = 10$ performed better than setting R2 which removed the effect of u. This parameter also has the effect of penalising the agent for rapidly changing the the traffic light, which is not optimal as the "do nothing" action often should be selected until queued vehicles discharge.

4.6.3 *Effect of Model Training Settings*

In ϵ-greedy training, the agent first decides whether to exploit its knowledge with probability $1 - \epsilon$ or explore with probability ϵ. The process of training aims to allow the model to act and obtain experience to refine its ability to identify the optimal action to take at any given time. Training is divided into three stages. In the first stage the agent *observes* the environment and ϵ is not annealed and held at its initial value. This stage allows the agent to collect experiences to fill its experience buffer, and the target network is not updated during this stage. Longer observation periods will give more varied experiences to sample from the initial training, but the time spent observing may be wasted if no valuable information is obtained or could be

Table 4.3: Comparative Results based on Rewards[a]

Name	Setting	Avg. Queue (Vehicles)	Avg. Delay (Seconds)	Conv. Step[b] (steps)
Baseline[c]	$b_x = 5, b_y = 0.4, u = 10 \times Q(t) \geqslant 2$	23.0	35.8	1050000
R1[d]	$b_x = 1, b_y = 0.4, u = 10 \times Q(t) \geqslant 2$	-	-	-
R2	$b_x = 5, b_y = 0.4, u = 1$	35.4	52.2	1100000
R3	$b_x = 10, b_y = 0.4, u = 10 \times Q(t) \geqslant 2$	30.2	43.5	1610000
	x_t = Q(t) behind green lights;	-	-	-
R4[d]	y_t = Q(t) behind red lights	-	-	-
	x_t = Q(t) behind green lights;	-	-	-
R5[d]	$y_t = 2 \times$ Q(t) behind red lights	-	-	-
	$x_t = 2 \times$ Q(t) behind green lights;	-	-	-
R6[d]	y_t = Q(t) behind red lights	-	-	-

[a] Results for 1 million time steps after initial convergence of the baseline model (1×10^6 to 2×10^6)
[b] Convergence Step defined as point where accumulated reward does not improve by more than 5% over the next 100 complete simulations.
[c] Baseline settings
[d] Does not converge to a stable policy.

unhelpful if the situations observed are not realistic. Following this stage, the value of ϵ is *annealed* by reducing it from an initial value to some smaller final value. This annealing occurs over the length of time assigned to this stage. During this period the agent changes from primarily exploring the relationship between states and actions to exploiting the knowledge it has developed. When ϵ is fully annealed, the agent may continue training but will generally exploit its knowledge and greedily select the best action.

When exploring by selecting a random action, the agent will weight each action equally and select one. However, the problem of traffic control in per-second continuous decision-making means that the agent will typically expect to select the *do nothing* action until a suitable time and then select one of the other actions. If each action is weighted equally, then the agent's exploration horizon will be limited to the first few frames after the signal has timed with the minimum green. To address this limitation, an alternative training method was considered where if the agent randomly decides to choose the *do nothing* action, the action is repeated for N times, with N is set to $30 \times h \times \epsilon$, where h is a random value between 0 and 1 and ϵ is the current value of ϵ during the training. N therefore decreases from a maximum value of 30 seconds as ϵ is annealed.

In this section the effect of lengthening or shortening the training periods is assessed for its impacts on training speed. This includes the length of the observation time, annealing time, initial ϵ value, learning rate, and discount factor γ. The learning rate controls how fast the weights of the model are updated and higher values can lead to faster training times. The tested models are all trained using the Adam Optimiser, which is a stochastic gradient descent algorithm (Kingma and Ba, 2014).

The results are shown in Table 4.4 and Figure 4.11. Each of the settings test different lengths of the training time, with M1 to M3 using shorter or longer observation lengths. Settings M4 to M6 change the length of the annealing period. Settings M7 to M10 test different learning rates and discount factors. The learning rates and discount factors tested are lower and higher than the default baseline. Finally setting M11 evaluates the performance of the alternative training strategy.

Table 4.4: Comparative Results based on Model Settings[a]

Name	Setting	Value	Avg. Queue	Avg. Delay	Convergence Step[b] (steps)
Baseline[c]	Baseline	-	23.0	35.8	1050000
M1	Observation Length	2000	23.1	36.0	1570000
M2	Observation Length	5000	24.9	37.4	1470000
M3	Observation Length	30000	32.5	37.8	1760000
M4	Annealing Length	20000	25.6	36.1	990000
M5	Annealing Length	40000	24.9	37.4	1450000
M6	Annealing Length	70000	37.5	29.8	2880000
M7[d]	Learning Rate	10^{-5}	-	-	-
M8[d]	Learning Rate	10^{-7}	-	-	-
M9	Discount Factor	0.75	23.0	35.3	780000
M10	Discount Factor	0.50	25.2	37.2	800000
M11	Random Green	$30s \times h \times \epsilon$	23.9	36.4	770000

[a] Results for 1 million time steps after initial convergence of the baseline model (1×10^6 to 2×10^6)

[b] Convergence Step defined as point where accumulated reward does not improve by more than 5% over the next 100 complete simulations.

[c] Baseline settings are 10000 and 50000 steps of observation and annealing respectively, discount of 0.99, and learning rate of 10^{-6}

[d] Does not converge to a stable policy.

In terms of performance, settings M3 and M6 highlight that longer observation and annealing intervals did not translate into faster convergence. While the final difference in performance of the models was not statistically significant from the baseline after convergence, convergence occurred faster for shorter observation and annealing intervals. Overall, the baseline provided the best performance, however setting M4 had faster convergence.

The benefit of shorter annealing or observation times could be caused by the tendency of exploration to lead to unrealistic traffic jams. This is strengthened by the fact that setting M11 that tested the alternative training strategy had the fastest convergence of all settings with convergence observed in 770,000 simulation steps compared to 1 million for the baseline setting.

Learning rates other than 10^{-6} did not converge onto optimal policies. This result was unexpected as other applications with similar Deep Neural Networks (DNN) designs have achieved success with such learning rates on different problems (e.g. (Krizhevsky, Sutskever, and Hinton, 2012; Zhu et al., 2017), however this learning rate is still normal but suggests that exploration in the problem space does not reliably produce experiences that would allow faster training. This implementation did not examine the effect of variable learning rates, and annealing the learning rate from a higher value to a lower one may also provide benefits over static learning rates (Chollet, 2017).

The discount factor controls how an agent prioritises immediate rewards and future rewards. High values of γ means the agent's time horizon is longer, while smaller values lead to prioritisation of immediate rewards. Some researchers have used values as low as 0.1 (Li, Lv, and Wang, 2016). Depending on the reward function and problem, high discount values can lead to extremely large Q values as the agent's time horizon approaches infinity and long time horizons can obscure the effect of immediate choices, making it more difficult to learn an optimal policy. Values of 0.5 and 0.75 were compared to the baseline of 0.99. When γ is set to 0.5 the effect of rewards beyond the next 5 steps is greatly diminished. Similarly, for 0.75 the effect of rewards beyond the next 15 steps is negligible, while for 0.99 rewards within the next 200 steps still contribute significantly. While both discount factors provided faster convergence than the baseline condition, the initial performance at convergence for the 0.5 setting was generally worse than the baseline condition. In general, setting the discount factor to a lower value of 0.75 encouraged faster training and provided the same or better performance when compared to the baseline.

Regardless of the settings chosen, all the models eventually converge onto a policy that provides a similar level of performance, and the primary impact observed was on training speed. For more complex problems, training speed becomes an important variable as complex action spaces can hinder the ability of the agent to explore effectively.

4.6.4 *Effect of Time of Day*

The baseline observation from the observation matrix includes vectors to encode the time of day T_h and day of week T_w. A sensitivity analysis was conducted by training the model with, without, and with only time of day information. The test scenarios are summarised in Table 4.5 and Figure 4.12. An alternative volume scenario was also tested with settings S3 and S4. In these settings the volume is adjusted to more clearly follow hourly lines. The NB/SB approaches have volumes of 1800vph and the

EB/WB approaches have volumes of 800vph. During the second hour, all approaches have volumes set to 1300vph.

Table 4.5: Comparative Results based on Observation Space Configuration. Results for time period between 3×10^6 and 4×10^6 timesteps

Name	Space	Queue	& Delay	Conv. Step (steps)
Baseline	$O_t = \{q_i, P, T_h, T_d\}$	24.0	36.7	1050000
S1	$O_t = \{q_i, P\}$	24.0	36.4	1050000
S2[a]	$O_t = \{P, T_h, T_d\}$	-	-	-
S3[b]	$O_t = \{q_i, P, T_h, T_d\}$	24.1	35.5	1480000
S4[b]	$O_t = \{q_i, P\}$	23.7	35.0	2450000

[a] Does not converge to a stable policy
[b] Settings with alternate volumes

The results showed that while the time of day information reduces the training time in some cases, the results are mixed. Setting S1 had similar performance to the base scenario and similar convergence speed. While setting S3 initially converged faster in terms of the rewards collected, setting S4 still had better initial performance, and ultimately the performance difference between the two models was not significant. Setting S2 was not able to converge to a stable policy when only time of day information was provided. Since decisions are made continuously on a per-second horizon and no green information was provided the agent cannot predict when to change the signal.

4.7 MODEL TRANSFERABILITY

The transferability of trained models is a major gap in the literature but is an important question when studying how such models could be applied in field settings with conditions that are different from those assumed when the models were trained. In practical situations, it would not be possible to train a model by allowing it to randomly select actions and navigate through traffic congestion. Although models could be pre-trained in simulation environments, the behaviours the model learns will be different and some adjustments would likely be needed. A study was designed to evaluate how difficult fine-tuning would be if the underlying driver behaviour was modified. The transferability of the trained DRL model is evaluated by training the model using data from one simulator (e.g. SUMO) and testing its performance in another (e.g. VISSIM). A model was pre-trained in SUMO for 1 million frames and then applied on a VISSIM network that replicated the layout and volumes of the training intersection, as shown in Figure 4.13. This training time was identified as the convergence time and stopping the pre-training here avoids over-fitting the model to the SUMO network. VISSIM's driver behaviour model is a psycho-physical model that has different driver behaviour regimes and is used in many professional applications.

SUMO's driver behaviour is based on gap distances and collision avoidance, and the model contains a number of changes to prioritise calculation speed that allows it to simulate large scale networks (Krauß, 1998).

Without calibration, both SUMO and VISSIM have noticeable performance differences under the same traffic conditions. Visual differences can be observed in the discharge rate of vehicles and there are noticeable congestion differences for the same actuated timing plan between both simulators. Pre-training a model on SUMO and then applying it on VISSIM is therefore similar in some regards to pre-training the model in simulation and then applying it to a field setting. During fine tuning, the model's ϵ value is not reset and training continues with the agent usually exploiting its pre-learned knowledge.

The results of the fine-tuning process are shown in Figure 4.14. The model's initial performance is not optimal and visually the agent initially repeatedly switches between phases too early resulting in congestion. However, convergence is much faster and after 75,000 simulation time-steps the model re-converges on an optimal policy, though after about 10,000 steps the models performance becomes acceptable and it no longer triggers terminal states due to congestion. This result suggests that some fine-tuning would be required in the field, but that much of the knowledge learned in the simulation environment could be transferred. In this instance, transferring the trained network to VISSIM required approximately 1 additional day of fine-tuning, though acceptable performance was observed sooner.

4.8 COMPARISON TO TRADITIONAL SIGNAL CONTROL AND TIMING METHODS

The baseline model's performance was compared to a traditional actuated control. Actuation is enabled through SUMO's ability to simulate loop detectors and three timing plans for each of the hours simulated were used. These plans were developed in SYNCHRO with average volumes for each of the hours. The same plan was run both in a fully-actuated configuration and in fixed time configuration. The following results in Table 4.6 highlight the comparison between the traditional approaches and the baseline model.

Table 4.6: Comparative Results to Traditional Control Methods

Name	**Average Queue**	**Average Delay**
Baseline[a,b]	23.0	35.8
Fully Actuated	25.1	38.9
Fixed Time	25.4	39.18

[a] Results for 1 million time steps after initial convergence of the baseline model (1×10^6 to 2×10^6)

[b] Baseline settings

When compared to traditional control methods, the baseline model reduces delays by 10% when compared to the fixed time controller and 6% when compared to actuated signals. Importantly, however, the baseline model is not seeded with any knowledge of the demand at the intersection.

4.9 FULL RING AND BARRIER CONTROLLER

A ring-and-barrier style controller was then evaluated using the full action space specified in Section 4.3.1.2. The scenario modelled in the previous case studies shown in Section 4.5 is adapted by modifying the base scenario to have turning volumes of 10% left, 10% right, and 80% through. Apart from the turning movements, all settings and configurations remain the same for the simulation and training process as outlined previously.

Delay results for the full actuated controller are shown in Figure 4.15. Accommodating turning traffic requires longer cycle lengths, especially in the middle hour where both directions have high volumes. This causes higher delays when compared to the two-phase case. The training time and results for the baseline scenario is also plotted on this figure. When compared to the baseline scenario, training times are also nearly four times as long and the model does not converge until over 2 million frames of experience have been recorded. Performance during the learning process is also initially very poor as the agent learns to balance when protected left turn phases should be called.

4.10 CONCLUSIONS

This chapter investigates the ability of DRL to operate and optimise the control at an isolated intersection. A platform was developed including an Application Programming Interface (API) interface by which different components can be removed and replaced, such as the learning model and simulator. This creates a de-coupled system where, in the future, it may be possible to conduct further research on this platform using field studies or hardware-in-the-loop simulation. A complete structure necessary for a DRL model to operate was then specified, including a state space and observation definition, action pattern, reward functions, and parameters of the model.

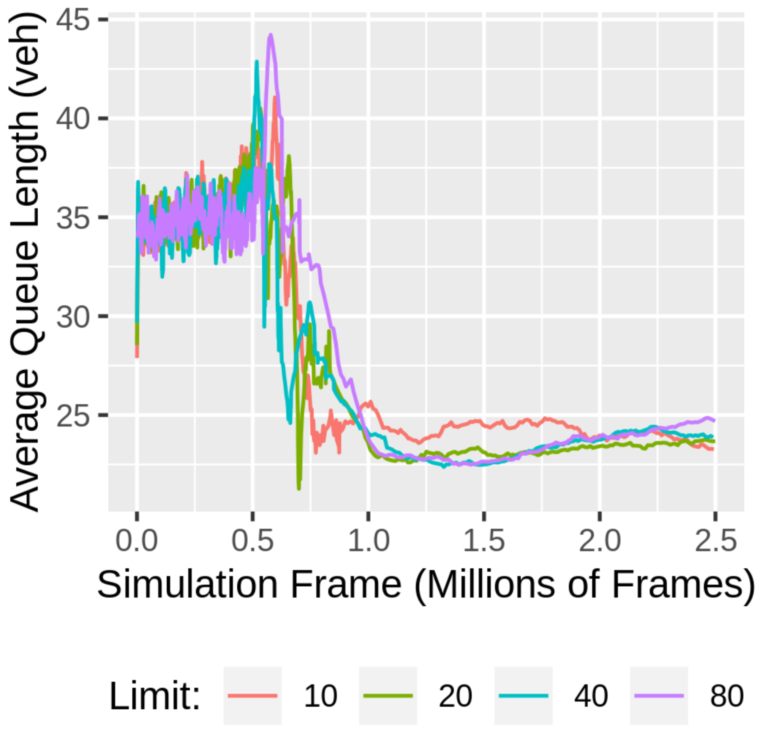
A sensitivity analysis was then conducted to identify the optimal settings for the model. The first parameter explored was the maximum queue length that the model can observe, and it was found that performance started to degrade as soon as fewer than 10 vehicles are observable. A variety of reward functions were then examined, but the best performance was observed when the model was penalised with 5 points for each queued vehicle and reward 0.4 points for each discharged vehicle at every decision point. In terms of training length, the best performance came from the baseline settings with observation occurring over the first 10,000 frames and then annealing ϵ over the next 50,000 frames. Longer annealing times did not improve performance. The optimal learning rate was found to be 10^{-6} and higher or lower rates resulted in models that would not converge to optimal policies. Setting the discount factor to 0.75 was associated with improved performance.

In terms of the state space, removal of the time of day information from the state space did not seem to have an effect on the final performance of the model, but this may be due to the volume scenario tested. However, in some cases, the time of day information seemed to help reduce training times.

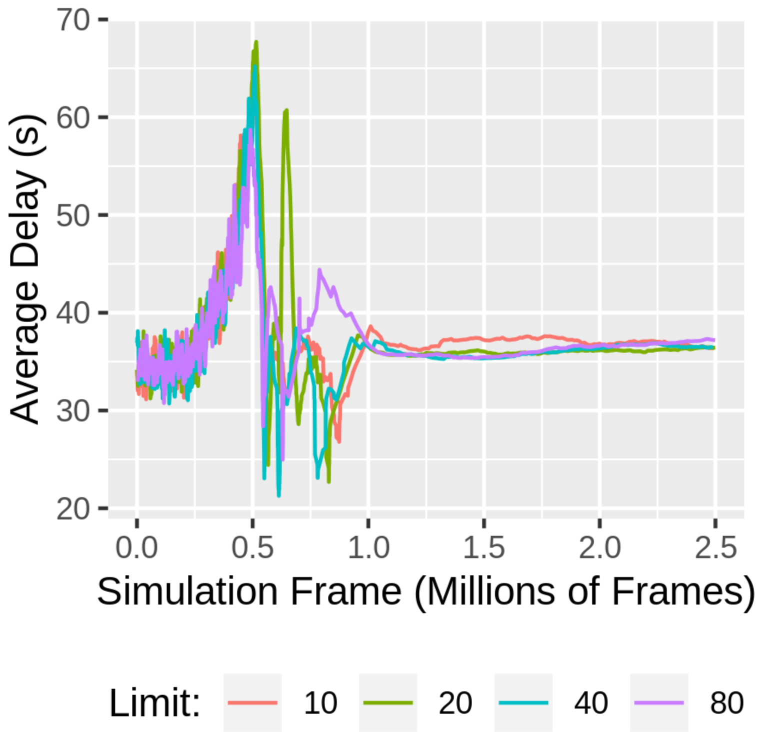
Providing the model with more useful experience when acting randomly was also associated with a reduction in training time. Since the agent typically must choose the *do nothing* action sequentially in most cases, the alternative training strategy forced the agent to on occasion select the *do nothing* action multiple times in a row. This allows for more varied experience as the performance of the traffic system rapidly degrades if the agent switches phases too soon.

Transfer learning is a strategy that allows a model to be pre-trained in one environment and then fine-tuned in another. This strategy has not been discussed at present in the literature, but will be vitally important if deep learning models are to be applied to traffic signal control problems. Transfer learning was found to significantly reduce training times when a model trained in one simulator was fine-tuned on a different simulator with different driver behaviour dynamics.

Finally, a full ring-and-barrier controller was trained and compared to the two-phase controllers. It was found that training a full ring-and-barrier controller requires almost three times the training time. Performance during the learning phase was also substantially worse than the two-phase design as the controller has more actions from which to choose.

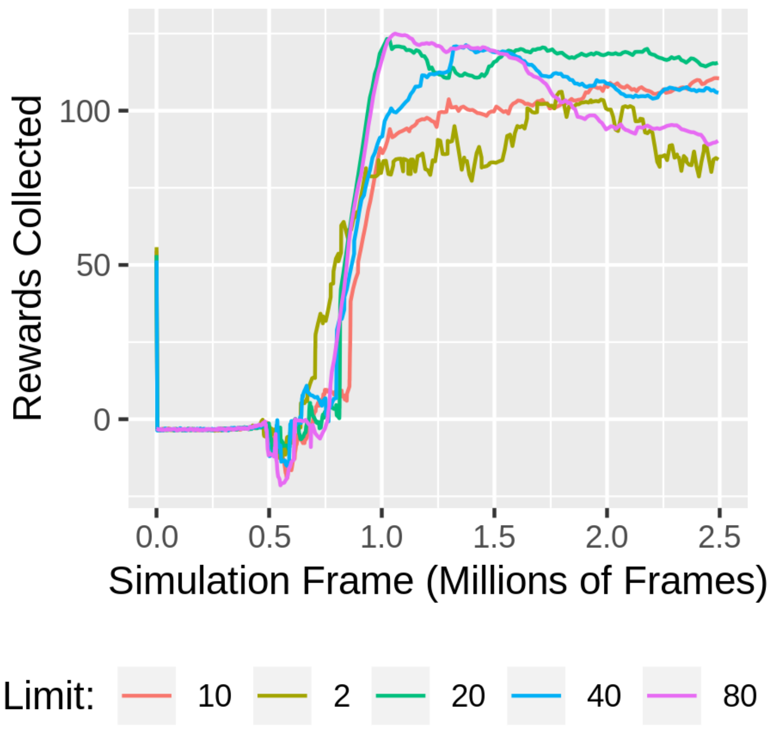


(a) Average Delay



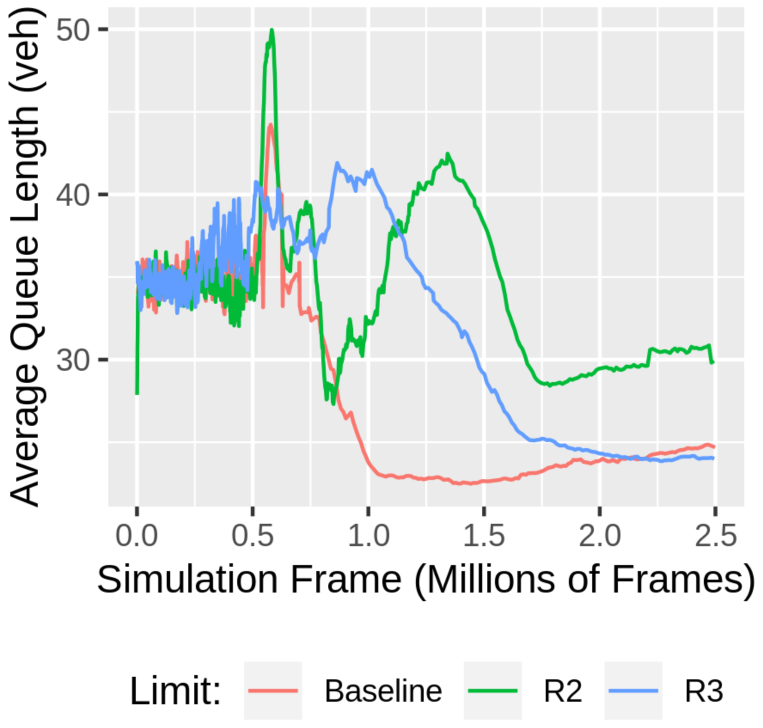
(b) Average Queue

Figure 4.9: Evaluation Results by Queue Length Observation Limit

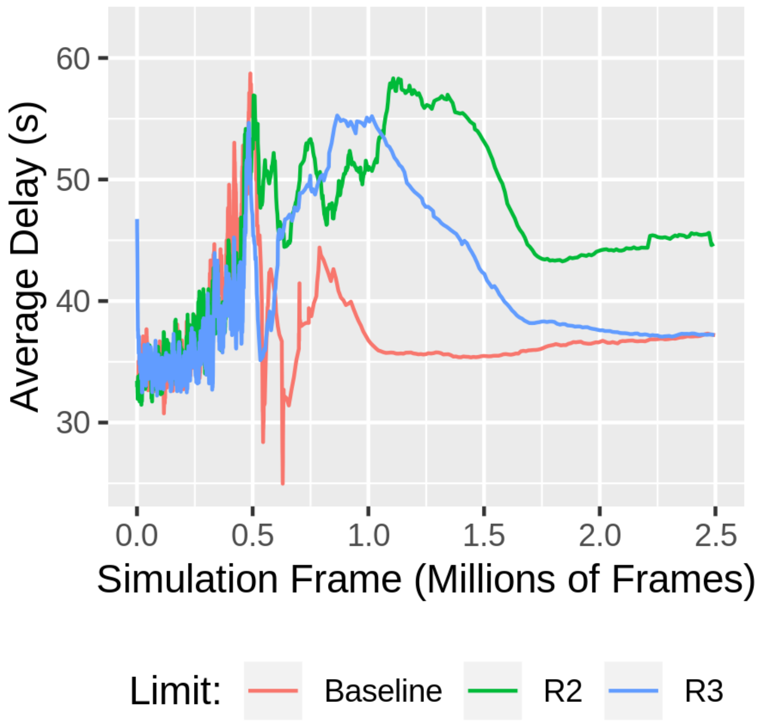


(c) Reward Accumulation

Figure 4.9: Evaluation Results by Queue Length Observation Limit

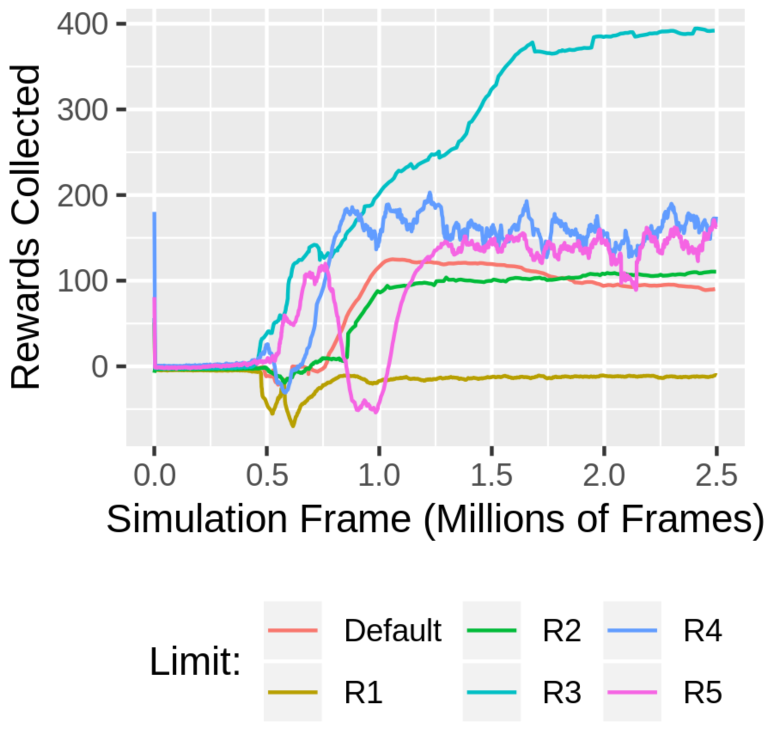


(a) Average Delay



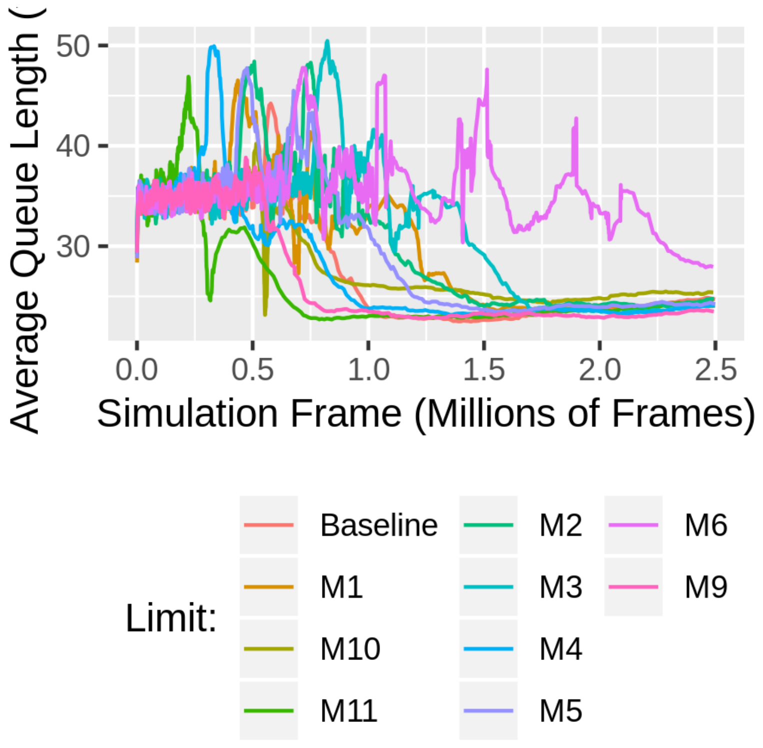
(b) Average Queue

Figure 4.10: Evaluation Results by Reward Function Parameters

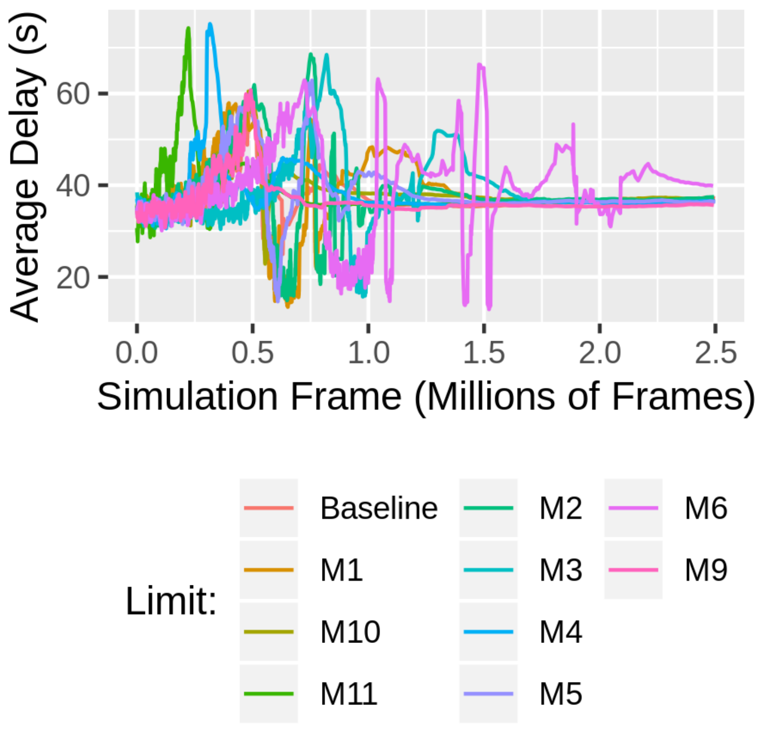


(c) Reward Accumulation

Figure 4.10: Evaluation Results by Reward Function Parameters

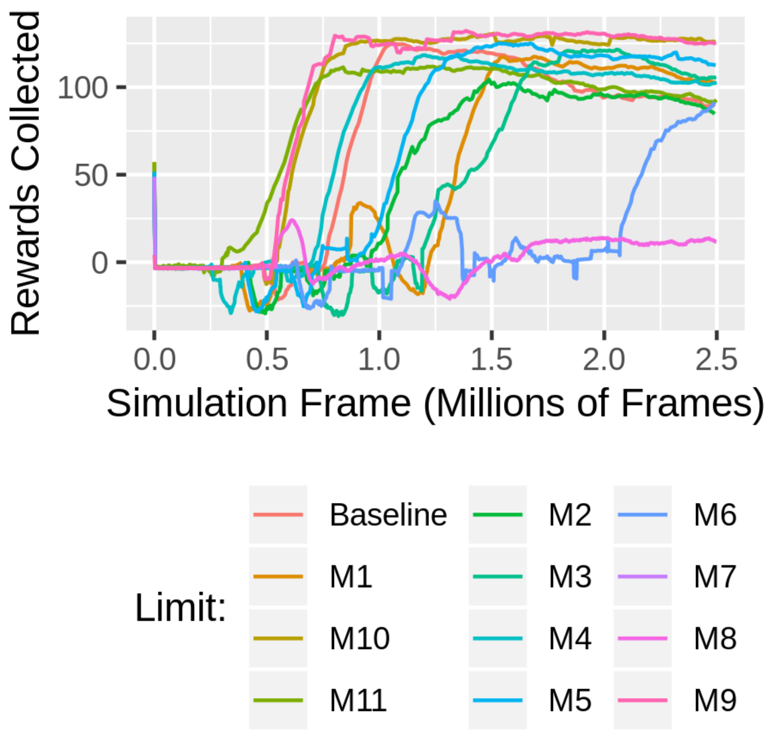


(a) Average Delay



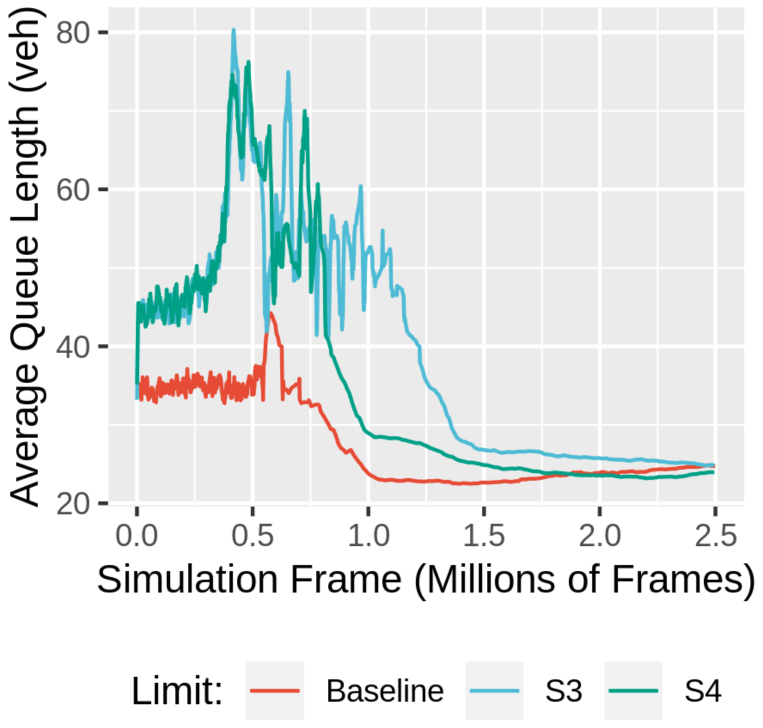
(b) Average Queue

Figure 4.11: Evaluation Results by Reward Function Parameters

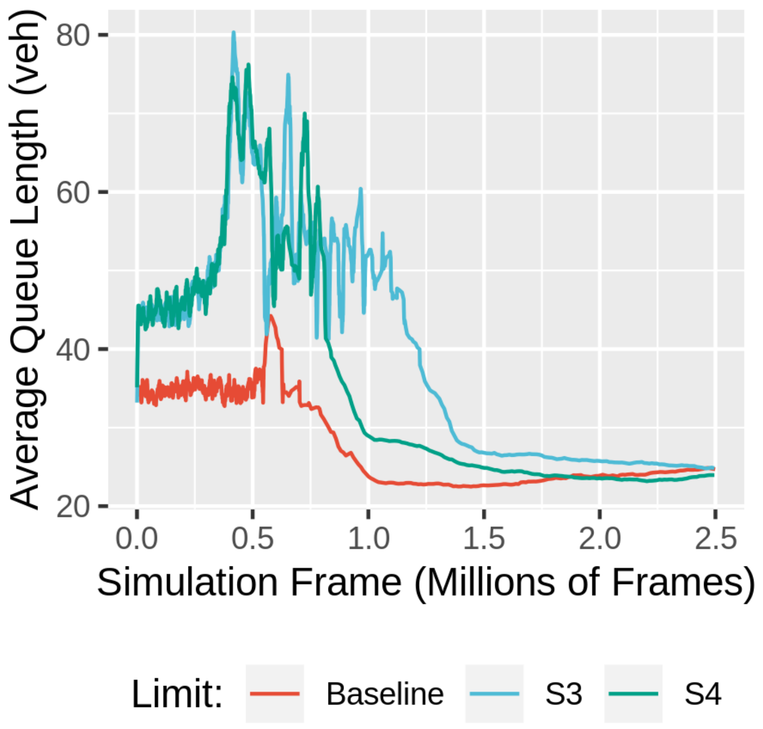


(c) Reward Accumulation

Figure 4.11: Evaluation Results by Reward Function Parameters

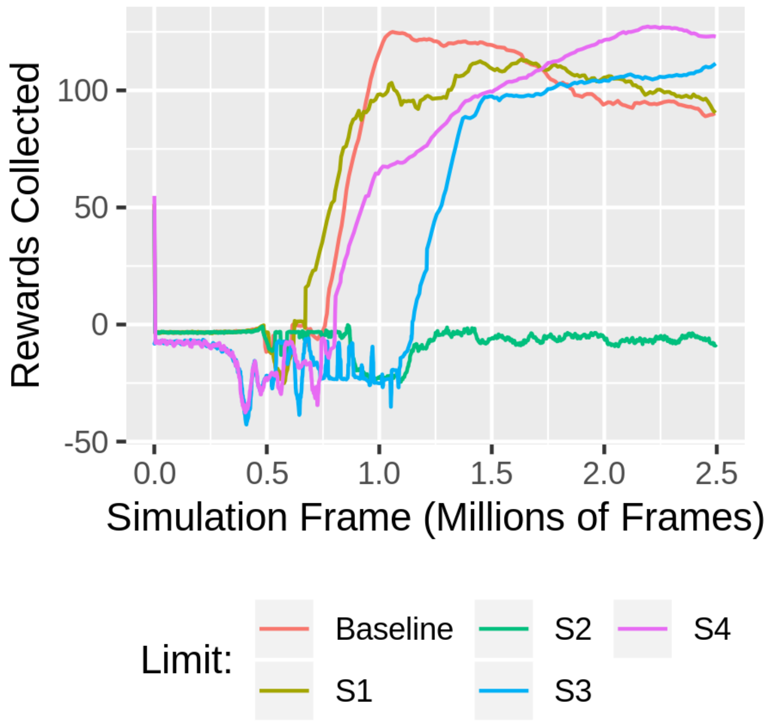


(a) Average Delay



(b) Average Queue

Figure 4.12: Evaluation Results by Time of Day Parameters



(c) Reward Accumulation

Figure 4.12: Evaluation Results by Time of Day Parameters

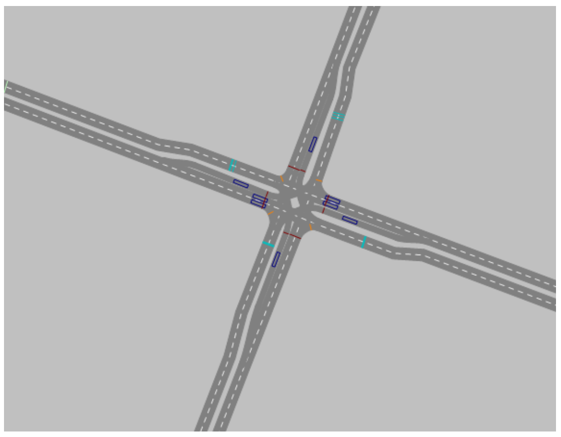
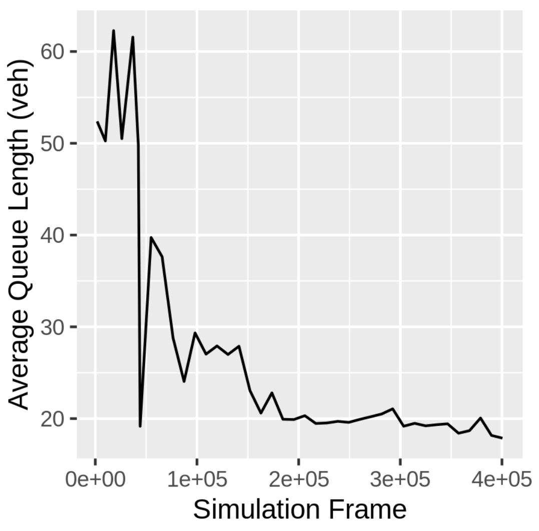
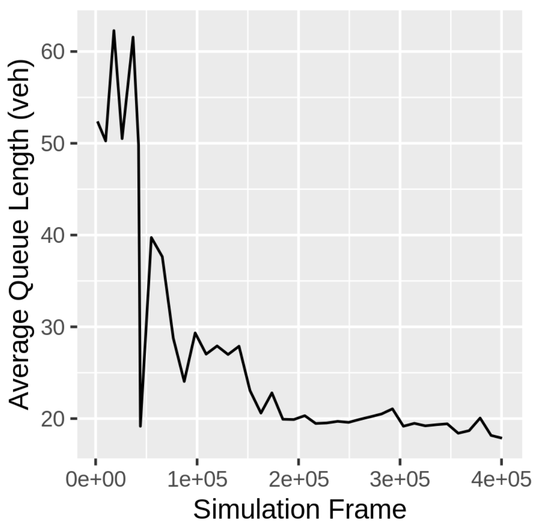


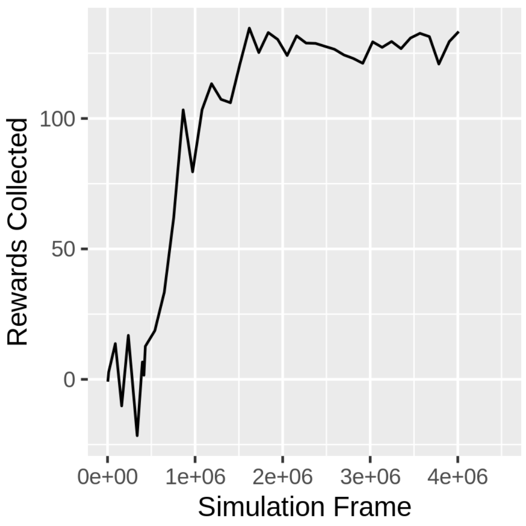
Figure 4.13: VISSIM Network



(a) Average Delay

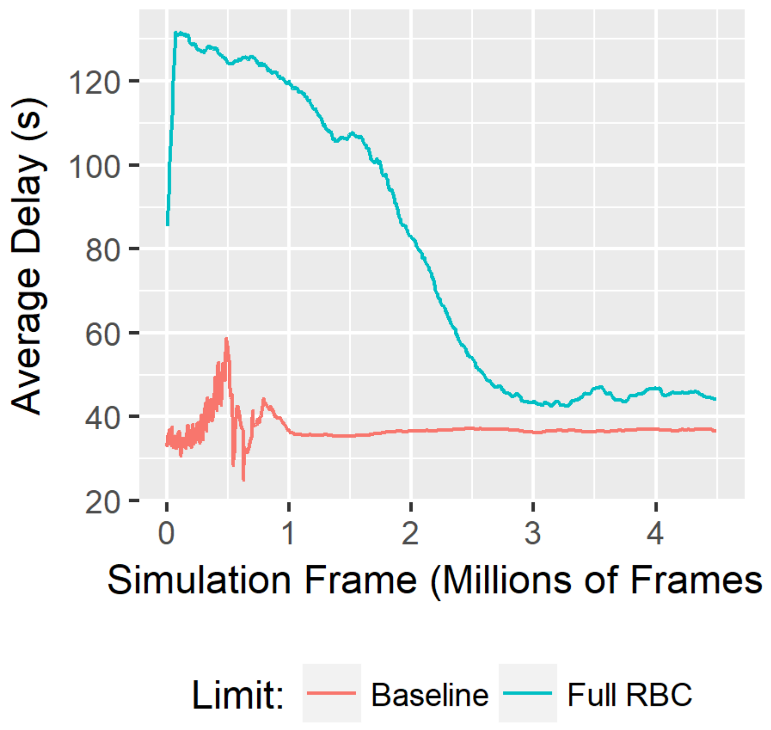


(b) Average Queue

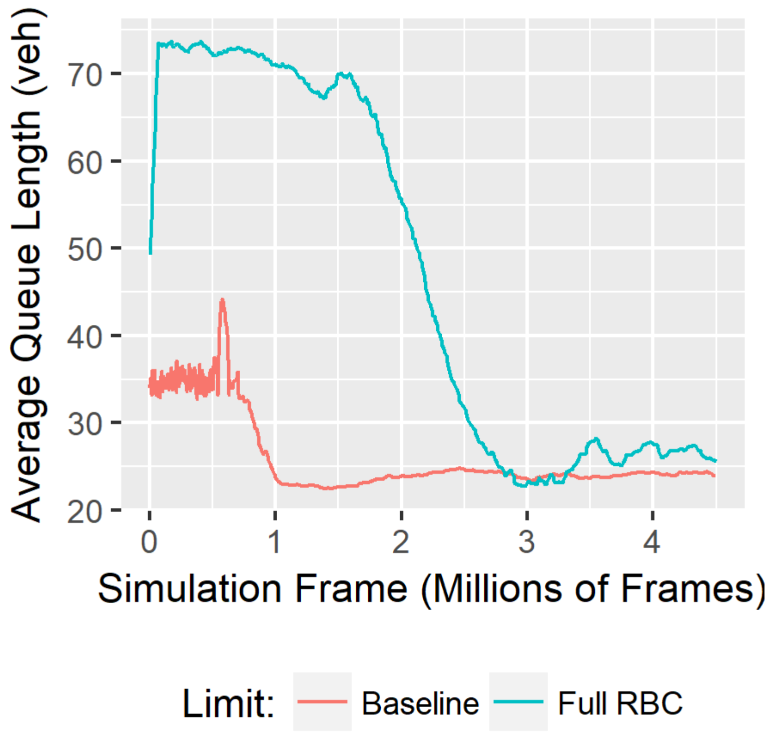


(c) Reward Accumulation

Figure 4.14: Evaluation Results from VISSIM with pre-trained Model

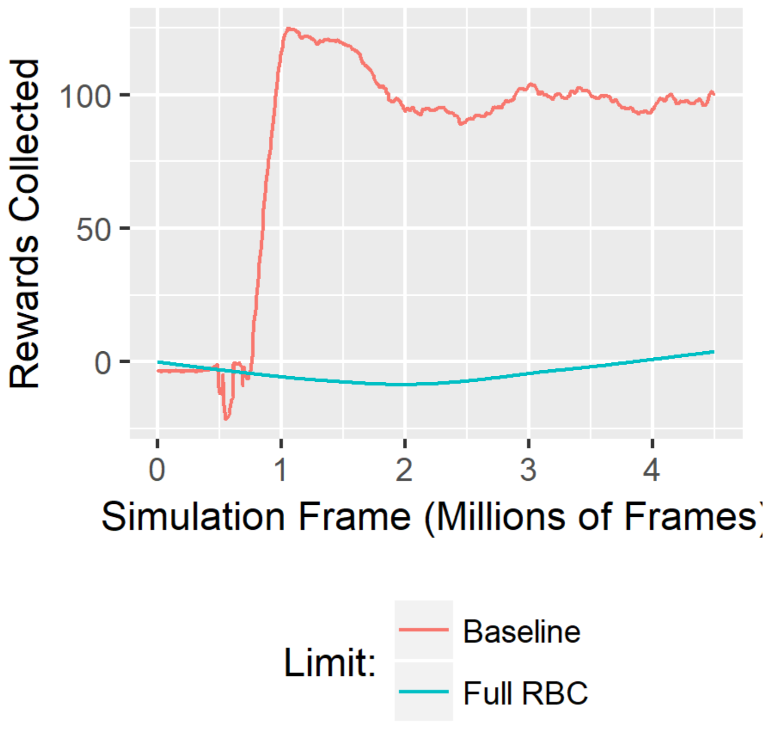


(a) Average Delay



(b) Average Queue

Figure 4.15: Evaluation Results from a Full Ring-and-Barrier Controller



(c) Reward Accumulation

Figure 4.15: Evaluation Results from a Full Ring-and-Barrier Controller

5 DEEP REINFORCEMENT LEARNING: SIGNAL CONTROL OF MULTIPLE INTERSECTIONS

Multi-intersection configurations introduce new challenges due to the traffic interaction between intersections in contrast to the case for the isolated intersections considered in Chapter 4. While an agent pre-trained on an isolated intersection will likely still perform well, some opportunities can be missed depending on the information available to the agent. In particular, in coordinated settings the actions taken at other intersections could be useful as indicators to predict when vehicles may arrive. However, using input from other intersections can create dependency relationships where the unavailability of this information can affect the performance of the model. Further, these relationships may not be generic and transferable as they are expected to depend on many factors such as the spacing of intersections, speed limits of roadways, and the level of congestion. This Chapter introduces a new DRL model for optimising the coordinated control of multiple intersections. The following sections present details of the proposed DRL model and a case study on its performance in comparison to the state-of-the-art adaptive traffic control system - SCOOT.

5.1 PROBLEM DESCRIPTION AND OVERVIEW OF METHODOLOGY

In Chapter 4 the single intersection traffic signal control problem was framed as a Markov Decision Process (MDP), in which the traffic system was described in terms of states S_t, which include all relevant elements such as the vehicles, their positions, speeds, the time of day, and an control agent was used to make decisions on the actions A_t based on observations O_t and the expected rewards R_t. A Deep Reinforcement Learning (DRL) model that uses a Deep Q Networks (DQN) was proposed to learn the Q value of choosing each action in a given state.

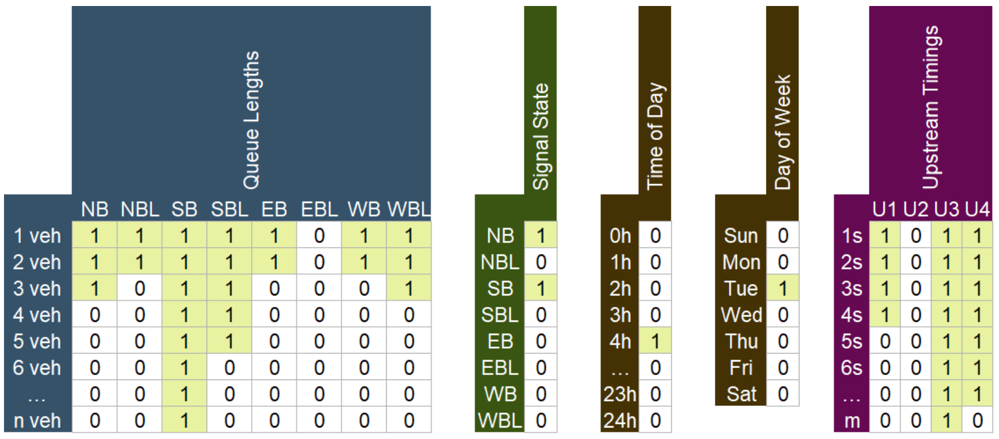
In this Chapter, we attempt to extend the single intersection model for adaptive traffic signal control and coordination of multiple intersections. Specifically, this research investigates the feasibility of making use of the trained single intersection DRL model for controlling individual intersections in a network with some minimal adjustment to the model structure. Unlike the conventional network or corridor signal coordination plan, there is no need to explicitly consider a common cycle time and signal offset between neighbouring intersections as decisions on phase transition at individual intersections are made on a second-by-second basis. As a result, the same MDP framework described in the previous chapter will be used for the control agent at each intersection, but with the additional assumption that each agent is able to observe the decisions of other agents (e.g., signal timing upstream intersections).

This research first considers the problem of applying the model for isolated intersections onto a network of multiple intersections. In this approach each agent solves its own local problem, and the key focus is on the scalability of the model by examining how the training of the model can be adapted to scale on a multi-intersection environment. In this approach one model is pre-trained on a generic intersection and then transfer learning is used to apply the model to intersections with different configurations. Then, this research considers the idea of providing timing information from upstream intersections to each agent as part of the observations they can make.

5.1.1 *Observation Matrix Changes*

The state space observation matrix as described in Section 4.3.1.1 is used as a starting point. This matrix contains vectors for queues at each approach, signal state, and time of day information. The implementation evaluated in this section allows up to 80 vehicles to be detected.

An extended structure is considered as an alternative scenario. Timing information $U_k = t_1, t_2, \ldots t_m$ from up to four upstream intersections $k \in 1, 2, 3, 4$ can be provided to each agent, as shown in Figure 5.1. This timing information is encoded similarly to all other data in a binary-vector format where t_i is 1 if the time in seconds since the last green is greater than i. In this design m is constrained to a maximum of 80 seconds. Since intersections may be arbitrarily spaced, this information's usefulness is tied to the specific intersection. To evaluate its impact, after training a model with a set of timing vectors, the vectors are removed to simulate the effect of a disconnection between the two intersections.



Queue Lengths

	NB	NBL	SB	SBL	EB	EBL	WB	WBL
1 veh	1	1	1	1	1	0	1	1
2 veh	1	1	1	1	1	0	1	1
3 veh	1	0	1	1	0	0	0	1
4 veh	0	0	1	1	0	0	0	0
5 veh	0	0	1	1	0	0	0	0
6 veh	0	0	1	0	0	0	0	0
…	0	0	1	0	0	0	0	0
n veh	0	0	1	0	0	0	0	0

	Signal State
NB	1
NBL	0
SB	1
SBL	0
EB	0
EBL	0
WB	0
WBL	0

	Time of Day
0h	0
1h	0
2h	0
3h	0
4h	1
…	0
23h	0
24h	0

	Day of Week
Sun	0
Mon	0
Tue	1
Wed	0
Thu	0
Fri	0
Sat	0

Upstream Timings

	U1	U2	U3	U4
1s	1	0	1	1
2s	1	0	1	1
3s	1	0	1	1
4s	1	0	1	1
5s	0	0	1	1
6s	0	0	1	1
…	0	0	1	1
m	0	0	1	0

Figure 5.1: Extended State Space

5.1.2 *Transfer Learning*

In Section 2.4.4 the idea of transfer learning was introduced and in the sensitivity analysis conducted in Section 4.7 transfer learning was evaluated by pre-training a model on one simulator and transferring the model to another. In this corridor evaluation the idea of transfer learning was further explored by transferring a model pre-trained on a generic intersection to an arbitrary set of real-world intersections in this corridor.

5.2 MODEL TRAINING AND EVALUATION

5.2.1 *SCOOT Emulation*

In order to evaluate the performance of the proposed model, an emulator for one of the most popular adaptive traffic control systems - SCOOT is implemented and used for benchmarking purposes. Traditional adaptive traffic control systems vary in their conventions and approach to the coorindation problem. Many of these systems are proprietary and the exact workings are not available. However, the theoretical basis for most of these approaches is widely published and available. To benchmark our proposed DRL optimisation model, one of the most successful adaptive traffic signal control systems - SCOOT is used as the comparison reference. SCOOT is an adaptive algorithm originally developed in the 1980's and works by incrementally adjusting the split time, cycle time, and offset at coordinated intersections. In order to enable a simulation-based analysis, the basic functions of the SCOOT system must be implemented in the same simulator where our proposed solution is tested. This section describes how the basic functions of the SCOOT system is emulated in SUMO based on past work published on SCOOT. These include the operating manual for SCOOT (Siemens Mobility, Traffic Solutions, 2016), some early work published outlining the approach when it was first proposed (Hunt et al., 1982; Robertson and Bretherton, 1991; Robertson, 1986), and versions of SCOOT replicated by other researchers (Raphael, 2018). The following sections provide an overview of the modules and functions replicated from SCOOT, including the split, cycle, and offset optimisers. A detailed overview of the implementation, including pseudo-code, is provided in Appendix A in Section A.4.2. The implementation outlined in these sections tries to maintain and use current SCOOT default parameters wherever possible.

5.2.1.1 *Split Optimiser*

SCOOT's split optimiser changes the amount of green time allocated to individual movements at the intersection. The fundamental principle governing the decision is balancing the degrees of saturation (i.e. the ratio of the predicted arrival flow rate to the capacity flow) for all approaches. This strategy attempts to ensure fairness for all users sharing the intersection's space and produces a solution that is approximately optimal in terms of delay minimisation. The balancing can prioritise one direction, but by default aims to make all approaches equal.

In SCOOT, these decisions are made 5 seconds before the scheduled end of each phase with three possible outcomes: terminating early, doing nothing, or extending the phase. When the controller chooses to terminate early or extend, by default this is done for a temporary extension or termination of 4 seconds. Then, for the following cycle, the timing plan is permanently altered by 1 second (Raphael, 2018; Siemens Mobility, Traffic Solutions, 2016). When making the decision, SCOOT uses predicted arrival flow rates for each approach by applying a cyclic flow model.

The cyclic flow model applies the Robertson platoon dispersion model (Raphael, 2018) with the same functional form previously discussed in Equations 2.5 and 2.6 in Section 2.2.2 on page 24. The Robertson equations require specification of the calibration parameters α and β which are taken as 0.35 and 0.8 respectively based on recommendations for moderate demand (Gordon, 2003).

5.2.1.2 *Cycle Optimiser*

With default settings, SCOOT's cycle optimiser runs every five minutes. The cycle optimiser looks at the degree of saturation at all intersections on the corridor under the control of SCOOT in a specified region and tries to adjust this cycle time such that the maximum degree of saturation at any intersection is closest to 0.9. This is done by evaluating an increase or decrease of 0, 4, 8, 16, or 32 seconds to the cycle time. SCOOT also considers the impact of changing some minor streets to double or half cycle lengths (Raphael, 2018; Siemens Mobility, Traffic Solutions, 2016)

5.2.1.3 *Offset Optimisation*

By default, offset optimisation is done once per cycle, just before the final phase. SCOOT considers the effect of adjusting the reference clock of the intersection by 4 seconds (forwards or backwards) or retaining the current offset. This evaluation occurs by comparing the number of stops that would be incurred for the cyclic flows predicted and selecting the adjustment associated with the fewest stops (Raphael, 2018; Siemens Mobility, Traffic Solutions, 2016).

5.2.1.4 *Saturation Flow Rate*

In field applications of SCOOT, a key parameter called the *saturation occupancy* is used to estimate the degree of saturation and should be configured and calibrated to ensure best performance. This value is similar to the concept of *saturation flow* but specific to loop-detector-based applications (Siemens Mobility, Traffic Solutions, 2016) and represents the fraction of time that a detector will be actuated when the intersection is saturated. In field settings, it depends on local driving conditions and characteristics of the detectors. However, in simulation settings the function of detectors is different and not dependant on the same factors. Loop detectors in SUMO distinguish individual vehicles and can count volume directly. Therefore, the concept of *saturation occupancy* is not emulated; instead saturation flow is directly considered when estimating the degree of saturation.

5.2.2 *Model Training*

An overview of the training process is shown in Figure 5.2 for all learning scenarios. When transfer learning is not applied, the process continues directly to fine tuning and evaluation on the same network. During transfer learning, models were first pre-trained on an isolated intersection for 1 million simulation frames before further fine-tuning on their specific intersections. This duration was chosen in reference to the sensitivity analysis conducted in Chapter 4. The training process is identical to the process outlined in 4.5 and 4.6. Volumes used for the pre-training are the same as those shown in Figure 4.8. As in Chapter 4, a model without turning movements and a model with turning movements are trained, and when turning movements are included the turning volume is set to 10% for each turning direction. An ϵ-greedy approach is used but with the optimised settings identified in the sensitivity analysis. In particular, an observation length of 10,000 timesteps and 200,000 further steps to anneal ϵ were used. A discount rate of 0.9 was used.

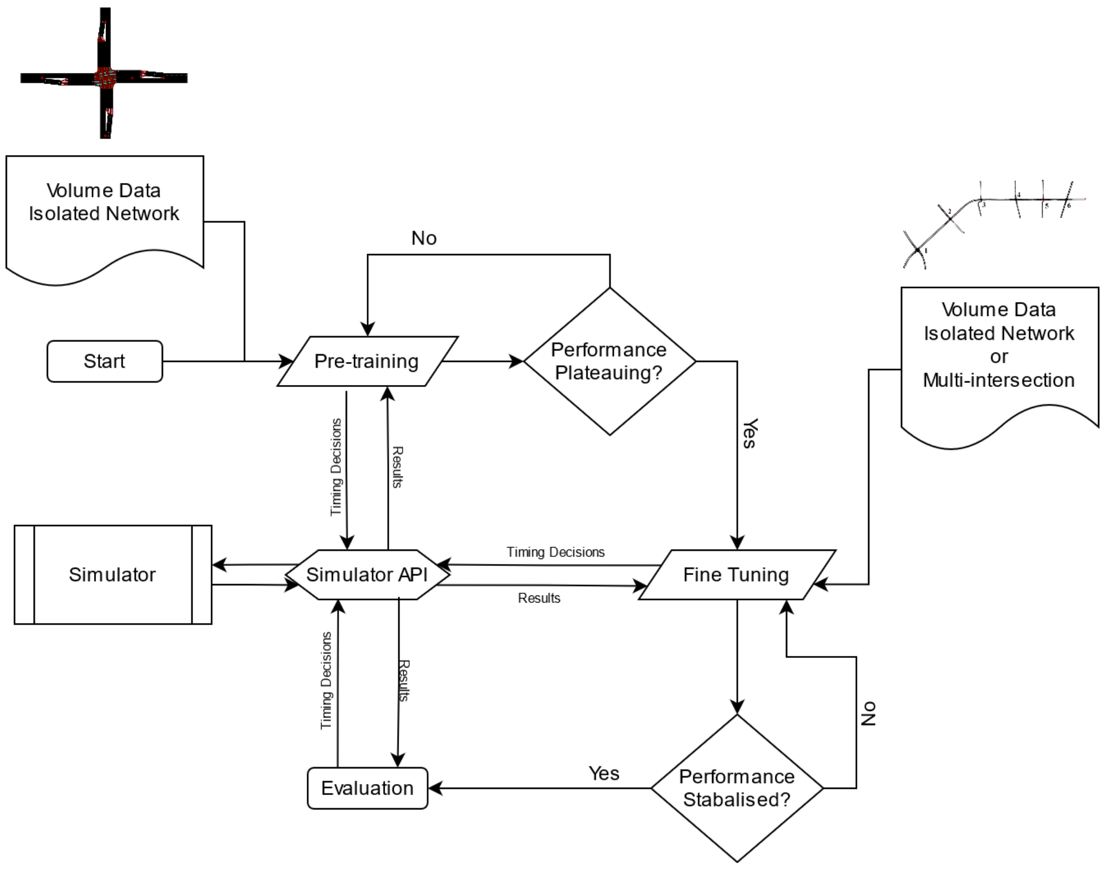


Figure 5.2: Training Process

The two-phase model is explicitly designed to test transfer learning, including both a transfer learning scenario and a scenario that trains the entire model on the corridor. A full model with all turning movements is then trained and evaluated using a

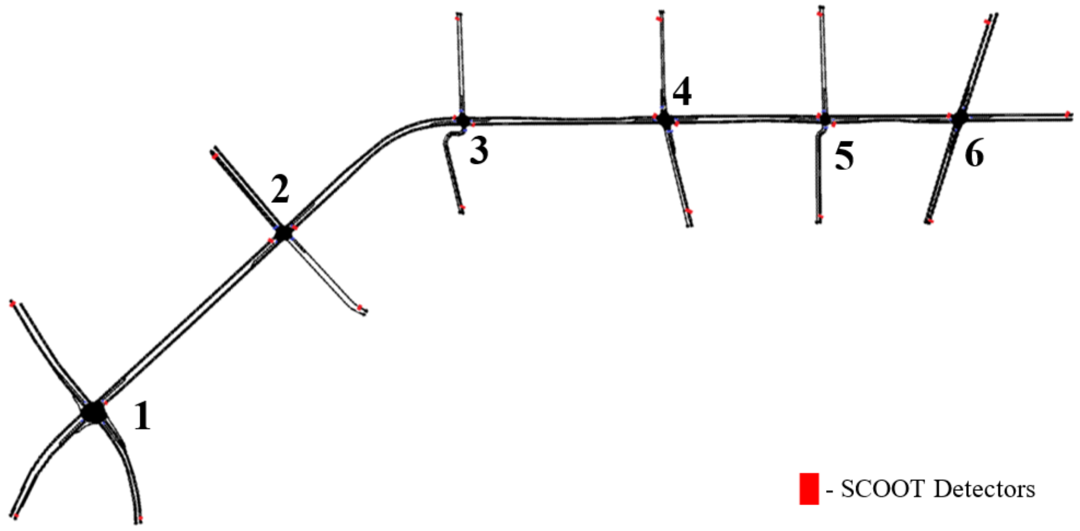


Figure 5.3: SUMO Simulated Corridor

transfer learning approach. During transfer learning, ϵ-greedy training is still used, however ϵ is set to 0.2 and the observation time is reduced to 5,000 simulation frames. ϵ is then annealed over 125,000 steps. This allows the controller to rely on exploration more as it learns to manage an intersection with a different configuration.

Similarly the SCOOT controller and DRL models with the extended state space are all trained and applied on the same corridor. For the DRL scenarios, two scenarios are evaluated. In both DRL models are trained on their individual intersections using the extended state space. Then, in the first scenario training is stopped and evaluation is conducted. In the second scenario, the upstream timing information is disabled from the state space and training is stopped. The model's performance is assessed without the timing information to assess its impact in the model's decision-making.

5.2.3 *Case Study Network*

A six intersection corridor was simulated in SUMO. An overview of the network is shown in Figure 5.3 and the locations of simulated SCOOT detectors are also highlighted. Detectors for SCOOT were placed immediately after the discharging exit for the corridor intersections and immediately as vehicles enter the simulation when no upstream intersection was present. In this corridor, intersections #1 and #6 are major, while intersections #2-#5 are minor. However, for simplicity all intersections are modelled as SCOOT enabled and a DRL model is trained and applied for each intersection. The field environment has actuation on all major left turns and some of the left turns for minor approaches; further, minor intersections are all semi-actuated. Field timings were obtained and used when evaluating scenarios with all turning movements. Synchro was used to generate timing plans for scenarios where turning movements were excluded. Intergreen time was altered from the reference plan to be 5 seconds instead of 4, as times less than 5 were found to cause lock-ups and priority issues with turning movements. A summary of the total volumes at all intersections is shown in Table 5.1.

5.3 RESULTS

The case study outlined above is used to generate a number of comparative studies. The transfer learning scenarios tested include studies to test the advantages of transfer learning, a comparative study to SCOOT, an evaluation of the extended observation matrix, and a scenario with additional volume.

5.3.1 *Effectiveness of Transfer Learning*

Transfer learning was evaluated by training the system using two approaches. In the first a single model is pre-trained and then transferred to each individual intersection wile in the second all intersections are trained fully from scratch.Average cumulative delay and queue lengths were collected in SUMO and averaged for all intersections in the simulation. Due to limitations with SUMO's data collection tools, delay is not recorded until a vehicle exits the simulation, thus in congested situations that trigger terminal states the results may are inaccurate. Transfer learning was first evaluated using the two-phase model to identify any implementation challenges and then using the full ring-and-barrier controllers.

In the two-phase evaluation, two models were developed to assess the impact that transfer learning has. In the first transfer learning is applied and in the second the model is fully trained on the corridor. During transfer learning, the model's weights are transferred after pre-training the model for 1 million frames to individual models on each intersection. Delay, queue, and per-route delays are shown in Figure 5.5 for the transfer learning scenario and full retraining scenario. For the transfer learning scenario, only results after the transfer has taken place are shown.

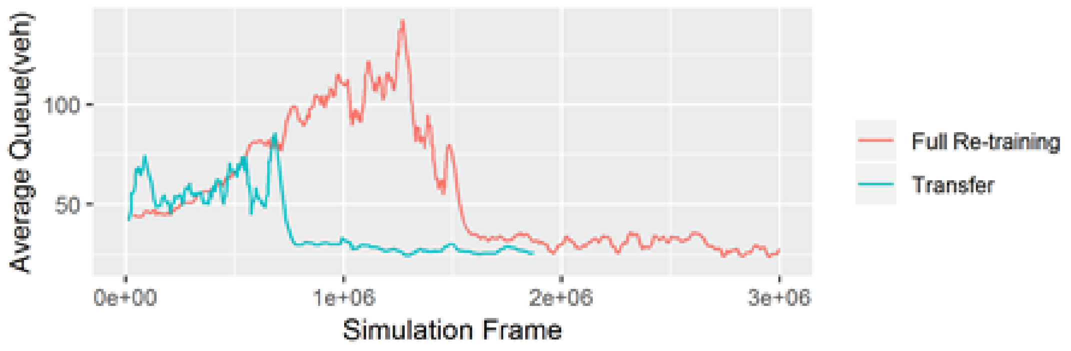
Since ϵ is reset to 0.2, the agent initially causes some minor congestion after transferring the model, however the issues were mostly localised to the streets that had different configurations and terminal states were never triggered by the model. Convergence is achieved in about half the time when compared to re-training the model with no experience and provides similar final performance. As can be seen in Figure 5.4b, when compared to the Synchro plan, the DRL models had a tendency to prioritise movements on the minor street rather than the corridor as the main corridor, University Avenue, has higher delay in both DRL scenarios but the side streets have lower delays. This may be a difference in design goals or may be due to the simplicity of two-phase operations, but as a result of this difference the overall delay in the DRL scenario is lower.

Following this assessment, the transfer learning approach was used with a pre-trained full Ring-and-Barrier controller. The agent was trained on an isolated intersection for a total of 3 million frames and weights were transferred similarly to the case

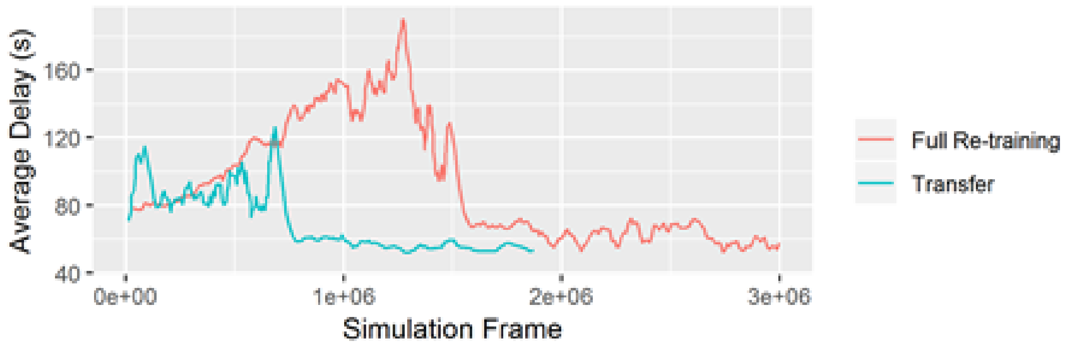
Table 5.1: Summary of Volumes at Each Intersection

Name	**Scenario Avg. Vol.**	**Peak 15-min. Vol.**[1]
University/Westmount (#1)	2504	3380
East Leg	831	1180
West Leg	393	460
North Leg	690	876
South Leg	589	864
University/Seagram (#2)	2082	2880
East Leg	1121	1640
West Leg	615	756
North Leg	96	336
South Leg	250	148
University/Phillip (#3)	1461	1860
East Leg	676	828
West Leg	648	808
North Leg	129	208
South Leg	8	16
University/Albert (#4)	1934	2828
East Leg	570	776
West Leg	608	816
North Leg	521	916
South Leg	234	320
University/Hazel (#5)	1377	1924
East Leg	592	744
West Leg	676	952
North Leg	35	60
South Leg	73	168
University/King (#6)	1865	2420
East Leg	525	632
West Leg	697	892
North Leg	290	408
South Leg	353	488

[1] Estimated by using the maximum total 15-minute volume from the data

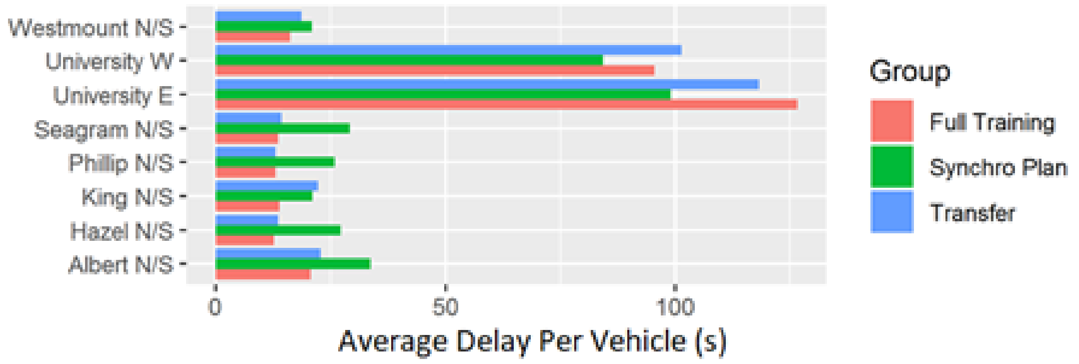


(a) Average Queues



(b) Average Delay

Figure 5.4: Evaluation Results for the Two-Phase Scenario



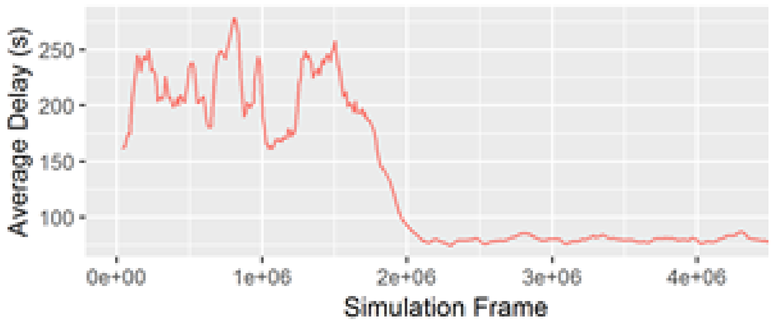
(a) Average Delay by Link

Figure 5.5: Evaluation Results for the Two-Phase Scenario

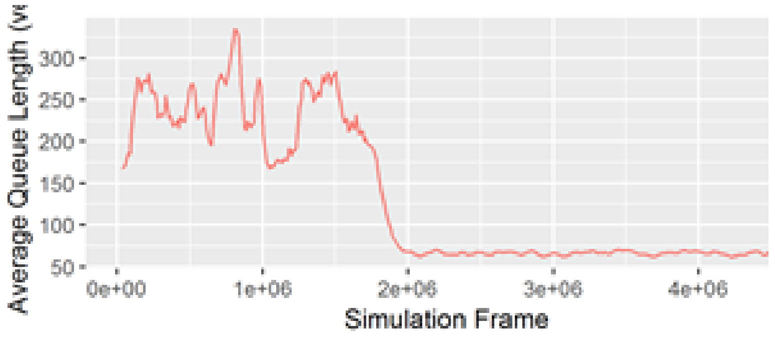
of the two-phase controllers. An additional 2 million frames of fine-tuning were required to fine-tune the individual models. The entire training process was completed over 10 days on an intel i7-4790 processor.

When compared to the actuated controller running with existing field timings, the DRL model provided noticeable benefits. The results are shown in Figure 5.6 and

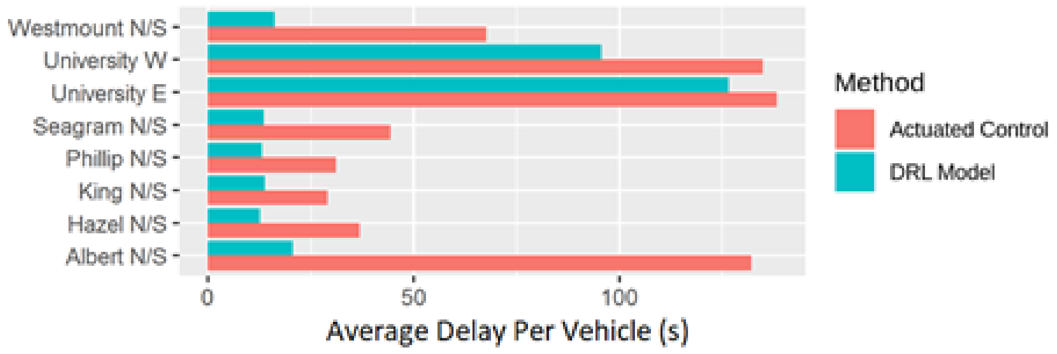
5.7.Field timings did not provide adequate service to the left turn movements at intersections #1 (Westmount) and #3 (Albert) during peak periods and queue spillback would block the through movement of these approaches. On average under the field plan vehicles had 83 seconds of delay and an average of 91 vehicles were stopped behind a red light throughout the entire simulation at any given time. The DRL reduced the delay to 78 s with 66 vehicles waiting behind a red light resulting an overall improvement of 5% on the delay. However, the improvements may be higher as queues at intersection #3 blocked vehicles from entering the simulation.



(a) Average Delay

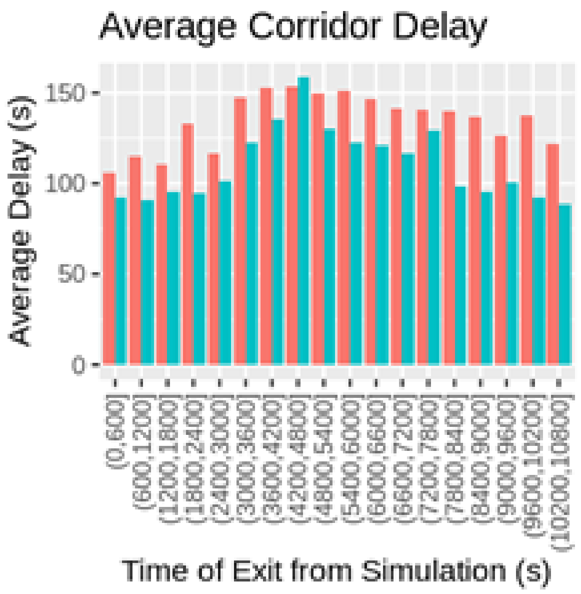


(b) Average Queue

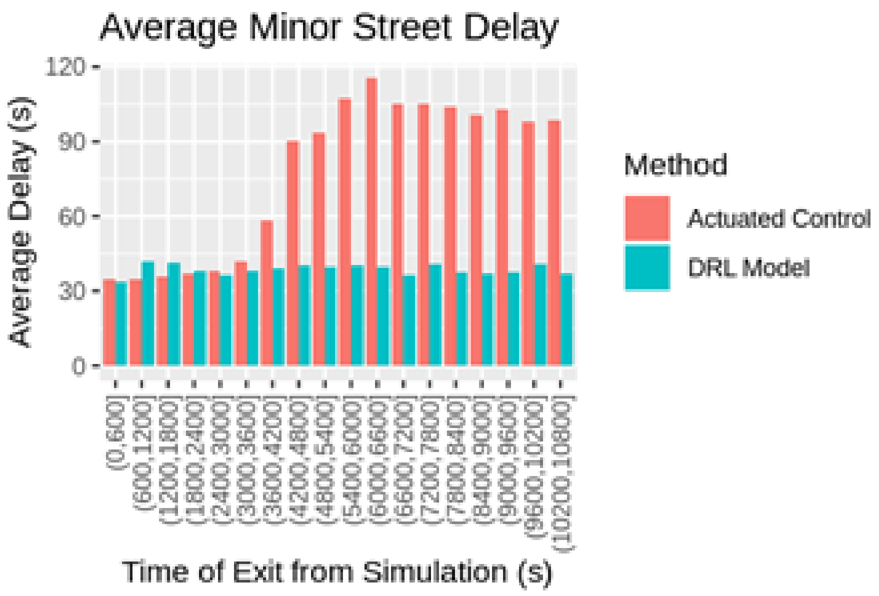


(c) Average Delay by Link

Figure 5.6: Evaluation Results for the Full Ring-and-Barrier Scenario



(a) Average Corridor Delay



(b) Average Minor Street Delay

Figure 5.7: Evaluation Results for the Full Ring-and-Barrier Scenario

When comparing the performance on the minor and major streets, the DRL model has a tendency to balance delays between the major and minor streets better and avoids the creation of congested conditions. In contrast to the results from the two-phase controller, in this setting the DRL model provides better performance to both the side streets and the main corridor.

5.3.2 *Coordinated Control: A Comparison to SCOOT*

The full Ring-and-Barrier design was extended and evaluated through a comparison against SCOOT and semi-actuated signals. In this study all DRL models were fully trained from scratch without transfer learning. A single model was trained for 5 million frames on the corridor over a multiple-day period. Then, training was stopped and the model's final performance was compared, the results of which are highlighted in Table 5.2

Table 5.2: Comparative Results to Traditional Control Methods and SCOOT

Name	**Average Travel Time**	**Average Delay**	**Average Queue**
Semi-Actuated	160.1 s	92.3 s	83.4 veh
SCOOT	143.0 s	75.8 s	67.8 veh
DRL	139.1 s	70.3 s	58.7 veh
DRL-NN[1]	169.1 s	102.0 s	92.1 veh

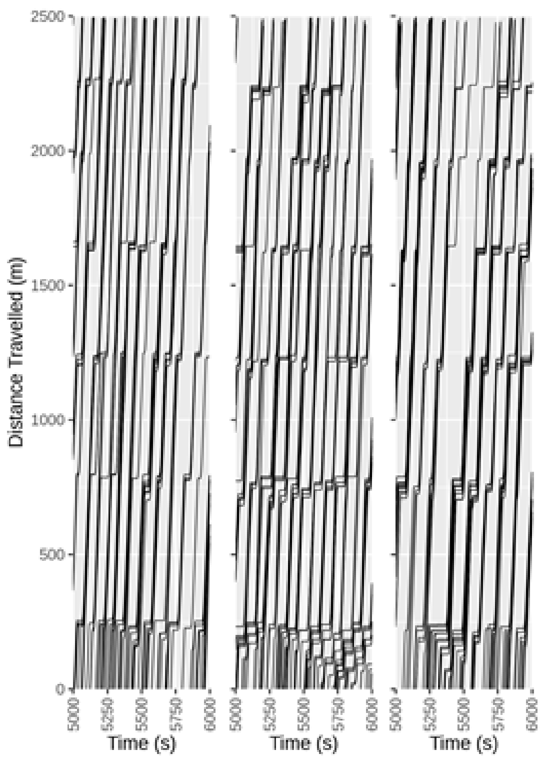
[1] In the NN scenario, neighbouring information is excluded during evaluation

Overall, the DRL model performed the best and reduced delays by 7% when compared to the SCOOT emulator. When the timing information from adjacent intersections was removed from the DQN model the performance of the agents degraded significantly, and performance was worse than the actuated controller. This result was surprising and may suggest some over-fitting to the scenario. Importantly, exclusion of the neighbouring timing information is indistinguishable from "zero seconds since the upstream phase showed green" to the agent as the vectors were blanked from the state space, and this may create a false signal to the model. Regardless, the results of this analysis show that the information from neighbouring intersections is valuable to the controller but that a dependency relationship may be formed. In real-world situations, if information becomes unavailable, the model will either need to be trained to consider this information or an alternative model may need to be applied as a backup.

5.3.2.1 *Trajectory Comparison*

Trajectory data was captured using SUMO's position tracing tools. Trajectories for the SCOOT, DRL, and DRL-NN scenario were plotted for the busiest 1000s period (5000s to 6000s) for vehicles travelling straight on the corridor from one of the evaluation runs. These trajectories are shown in Figure 5.8 and 5.9

When compared to the SCOOT scenario, the DRL model has tendency to prefer lower cycle lengths. Importantly, unlike SCOOT the DRL model is not constrained to a network cycle time, and so individual intersections may change the light as best suits their demand. Consequently, the SCOOT emulator creates some additional delay due to the longer cycles. However, the progression provided by both models is very similar.



(a) DRL (b) DRL-NN (c) SCOOT

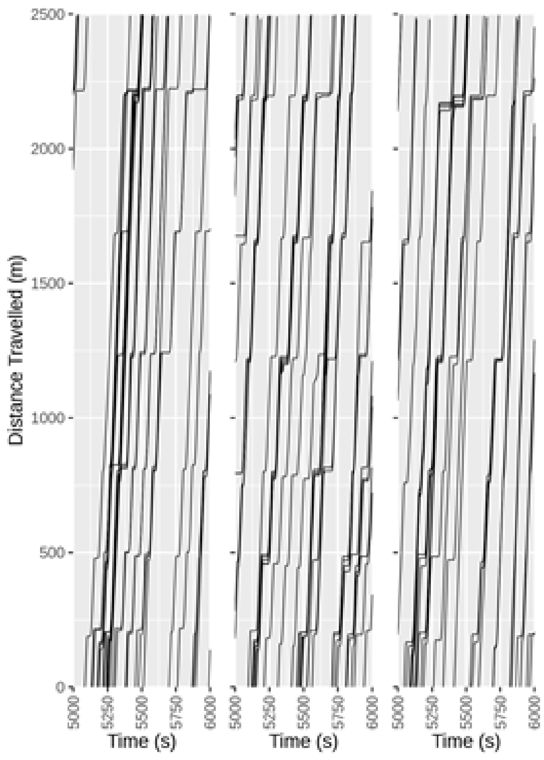
Figure 5.8: Eastbound Trajectory Results for Each Model Scenario

In contrast, the DRL-NN scenario's performance was noticeably worse and progression is routinely interrupted. Additionally, the model causes over-saturated congested conditions at some intersections, especially the first intersection in the eastbound direction. The tendency of the DRL model to minimise the cycle length was counterproductive in the case of the DRL-NN and the model had a tendency to shorten the cycle lengths even more.

5.3.2.2 *Performance Sensitivity to Traffic Demand*

Additional volume was then applied to the fully trained network to model changing traffic conditions. These were done as linear volume increases of of 25% additional volume and 50% additional volume on all routes compared to the baseline scenario. The 25% additional increase in volume creates over-saturated conditions at some intersections during the peak 15 minutes, while the additional 50% volume creates over-saturated conditions throughout the network, blocking vehicles from entering the simulation on some approaches.

The performance of the DRL algorithm and SCOOT are shown below in table 5.3. In this table *Blocked Vehicles* are vehicles that could not be inserted into the simulation (and so are not counted in the delay measures.



(a) DRL (b) DRL-NN (c) SCOOT

Figure 5.9: Westbound Trajectory Results for Each Model Scenario

Table 5.3: Comparative Results to Traditional Control Methods and SCOOT

Volume Scenario	**Model**	**Average Delay**	**Blocked Vehicles**
25%	SCOOT	131 s	300 veh
25%	DRLs	115 s	251 veh
50%	SCOOT	312 s	1103 veh
50%	DRL s	288 s	721 veh

These results show that the DRL model is able to keep its performance as the volume scenario changes, and continues to provide improvements when compared to SCOOT. Average delays were lower by 13% and 8% respectively for the 25% and 50% additional volume scenarios. Approximately 17% and 35% more vehicles were able to be inserted into the network when compared to the SCOOT scenario for the 25% and 50% volume scenarios, which suggest the DRL model is able to improve network throughput when compared to other alternatives.

5.4 CONCLUSIONS

The results of the analyses in this chapter have highlighted the potential for DRL to improve the performance of multi-intersection configurations. The analysis was divided into two main approaches. In the uncoordinated configuration, transfer learning was used to pre-train models on a generic intersection and then fine-tuned on each individual intersection. The results of the analysis showed that transfer learning cut convergence time in half. In addition, during training performance was better than the full re-training as the model was able to exploit its experience to avoid extremely poor decisions, with peak delays during training about 60% less than those when retraining from scratch. In the multi-intersection configuration, each model makes control decisions that are locally optimal, and the model had a tendency to provide much better service on the minor approaches than the actuated controllers with timings used from the field and Synchro. The model was also better able to react to the changes in demand during the analysis period and avoid significant delays. When implementing a full ring-and-barrier controller, the DRL model reduced average delays by at least 5%.

A coordinated design was also evaluated by extending the state space of the model. In this design the timing information upstream was provided to each downstream intersection, and the agent was trained from scratch to use this information. The resulting analysis highlighted that when compared to SCOOT the DRL model had a tendency to select shorter cycle times. This resulted in shorter delays, especially at the entry intersection. Delays were reduced by 7% when compared to the SCOOT emulator. A sensitivity analysis was also conducted by removing the timing information after the models were trained. This was done by setting the values of the matrix to zero (which is equivalent to a situation where the upstream intersection has just timed or is timing green). The performance of the model declined significantly with this change, and had worse performance than the Semi-actuated controller. This result suggests that the timing information is used by the model to make control decisions, and that the controller may make sub-optimal decisions in the presence of false information.

Part III

CONCLUSIONS AND CONTRIBUTIONS

This part's chapter gives a summary of the research done, the key findings, and the limitations.

6 CONCLUSIONS AND CONTRIBUTIONS

The research conducted as part of this thesis has focussed on enabling traffic control systems to capitalise on recent advances in technology, data availability, and developments in AI. These areas are of critical importance in developing the next generation of traffic control systems, and offer significant opportunities for transportation agencies to improve the performance of the existing transportation system. These are especially important as cities continue to grow and embark on their Smart City journey.

This research has made contributions to the literature through two key developments. First a practical traffic signal re-timing method has been developed to leverage low-cost travel-time solutions such as Bluetooth and WiFi sniffing. Secondly, this research developed two deep learning based adaptive traffic signal control models and tested their performance across a wide variety of settings and configurations. The following sections highlight the key findings and contributions from each of these two areas of research.

6.1 SUMMARY OF RESEARCH FINDINGS

6.1.1 *Traffic Signal Re-timing Using Travel Time Data*

One of the major limitations with traditional signal timing management methods is caused by the limited availability of data that can be used to update signal timing plans on a regular basis. Signal re-timing is often done using a manual and labour intensive process where field volumes must first be obtained, then entered into software to produce signal timings that are then implemented in the field. One of the most practical ways to address this limitation is to take advantage of Big Data by leveraging sources such as Bluetooth and WiFi signals. These technologies have shown promise for use in monitoring applications; but limited studies have been conducted in the literature on other applications such as traffic management control.

In Chapter 3 a new traffic signal control model using travel time data as a basis was developed. The proposed model relies on the HCM's delay equations to estimate the degree of saturation (X) and makes adjustments to three major timing parameters: split times, cycle lengths, and offsets. The model implements these adjustments through three major steps: 1) the delay is estimated from travel time data collected between mid-block Bluetooth or WiFi detectors; 2) the degree of saturation and flow ratio (Y) is estimated from the delay data; 3) a new optimal cycle time is calculated; 4) the adjusted cycle times are then allocated to individual phases; 5) the volumes and patterns of vehicle movements between intersections are estimated; and finally 6) the offset is adjusted based on these predictions. An incremental approach is used to make these adjustments similar to how SCOOT operates where the model chooses at key intervals whether or not to make an adjustment to the timing plan.

Two separate case studies were conducted to evaluate the performance and sensitivity of the proposed model to measurement errors and system conditions. In the first approach a Monte Carlo simulation was developed to test the key parameters of the model and determine optimal settings. The Monte Carlo simulator evaluated the effect of critical parameters such as the optimisation step size and calibration parameters that control the how quickly the model responds to estimated demand changes. Particular focus was also given to the effect measurement errors would have on the decisions. The work led to the following key findings:

- The results of the analysis showed that situations with low volumes were particularly vulnerable to sub-optimal decisions. This vulnerability arises from the fact that lower volumes produce lower delays and thus errors of even 1 second can become significant. Volumes below 800 vph (near saturation) produced poor decisions when the standard deviation of the error ($\sigma(\epsilon)$) was 5 seconds. For the scenario where $\sigma(\epsilon)$ was set to 1 second, performance decreased when volumes dropped below 200 vph.

- Aggressive strategies with high step sizes and quick responses were also found to produce unreliable decisions for high values of $\sigma(\epsilon)$. In particular a *lag* in response was noted in changing volume situations when compared to the optimal decisions.

Following this, a microscopic traffic simulation analysis was conducted to evaluate the performance of the proposed signal updating method under realistic real-world settings with emulated emulation of Bluetooth and WiFi detectors; the study focused specifically on the performance of the proposed model and its sensitivity to some system parameters such as market penetration, measurement error, and traffic patterns. The main findings are summarised as follows:

- The results of the analysis showed that the most stable performance when compared to traditional actuated control came when optimisation frequency was kept at 3 cycles, with 4 cycles having the worst performance. Despite this, frequent optimisation was still found to perform well in many scenarios, though performance was often erratic. Good performance was still observed even in situations of low penetration rates.

- A key limitation of the approach was observed where queue spill-back can be improperly handled by the algorithm as by default the algorithm assumes that delays can be remedied by increasing the green time for the movement. In particular left turns can easily block through movements under highly saturated conditions, leading to a situation where extra time is allocated to through movement instead of the left turn movement.

6.2 DEEP REINFORCEMENT LEARNING FOR ADAPTIVE TRAFFIC SIGNAL CONTROL

In addition to the advantages Big Data brings, a number of advances in machine learning have opened up possibilities for real-time human-like decision making. Presently, many systems rely on rigid rule-based decisions. This research therefore attempted to address their limitations with an intelligent AI-based solution that applies a state-of-the-art machine learning technique, Deep Reinforcement Learning (DRL), and trains an agent to operate in an environment and to learn an optimal policy to operate in it. This work was divided into two major parts, an initial explorative study examining the sensitivity key parameters of a DRL system, including the identification of optimal parameters, on an isolated intersection, and an evaluation of a fully developed DRL platform on a real-world corridor.

6.2.1 *State Space Configurations*

In this research, the sensitivity of key parameters to the performance of the DRL agent was examined. These parameters include the state space representations and the model's ability to sense the environment, rewarding system and function used, transfer learning techniques, and multi-agent learning. The sensitivity analysis conducted tested a variety of input parameters for the model. The fundamental basis for the model's operation was queue length. Secondary inputs of time of day, day of the week, and current signal status were also provided to the model.

In terms of the state space and sensing abilities of the model, the following key results were observed:

- The effect of measurement sensitivity on queue lengths was tested by using queue length detections limited to 2, 10, 20, 40, and 80 vehicles. The algorithm's performance was statistically the same for cases with a queue detection range covering 20 or more vehicles. In the case of 10 vehicles, the model's policy was stable but performance was lower than other scenarios. When the detection range covers only two vehicles, however, the model did not converge to a stable policy. All other models converged to a stable policy after approximately 1 million frames of experience.

- Excessive training was found to create an over-fitting problem and average delays increased by at least 10% beyond 2 million frames for most models. Visual observation of the model highlighted that when queue detection is limited the state space is frequently indicating that vehicles are present and identifying patterns becomes difficult.

- The inclusion or removal of time-of-day and day-of-week information was found to have mixed results. In general training time was reduced slightly as more information could be leveraged, but the information was not sufficient on its own without queue information to provide a stable policy. This is likely due to limitations of the state space, which does not directly provide the amount of green that has been timed.

6.2.2 *Reward Functions*

Reward functions guide the model to an optimal policy by providing feedback on whether the state an agent is in is good or bad. In this research a few different rewarding systems and parameters were tested. As control is derived from queue length, these rewarding systems depend on queue length and vehicle discharges to reward or penalise an agent. The following key findings were observed:

- The most effective rewarding system was penalising the agent for each waiting vehicle behind a red light and rewarded it for each discharged vehicle. A sensitivity analysis was conducted to determine which parameters had the best performance. A reward of 5 for each discharged vehicle and -0.4 for each waiting vehicle was found to be the most effective. This corresponds to a reward-to-penalty ratio of 12.5 to 1. In addition, penalising the agent for causing split failures was also found to have a positive effect on training speed and the final performance of the model.
- An alternative reward framing that penalises the agent for queues behind red lights and rewards it for queues behind green lights was found to be ineffective.

6.2.3 *Training Strategies*

Training strategies can be used to reduce the amount of time an agent spends on exploring various state-action pairs, which can lead to faster convergence to an optimal policy and can prevent the agent from being trapped in a local-minimum. One of the most common strategies employed when training RL agents is called ϵ-greedy training. In ϵ-greedy, the agent exploits its learned knowledge with probability $1 - \epsilon$ or otherwise takes a random action. A variety of different strategies were explored, such as changing the length of the observation time, annealing (exploration) time. Additionally, changes to other parameters such as the learning rate and discount factor were also explored. An additional learning strategy that recognises that the *do nothing* action should be selected more frequently than other actions was explored. In this strategy if the agent chooses the *do nothing* action, a separate probability check is also made to determine if the action should be repeated and for how long. This analysis resulted in the following key findings:

- Middle-length observation periods of 5000 frames led to faster convergence than shorter or longer periods. The shorter annealing period of 20000 frames also led to to faster convergence. The longest settings for observation and annealing were also found to result in some over-fitting and sub-optimal performance and had higher final delays and queues than other settings.

- Learning rates other than the default value of 10^{-6} did not converge to optimal policies.
- A discount factor of 0.75 was associated with both better performance and faster convergence than the default of 0.99 and the lower value of 0.5. This value represents a compromise between prioritising immediate rewards while still maintaining some consideration for future rewards. In contrast, values of 0.99 were prone to inflating Q-values for state-action pairs.
- The alternative training strategy led to faster training times and convergences and was found to reduce frequent changes that were observed in ordinary ϵ-greedy training. The strategy reduced training times by approximately 50% when compared to baseline settings.

6.2.4 *Transfer Learning and Full Ring-and-Barrier Designs*

The sensitivity analysis was conducted using a two-phase controller, as the simplicity of the configuration leads to faster training times. A multi-action controller emulating full ring-and-barrier operation was also tested. A transfer-learning experiment was also conducted. In this experiment the model is first pre-trained using one simulator (SUMO) then fine-tuned on another (VISSIM). The difference in driver behaviour emulates the changes expected when pre-training a model on a simulation environment for application in the field. In this experiment the model is first pre-trained to a point just after convergence (approximately 1 million simulation frames). The simulation environment is then switched and ϵ is re-initialised and a short annealing is run. The results of this study revealed the following key findings:

- The added complexity from a full ring-and-barrier resulted in nearly four times the training time when compared to a two-phase design.
- Using transfer learning, convergence is much faster than a full retraining and within 75,000 simulation steps the model's performance re-converges to an optimal policy.
- Peak delays were 60% lower during training when using transfer learning when compared to retraining from scratch.

6.2.5 *DRL Model for Multi-Intersection and Corridor Control*

Multiple intersections introduce additional variables to the problem as the arrival pattern at an intersection is dependant on the actions of upstream agents. Two major approaches to multi-intersection control were evaluated. In the first, a multi-intersection control strategy was tested. In this strategy the isolated DRL controllers are applied on multiple intersections with each controller implementing an optimal solution locally. Then, in the second scenario, coordinated control was tested by extending the

DRL model's state space to include information about the time since the last green at upstream intersections, which is similar to the traditional *offset* parameter. Its performance was tested by comparing it against traditional actuated control and a SCOOT emulator. The SCOOT emulator was coded in reference to exiting literature on SCOOT.

A corridor based off University Avenue in Waterloo was coded in SUMO and used as a test network. The multi-intersection control strategy further extended the analysis done on transfer learning by pre-training models on a generic intersection and fine-tuning them on each of the target intersections. These intersections each have different volume patterns and geometries, including T intersections. Tests were run using two-phase and full ring-and-barrier controllers. The following key findings were observed:

- When compared to training all intersections simultaneously from scratch, the transfer learning scenario reduced training times by 50% for both the two-phase controllers and full ring-and-barrier controllers.
- In the full ring-and-barrier comparison, delays were reduced by over 6%. Side streets in particular saw significant delay reductions.

The state space of the DRL model was then extended to consider the benefits of including timing information from upstream intersections, and performance was compared against a SCOOT emulator and actuated signals. To test the dependency of the model on the timing information, the model was also run with no information by setting the time since the last upstream green to zero. The following key findings were observed:

- The DRL algorithm's performance resulted in delay reductions of approximately 7% network-wide and queue length reductions of approximately 11%.
- When compared to semi-actuated control, delays were reduced by over 20%.
- Performance was degraded by 40% when the time since the last upstream green was set to zero. This suggests that the model considers this offset information when making timing decisions.

6.3 MAJOR CONTRIBUTIONS

This research has advanced the state of the art in adaptive traffic signal control in three key areas: application of big data and novel data sources, adaptive signal control, and application of machine learning. In the literature, there has been limited research focussing on using alternative data sources and applying state-of-the-art deep learning techniques to optimise signals, such as travel time. This research has proposed new models and methods, conducted new analyses, and developed a novel platform.

6.3.1 *Contributions on Applying Travel Time for Signal Retiming*

Optimisations based on travel-time and delay have not been previously explored substantially in the literature. In particular, to the best of our knowledge, on-line signal optimisation has not been studied in any prior work. The results of this research has therefore advanced the state-of-the art by proposing and evaluating a new model that can be used to retime signals off-line or on-line in a adaptive-like configuration (Muresan, Fu, and Zhong, 2019). The work also contributed to the literature by examining and identifying the optimal settings, effects of confounding factors such as measurement error and market penetration, and comparing the method to existing technologies. The results of this research will be useful as Big Data technologies continue to see more use in industry.

6.3.2 *Contributions on Applying DRL for Adaptive Traffic Signal Control*

This research's work on DRL has advanced the state of the art in the application of machine learning for adaptive signal control. As an emerging topic, the idea of DRL for signal control has just begun to receive attention. The model developed for isolated intersections is one of the first to consider the idea of DRL, combining Q-learning and ordinary reinforcement learning.The contribution entails the development of a completely new model, including state space, action formulation, and reward function (Muresan, Fu, and Pan, 2018; Muresan and Fu, 2020; Muresan and Fu, 2021). The comprehensive sensitivity analysis done is among the first to be done in the area of deep learning and signal control and included significant findings on the sensitivity of the model to limited information, the optimal settings of parameters such as the learning rate, discount factor, learning periods, reward functions, and strategies to deploy the model to the field such as transfer learning (Muresan and Fu, Under Fourth Review).

Many of the existing studies on the application of deep learning for traffic signal control are constrained to simple models and application settings and the idea of multi-intersection and corridor-based control has been rarely explored. In particular, work on transfer learning and with full ring-and-barrier controllers is very limited in the literature. This research has advanced the state of the art by first studying how transfer learning can be used to simplify the deployment and training of models to multiple intersections, which has addressed the significant scalability problems and training challenges, and could be applied to scale the solution to networks of arbitrary configurations (Muresan and Fu, 2021). Secondly the proposed model included a complete action space with turning traffic and can be applied to intersections with complex phasing. Thirdly, the idea of providing information from adjacent intersections can be incorporated into the decision process is novel, which is simple but effective in solving the signal coordination problem. The proposed model is also competitive with the performance of existing state-of-the-art models such as SCOOT.

A DRL signal control platform was developed featuring the following key elements:

- A modular API system that consisted of a variety of changeable components, including the ability to interface with different simulators.
- An emulation of multi-device functionality through a multi-threaded design for each model being trained, including inter-thread communication to facilitate communication between different models.
- A ring-and-barrier style controller was also implemented with actions emulating typical behaviour present in modern systems such as protected-permissive left turn operation.

6.4 LIMITATIONS AND AREAS FOR FUTURE RESEARCH

In order to maintain the scope of this research, a number of factors were not explored explicitly, which requires further research, as discussed in the following sections.

6.4.1 *Travel-Time Based Signal Optimisation*

The performance of the travel-time based signal timing method proposed in this research could be affected by many factors such as data quality, intersection conditions, or traffic dynamics. One of the critical factors is the level of congestion. Although the effect of the degree of traffic congestion was studied to some extent, the impacts of over-saturated conditions was not explored extensively. Over-saturated conditions introduce a number of key challenges as the HCM equations begin to depend more heavily on the saturation flow rate and the evaluation time used. The random overflow component of the overall delay becomes more prominent, and the expected value of this can only be captured by repeated long-term observation. For real-time control (or signal adjustment), the observed delay may not be a true representation of the overall traffic conditions due to random variation. Furthermore, the evaluation time is particularly critical and although a fixed value was used the evaluation time should ideally track how long the intersection has been in an over-saturated condition. Finally, the method used to estimate the delay requires additional considerations for over-saturated conditions and in over-saturated conditions vehicles may queue outside the detection range of detectors.

To properly serve near and over-saturated movements, an estimation of the saturation flow-rate is required as the HCM overflow delay (d_2) becomes important and requires the saturation flow when calculating the movement's capacity. This complicates the application of this method to permissive movements like left and right turns as the HCM process would need to be followed to provide an estimate of the adjusted saturation flow and capacity. The sensitivity of this algorithm's performance to the estimated saturation flow rate was not studied and should be considered in future work. Furthermore some other assumptions, such as the distribution of traffic between lanes and shared lanes may also have significant impacts particularly on the d_2 value as their impacts on saturation flow were not extensively examined. Although the test cases did result in over-saturated conditions on some cases, the impact of over-saturated movements was not thoroughly studied.

Another critical limitation is the assumption that the delays observed are caused solely by the amount of green time displayed to the movement. This was observed to cause problems particularly on protected left turn movements where an over-saturated left turn movement would block the through lanes resulting in higher delays for the through movement. The controller would then increase both the through and left turn greens rather than just the left turn green resulting in sub-optimal control.

The progression factor in the delay equations 2.14 was set in reference to guideline values in the CCG, however it should be noted that these values may not be suitable for all cases, and particularly as the offset improves. Exact specification of this factor is integral to coordinated settings as uniform delay will decrease if vehicles are more likely to arrive on green. This problem requires further exploration that was not covered by this research.

For permissive left turn movements, the HCM typically defines a way by which volumes and delays can be estimated by considering the number of vehicles that would discharge in the permissive movement. This research estimated the contribution of both the permissive and protected green portions to the saturation flow rate and considered the entire green time for the left turn movement, however this method could use additional validation and verification as on some approaches large queues formed for the left turn movement.

Due to design limitations, the algorithm's timing plans were also implemented through fixed-time controllers. However, this means that the actuated controllers have some benefits in terms of their operation when compared to the algorithm, including the ability to gap-out the actuated protected left movements. By adjusting the maximum green times, however, it is possible that actuated controllers could also be improved through this algorithm.

While the approach tested needs only per-movement delay data, the primary implementation examined in simulation relies on Bluetooth and WiFi detectors installed at midblock locations. This deployment may not be feasible in all cases and further exploration is required to determine if other configurations work.

This work focussed on the problem of re-timing signals using travel time data and the impacts of measurement errors. While some challenges pertaining to the use of real-world Bluetooth or WiFi data are addressed, some simplifications are made. In particular, issues surrounding detection frequency and detection range, filtering out non-vehicles, and duplicate detections were not explored extensively. While some of these issues could be controlled by changing the parameters of the device scanning (e.g. increasing or decreasing receiver power) proper configuration would likely require local calibration. Further, the k-means algorithm used to categorise the data, while simple, is likely not the most effective, especially under over-saturated conditions. The effect of these on the estimation error has seen limited consideration in the literature and further work in this area would likely be needed to successfully estimate delay from travel time in all situations. However, this may not be an issue in some application settings such as rural areas or areas where travel-time data can come from alternative sources such as connected vehicles.

6.4.2 *Deep Reinforcement Learning (DRL) for Adaptive Traffic Signal Control*

The deep learning-based traffic signal control models proposed in this research focusses specifically on the application of deep reinforcement learning (DRL) and with its specific implementation that relies on Deep Q Networks. In recent years other techniques such as Actor-Critic designs have also shown great promise and could be explored in future research for the potential applications to traffic signal control. Some preliminary work was done in this area and future work would be useful in exploring if other model structures may yield better performance. These model structures may also require modifications to the action space, state space, and reward system.

This thesis research work has also focussed specifically on using queue length as the primary observable variable to represent the system state. WHile modern sensor technology such as video cameras or LiDAR are increasingly making queue length detection feasible, it remains not as easily obtainable as other data, such as flow profiles at specific locations from loop detectors or zone-based video detection systems. These traffic flow data have been used as the major input for many traditional adaptive traffic control systems. How they can be used, either solely or in combination with queue data in this proposed DRL framework and what the potential value of improving the performance of the proposed system is still an open question to be addressed in future research.

Another limitation of our research is the case study comparing the performance of out proposed system to the state-of-the-art adaptive traffic control method – SCOOT. A SCOOT emulator was coded, which may not fully reflect the performance of SCOOT in practical field settings. Furthermore, SCOOT is a proprietary system and while the methods used are available for analysis in literature, it is not possible to fully replicate all functions. Future research should look at hardware-in-the-loop simulations as the next step to evaluating the feasibility of AI systems.

This work has also assumed that detection of queue lengths and discharging vehicles is possible and these data can be reliably measured at per-second intervals. Real world implementations will likely involve some form of measurement error and erroneous inputs that may need to be handled by the model, but further research is required to determine what effect these would have.

The evaluation of the proposed control system is also empirical in nature, which means the findings and conclusions are limited to the cases and scenarios being considered. Future research should explore its performance over a wider range of network configurations, traffic variations and control constraints. It would also be interesting to explore the feasibility of applying the proposed model for area-wide control of signalized intersections.

BIBLIOGRAPHY

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. (2016). "Tensorflow: A system for large-scale machine learning." In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283.

Abbott-Jard, Michael, Harpal Shah, and Ashish Bhaskar (2013). "Empirical evaluation of Bluetooth and Wifi scanning for road transport." In: *36th Australasian Transport Research Forum (ATRF)*. Vol. 2. 4.

Abdulhai, Baher, Rob Pringle, and Grigoris J Karakoulas (2003). "Reinforcement learning for true adaptive traffic signal control." In: *Journal of Transportation Engineering* 129.3, pp. 278–285.

Abedi, Naeim, Ashish Bhaskar, and Edward Chung (2013). "Bluetooth and Wi-Fi MAC address based crowd data collection and monitoring: benefits, challenges and enhancement." In:

Adams, William Frederick (1937). *Road traffic considered as a random series*. Institution of Civil Engineers.

Agarap, Abien Fred (2018). "Deep learning using rectified linear units (relu)." In: *arXiv preprint arXiv:1803.08375*.

Arel, Itamar, Cong Liu, T Urbanik, and AG Kohls (2010). "Reinforcement learning-based multi-agent system for network traffic signal control." In: *IET Intelligent Transport Systems* 4.2, pp. 128–135.

Astrom, Karl J (1965). "Optimal control of Markov processes with incomplete state information." In: *Journal of mathematical analysis and applications* 10.1, pp. 174–205.

Barth, Dave (Aug. 2009). *The bright side of sitting in traffic: Crowdsourcing road congestion data*. Google Maps.

Bas, Erhan, A Murat Tekalp, and F Sibel Salman (2007). "Automatic vehicle counting from video for traffic flow analysis." In: *Intelligent Vehicles Symposium, 2007 IEEE*. Ieee, pp. 392–397.

Basheer, Imad A and Maha Hajmeer (2000). "Artificial neural networks: fundamentals, computing, design, and application." In: *Journal of microbiological methods* 43.1, pp. 3–31.

Beess, KensrnN G (1988). "Analysis of platoon dispersion with respect to traffic volume." In: *Trffic Flow Theory ønd* 1, p. 64.

Bekker, René, GM Koole, Bo Friis Nielsen, and Thomas Bang Nielsen (2011). "Queues with waiting time dependent service." In: *Queueing systems* 68.1, pp. 61–78.

Bonneson, JA, MP Pratt, and MA Vanderhey (2008). "Predicting the Performance of Automobile Traffic on Urban Streets. Final Report, NCHRP 3-79." In: *Transportation Research Board*.

Centennial FDOT (May 2016). *Advanced Signal Control Technology*.

Chakraborty, Goutam, Kshirasagar Naik, Debasish Chakraborty, Norio Shiratori, and David Wei (2010). "Analysis of the Bluetooth device discovery protocol." In: *Wireless Networks* 16.2, pp. 421–436.

Chien, Steven I, Kitae Kim, and Janice Daniel (2006). "Cost and Benefit Analysis for Optimized Signal Timing-Case Study: New Jersey Route 23." In: *Institute of Transportation Engineers. ITE Journal* 76.10, p. 37.

Chiu, Stephen and Sujeet Chand (1993). "Adaptive traffic signal control using fuzzy logic." In: *Fuzzy Systems, 1993., Second IEEE International Conference on*. IEEE, pp. 1371–1376.

Chollet, François (2017). "Xception: Deep learning with depthwise separable convolutions." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258.

Chronopoulou, Alexandra, Christos Baziotis, and Alexandros Potamianos (2019). "An embarrassingly simple approach for transfer learning from pretrained language models." In: *arXiv preprint arXiv:1902.10547*.

City of Calgary (2017). *City of Calgary: Bluetooth travel time system.*

City of Toronto ITS Operations (2016). URL: `http://www1.toronto.ca/wps/portal/contentonly?vgnextoid=af8cdee2b80a4510VgnVCM10000071d60f89RCRD`.

City of Toronto (Jan. 2012). *Drawings for Traffic Control Devices.*

— (2017). *City of Toronto: Traffic Signals.* URL: `http://www1.toronto.ca/wps/portal/contentonly?vgnextoid=9452722c231ec410VgnVCM10000071d60f89RCRD`.

Da Silva, Felipe Leno and Anna Helena Reali Costa (2019). "A survey on transfer learning for multiagent reinforcement learning systems." In: *Journal of Artificial Intelligence Research* 64, pp. 645–703.

Daganzo, Carlos F (1994). "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory." In: *Transportation Research Part B: Methodological* 28.4, pp. 269–287.

Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. (2012). "Large scale distributed deep networks." In: *Advances in neural information processing systems*, pp. 1223–1231.

Dresner, Kurt and Peter Stone (2008). "A multiagent approach to autonomous intersection management." In: *Journal of artificial intelligence research* 31, pp. 591–656.

El-Tantawy, Samah, Baher Abdulhai, and Hossam Abdelgawad (2013). "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto." In: *IEEE Transactions on Intelligent Transportation Systems* 14.3, pp. 1140–1150.

Federal Highway Administration (2011). *EDC-1: Adaptive Signal Control Technology.* Centre for Acceleratiing Innovation. URL: `https://www.fhwa.dot.gov/innovation/everydaycounts/edc-1/asct.cfm`.

— (Apr. 2017a). *Leveraging the Promise of Connected and Autonomous Vehicles to Improve Integrated Corridor Management and Operations: A Primer - FHWA Operations.*

— (2017b). *Traffic Signal Timing Manual.* URL: `https://ops.fhwa.dot.gov/publications/fhwahop08024/index.htm`.

Fellendorf, Martin and Peter Vortisch (2000). "Integrated modeling of transport demand, route choice, traffic flow and traffic emissions." In: *79th Annual Meeting of the Transportation Research Board, Washington, DC*.

Feng, Yiheng, K Larry Head, Shayan Khoshmagham, and Mehdi Zamanipour (2015). "A real-time adaptive signal control in a connected vehicle environment." In: *Transportation Research Part C: Emerging Technologies* 55, pp. 460–473.

Feng, Yiheng, Jianfeng Zheng, and Henry X Liu (2018). "Real-time detector-free adaptive signal control with low penetration of connected vehicles." In: *Transportation Research Record* 2672.18, pp. 35–44.

Fontaine, Michael D., Jiqqi Ma, and Jia Hu (June 2015). *Evaluation of the Virginia Department of Transportation Adaptive Signal Control Technology Pilot Project*. Report No. VCTIR 15-R24. Rhythm Engineering.

Freudiger, Julien (2015). "How talkative is your mobile device? An experimental study of Wi-Fi probe requests." In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pp. 1–6.

Friedrich, Bernhard (2002). "Adaptive signal control: an overview." In: *13th Mini Euro Conference–Handling Uncertainty in the Analysis of Traffic and Transportation systems*.

Genders, Wade and Saiedeh Razavi (2016). "Using a Deep Reinforcement Learning Agent for Traffic Signal Control." In:

Goodall, Noah, Brian Smith, and Byungkyu Park (2013). "Traffic signal control with connected vehicles." In: *Transportation Research Record: Journal of the Transportation Research Board* 2381, pp. 65–72.

Google Deep Mind (2017). *DeepMind*. URL: `https://deepmind.com/research/alphago/`.

Gordon, R.L., National Research Council (U.S.). Transportation Research Board, National Cooperative Highway Research Program, American Association of State Highway, Transportation Officials, and United States. Federal Highway Administration (2010). *Traffic Signal Retiming Practices in the United States*. Nchrp Synthesis. Transportation Research Board. ISBN: 9780309143172.

Gordon, Robert L (2003). *Systems Engineering Processes for Developing Traffic Signal Systems*. Vol. 307. Transportation Research Board.

Group, PTV et al. (2014). "PTV Vissim 7 User Manual." In: *Karlsruhe, Germany*.

Guler, S Ilgin, Monica Menendez, and Linus Meier (2014). "Using connected vehicle technology to improve the efficiency of intersections." In: *Transportation Research Part C: Emerging Technologies* 46, pp. 121–131.

HDR Corporation (Dec. 2008). *Costs of Road Congestion in the Greater Toronto and Hamilton Area*.

Haghani, Ali, Masoud Hamedi, and Kaveh Farokhi Sadabadi (2009). *I-95 Corridor coalition vehicle probe project: Validation of INRIX data*.

Haghani, Ali, Masoud Hamedi, Kaveh Sadabadi, Stanley Young, and Philip Tarnoff (2010). "Data collection of freeway travel time ground truth with bluetooth sensors." In: *Transportation Research Record: Journal of the Transportation Research Board* 2160, pp. 60–68.

Hamerly, G and C Elkan (2003). *Learning the K in K-means Advances in Neural Information Processing Systems*.

Hart-Bishop, J, B Hellinga, and A Zarinbal (2016). "Advanced Traffic Signal Control Using Bluetooth/Wi-Fi Detectors." In: *Kelowna 2016-CITE Annual Meeting and Conference-Technical Compendium*.

Hartigan, John A (1975). *Clustering algorithms*. John Wiley & Sons, Inc.

Hartigan, John A and Manchek A Wong (1979). "Algorithm AS 136: A k-means clustering algorithm." In: *Journal of the royal statistical society. series c (applied statistics)* 28.1, pp. 100–108.

Higgs, Bryan, M Abbas, and Alejandra Medina (2011). "Analysis of the Wiedemann car following model over different speeds using naturalistic data." In: *Procedia of RSS Conference*, pp. 1–22.

Hinton, Geoffrey E, Simon Osindero, and Yee-Whye Teh (2006). "A fast learning algorithm for deep belief nets." In: *Neural computation* 18.7, pp. 1527–1554.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks." In: *science* 313.5786, pp. 504–507.

Hoey, Jesse and Pascal Poupart (2005). "Solving POMDPs with continuous or large discrete observation spaces." In: *IJCAI*, pp. 1332–1338.

Hoogendoorn, Serge, Raymond G Hoogendoorn, and Winnie Daamen (2011). "Wiedemann revisited: new trajectory filtering technique and its implications for car-following modeling." In: *Transportation research record* 2260.1, pp. 152–162.

Hunt, PB, DI Robertson, RD Bretherton, and M Cr Royle (1982). "The SCOOT on-line traffic signal optimisation technique." In: *Traffic Engineering & Control* 23.4.

INRIX Traffic (2017). URL: `http://inrix.com/about/`.

Jiang, Zhengyao, Dixing Xu, and Jinjun Liang (2017). "A deep reinforcement learning framework for the financial portfolio management problem." In: *arXiv preprint arXiv:1706.10059*.

Jin, Xuexiang, Yi Zhang, Fa Wang, Li Li, Danya Yao, Yuelong Su, and Zheng Wei (2009). "Departure headways at signalized intersections: A log-normal distribution model approach." In: *Transportation research part C: emerging technologies* 17.3, pp. 318–327.

Kamarajugadda, Anil and Byungkyu Park (2003). *Stochastic traffic signal timing optimization*. Tech. rep. Center for Transportation Studies, University of Virginia.

Kergaye, Cameron, Aleksandar Stevanovic, and Peter T Martin (2008). "An evaluation of SCOOT and SCATS through microsimulation." In: *International Conference on Application of Advanced Technologies in Transportation, Transportation and Development Institute, Athens, Greece*.

Ketchen, David J and Christopher L Shook (1996). "The application of cluster analysis in strategic management research: an analysis and critique." In: *Strategic management journal* 17.6, pp. 441–458.

Kim, Kitae, Dennis Motiani, Lazar N Spasovic, Branislav Dimitrijevic, and Steven Chien (2014). "Assessment of Speed Information Based on Probe Vehicle Data: A Case Study in New Jersey." In: *Transportation Research Board 93rd Annual Meeting*. 14-4464.

Kim, Seoungbum and Benjamin Coifman (2014). "Comparing INRIX speed data against concurrent loop detector stations over several months." In: *Transportation Research Part C: Emerging Technologies* 49, pp. 59–72.

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization.” In: *arXiv preprint arXiv:1412.6980*.

Koerner, Ralph J (Nov. 1976). *Inductive loop vehicle detector*. US Patent 3,989,932.

Kotu, Vijay and Bala Deshpande (2019). “Chapter 7 - Clustering.” In: *Data Science (Second Edition)*. Ed. by Vijay Kotu and Bala Deshpande. Second Edition. Morgan Kaufmann, pp. 221 –261. ISBN: 978-0-12-814761-0. DOI: `https://doi.org/10.1016/B978-0-12-814761-0.00007-1`. URL: `http://www.sciencedirect.com/science/article/pii/B9780128147610000071`.

Krajzewicz, Daniel, Jakob Erdmann, Michael Behrisch, and Laura Bieker (2012). “Recent development and applications of SUMO-Simulation of Urban MObility.” In: *International journal on advances in systems and measurements* 5.3&4.

Krauß, Stefan (1998). “Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics.” PhD thesis.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks.” In: *Advances in neural information processing systems*, pp. 1097–1105.

Laney, Doug (2001). “3D data management: Controlling data volume, velocity and variety.” In: *META Group Research Note* 6, p. 70.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). “Deep learning.” In: *nature* 521.7553, pp. 436–444.

Li, Li, Yisheng Lv, and Fei-Yue Wang (2016). “Traffic signal timing via deep reinforcement learning.” In: *IEEE/CAA Journal of Automatica Sinica* 3.3, pp. 247–254.

Lighthill, Michael James and Gerald Beresford Whitham (1955). “On kinematic waves II. A theory of traffic flow on long crowded roads.” In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 229.1178, pp. 317–345.

Lin, Wei-Hua and Chenghong Wang (2004). “An enhanced 0-1 mixed-integer LP formulation for traffic signal control.” In: *IEEE Transactions on Intelligent transportation systems* 5.4, pp. 238–245.

Lo, Hong K (1999). “A novel traffic signal control formulation.” In: *Transportation Research Part A: Policy and Practice* 33.6, pp. 433–448.

MacQueen, James et al. (1967). “Some methods for classification and analysis of multivariate observations.” In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA, pp. 281–297.

Malinovskiy, Yegor, Nicolas Saunier, and Yinhai Wang (2012). “Analysis of pedestrian travel with static bluetooth sensors.” In: *Transportation Research Record: Journal of the Transportation Research Board* 2299, pp. 137–149.

Manar, Abdelaziz and Karsten G Baass (1996). “Traffic platoon dispersion modeling on arterial streets.” In: *Transportation Research Record* 1566.1, pp. 49–53.

Manual, Highway Capacity (2010). “HCM2010.” In: *Transportation Research Board, National Research Council, Washington, DC*.

Mardia, KV (1979). “JT Kent. and J. M. Bibby.” In: *Multivariate Analysis*.

Miller, Alan J (1963). “A COMPUTER CONTROL SYSTEM FOR TRAFFIC NETWORKS.” In:

Miovision (2017). *Traffic Data Collection and Signal Operations - Miovision*. URL: `https://miovision.com/`.

Mirchandani, Pitu B and David E Lucas (2001). *RHODES-ITMS Tempe field test project: Implementation and field testing of RHODES, a real-time traffic adaptive control system*. Tech. rep.

Mirchandani, Pitu and Larry Head (2001). "A real-time traffic signal control system: architecture, algorithms, and analysis." In: *Transportation Research Part C: Emerging Technologies* 9.6, pp. 415–432.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. (2015). "Human-level control through deep reinforcement learning." In: *nature* 518.7540, pp. 529–533.

Moghaddam, Soroush and Bruce Hellinga (2013). "Quantifying measurement error in arterial travel times measured by bluetooth detectors." In: *Transportation Research Record: Journal of the Transportation Research Board* 2395, pp. 111–122.

— (2014). "Real-time prediction of arterial roadway travel times using data collected by bluetooth detectors." In: *Transportation Research Record: Journal of the Transportation Research Board* 2442, pp. 117–128.

Moore Ii, James E, Stephen P Mattingly, C Arthur MacCarley, and Michael G McNally (2005). "Anaheim advanced traffic control system field operations test: A technical evaluation of SCOOT." In: *Transportation planning and technology* 28.6, pp. 465–482.

Mueller, Edward A (1970). "Aspects of the history of traffic signals." In: *IEEE Transactions on Vehicular Technology* 19.1, pp. 6–17.

Muresan, Matthew and Guangyuan Panand Liping Fu (Under Fourth Review). "Deep Reinforcement Learning for Adaptive Traffic Signal Control: Alternative Implementations, System Stability and Optimal Settings." In: *IEEE Transactions on Intelligent Transportation Systems*.

Muresan, Matthew and Liping Fu (2020). "Multi-Intersection Control With Deep Reinforcement Learning And Ring-And-Barrier Controllers." In: *99th Annual Meeting of the Transportation Research Board*.

— (2021). "Multi-Intersection Control With Deep Reinforcement Learning And Ring-And-Barrier Controllers." In: *Transportation Research Record*.

Muresan, Matthew, Liping Fu, and Guangyuan Pan (2018). "Adaptive Traffic Signal Control with Deep Reinforcement Learning: An Exploratory Investigation." In: *97th Annual Meeting of the Transportation Research Board*.

Muresan, Matthew, Liping Fu, and Ming Zhong (2019). "Continuous Updating of Traffic Signal Timing Plans Using Bluetooth and Wifi Data." In: *98th Annual Meeting of the Transportation Research Board*.

Pan, Guangyuan, Liping Fu, Ruifan Yu, and Matthew Iulian Muresan (2018). *Winter road surface condition recognition using a pre-trained deep convolutional neural network*. Tech. rep.

Pan, Sinno Jialin and Qiang Yang (2009). "A survey on transfer learning." In: *IEEE Transactions on knowledge and data engineering* 22.10, pp. 1345–1359.

Pelleg, Dan, Andrew W Moore, et al. (2000). "X-means: Extending k-means with efficient estimation of the number of clusters." In: *Icml*. Vol. 1, pp. 727–734.

Pew Research Center (2015). *U.S. Smartphone Use in 2015*. URL: `http://assets.pewresearch.org/wp-content/uploads/sites/14/2015/03/PI_Smartphones_0401151.pdf`.

— (2016). *Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies*. URL: `http://www.pewglobal.org/files/2016/02/pew_research_center_global_technology_report_final_february_22__2016.pdf`.

Phua, Peilin, Bill Page, and Svetlana Bogomolova (2015). "Validating Bluetooth logging as metric for shopper behaviour studies." In: *Journal of Retailing and Consumer Services* 22, pp. 158–163.

Rao, Yongming, Jiwen Lu, and Jie Zhou (2017). "Attention-aware deep reinforcement learning for video face recognition." In: *Proceedings of the IEEE international conference on computer vision*, pp. 3931–3940.

Raphael, Jeffery (2018). "An Exploration of Traffic Signal Control using Multi-agent Market-based Mechanisms." PhD thesis. University of Liverpool.

Ratrout, Nedal T and Imran Reza (2014). "Comparison of optimal signal plans by Synchro & TRANSYT-7F using PARAMICS–A case study." In: *Procedia Computer Science* 32, pp. 372–379.

Rhythm Engineering (2017). *How InSyncs Real-Time Adaptive Traffic Control Works*.

Richards, Paul I (1956). "Shock waves on the highway." In: *Operations research* 4.1, pp. 42–51.

Robertson, Dennis I (1969). "TRANSYT: a traffic network study tool." In:

— (1986). "Research on the TRANSYT and SCOOT Methods of Signal Coordination." In: *ITE journal* 56.1, pp. 36–40.

Robertson, Dennis I and R David Bretherton (1991). "Optimizing networks of traffic signals in real time-the SCOOT method." In: *IEEE Transactions on vehicular technology* 40.1, pp. 11–15.

Rodriguez-Ramos, Alejandro, Carlos Sampedro, Hriday Bavle, Paloma De La Puente, and Pascual Campoy (2019). "A deep reinforcement learning strategy for UAV autonomous landing on a moving platform." In: *Journal of Intelligent & Robotic Systems* 93.1-2, pp. 351–366.

Rosenberg, Chuck (2013). *Improving Photo Search: A Step Across the Semantic Gap*. Google. URL: `https://research.googleblog.com/2013/06/improving-photo-search-step-across.html`.

SCATS (2000). *An Introduction To The New Generation Scats 6*. URL: `http://www.scats.com.au/files/an_introduction_to_scats_6.pdf`.

SCOOT Systems (2014). *How Scoot Works*. URL: `http://www.scoot-utc.com/DetailedHowSCOOTWorks.php?menu=Technical`.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview." In: *Neural networks* 61, pp. 85–117.

Shen, Luou, Ronghui Liu, Zhihong Yao, Weitiao Wu, and Hongtai Yang (2018). "Development of dynamic platoon dispersion models for predictive traffic signal control." In: *IEEE Transactions on Intelligent Transportation Systems* 20.2, pp. 431–440.

Siemens Mobility, Traffic Solutions (2016). *SCOOT User Guide*. Tech. rep. Siemens PLC.

Sims, Arthur G and Kenneth W Dobinson (1980). "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits." In: *IEEE Transactions on vehicular technology* 29.2, pp. 130–137.

Statistics Canada (2017). *The Internet and Digital Technology, 2017032*. URL: `https://www150.statcan.gc.ca/n1/pub/11-627-m/11-627-m2017032-eng.pdf`.

Stevanovic, Aleksandar, Cameron Kergaye, and Peter T Martin (2008). "Field evaluation of SCATS traffic control in Park City, UT." In: *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting*.

— (2009). "Scoot and scats: A closer look into their operations." In: *88th Annual Meeting of the Transportation Research Board. Washington DC*.

Szepesvari, Csaba (2010). "Algorithms for reinforcement learning: Synthesis lectures on artificial intelligence and machine learning." In: *Morgan and Claypool*.

Tai, Lei and Ming Liu (2016). "Mobile robots exploration through cnn-based reinforcement learning." In: *Robotics and biomimetics* 3.1, pp. 1–8.

Teply, S, DI Allingham, DB Richardson, and BW Stephenson (2008). "Canadian capacity guide for signalized intersections." In:

TomTom (2017). *About TomTom Traffic*. URL: `https://www.tomtom.com/en_gb/traffic-news/traffic-incidents`.

Tong, Huijiao, Leiji Zhu, Yong Xiong, and Wei Yao (2017). "Modeling large passenger flow safety by simulation and testing." In: *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*. IEEE, pp. 235–239.

Tong, Yue, Lei Zhao, Li Li, and Yi Zhang (2015). "Stochastic programming model for oversaturated intersection signal timing." In: *Transportation Research Part C: Emerging Technologies* 58, pp. 474–486.

Transport Canada (Sept. 2016). *ACTIVE-AURORA project: wireless connected vehicle technology now testing a variety of applications*.

Transport Canada, Delcan, iTRANS, and ADEC (Apr. 2006). *The Cost of Urban Congestion in Canada*. Tech. rep. Transport Canada.

Tyack, FG (1938). "Street traffic signals, with particular reference to vehicle actuation." In: *Journal of the Institution of Electrical Engineers* 82.494, pp. 125–154.

US Department of Transportation (2007). *Intelligent Transportation Systems for Traffic Signal Control: FHWA-JPO-07-004*. Tech. rep. URL: `https://ntl.bts.gov/lib/jpodocs/brochure/14321_files/a1019-tsc_digital_n3.pdf`.

University of Alberta (Sept. 2016). *The CST and the First Connected Vehicle Test Bed in Canada*.

Van Houdt, B and C Blondia (2005). "Approximated transient queue length and waiting time distributions via steady state analysis." In: *Stochastic Models* 21.2-3, pp. 725–744.

Wallace, Charles E, KG Courage, DP Reaves, GW Schoene, and GW Euler (1984). *TRANSYT-7F user's manual*. Tech. rep.

Wang, Fengzi, Xinning Zhu, and Jiansong Miao (2017). "Semantic trajectories-based social relationships discovery using WiFi monitors." In: *Personal and Ubiquitous Computing* 21.1, pp. 85–96.

Wasson Jason S., P.E., P.E. Sturdevant James R., and P.E. Bullock Darcy M. (June 2008). "Real-Time Travel Time Estimates Using Media Access Control Address Matching." English. In: *Institute of Transportation Engineers.ITE Journal* 78.6, pp. 20–23.

Watkins, Christopher JCH and Peter Dayan (1992). "Q-Learning." In: *Machine learning* 8.3-4, pp. 279–292.

Wey, Wann-Ming (2000). "Model formulation and solution algorithm of traffic signal control in an urban network." In: *Computers, environment and urban systems* 24.4, pp. 355–378.

Wilson, Christopher Joseph, Gareth Millar, and Roderick Tudge (2006). "Microsimulation Evaluation of Benefits of SCATS-Coordinated Traffic Control Signals." In: *Transportation Research Board 85th Annual Meeting*. 06-1984.

Yang, Zhen, Yiheng Feng, and Henry X. Liu (2021). "A cooperative driving framework for urban arterials in mixed traffic conditions." In: *Transportation Research Part C: Emerging Technologies* 124, p. 102918. ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2020.102918. URL: https://www.sciencedirect.com/science/article/pii/S0968090X20308172.

Yau, Kok-Lim Alvin, Junaid Qadir, Hooi Ling Khoo, Mee Hong Ling, and Peter Komisarczuk (2017). "A survey on reinforcement learning models and algorithms for traffic signal control." In: *ACM Computing Surveys (CSUR)* 50.3, p. 34.

Yoshimura, Yuji, Anne Krebs, and Carlo Ratti (2016). "An analysis of visitors' length of stay through noninvasive Bluetooth monitoring in the Louvre Museum." In:

Zanin, Michele, Stefano Messelodi, and Carla Maria Modena (2003). "An efficient vehicle queue detection system based on image processing." In: *Image Analysis and Processing, 2003. Proceedings. 12th International Conference on*. IEEE, pp. 232–237.

Zhao, Yi and Zong Tian (2012). "An overview of the usage of adaptive signal control system in the United States of America." In: *Applied Mechanics and Materials*. Vol. 178. Trans Tech Publ, pp. 2591–2598.

Zhu, Yuke, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi (2017). "Target-driven visual navigation in indoor scenes using deep reinforcement learning." In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3357–3364.

Part IV

APPENDIX

A CODE IMPLEMENTATIONS

A.1 MAIN

A high-level overview of the components of the platform developed is shown in Figure A.1 below. A main script is used to initialise and control all parts of the model training or evaluation process. The platform was developed in Python and uses an object-oriented approach to programming where functions, variables, and other aspects are grouped together into related containers called *objects*. Functions and variables can then be accessed from these objects (in Python this is done through the dot operator, e.g. *object.function(arguments)*). An overview of each piece is provided in the paragraphs and sections below.

The script begins by loading the simulator and API object to interact with it. Two APIs were developed to interact with SUMO and VISSIM, but additional APIs implementing the methods discussed in Section A.2 could be added. Then the algorithm loads the selected signal control method (as the *learning model*). This includes the DRL model developed, SCOOT, or traditional fixed time or actuated control.

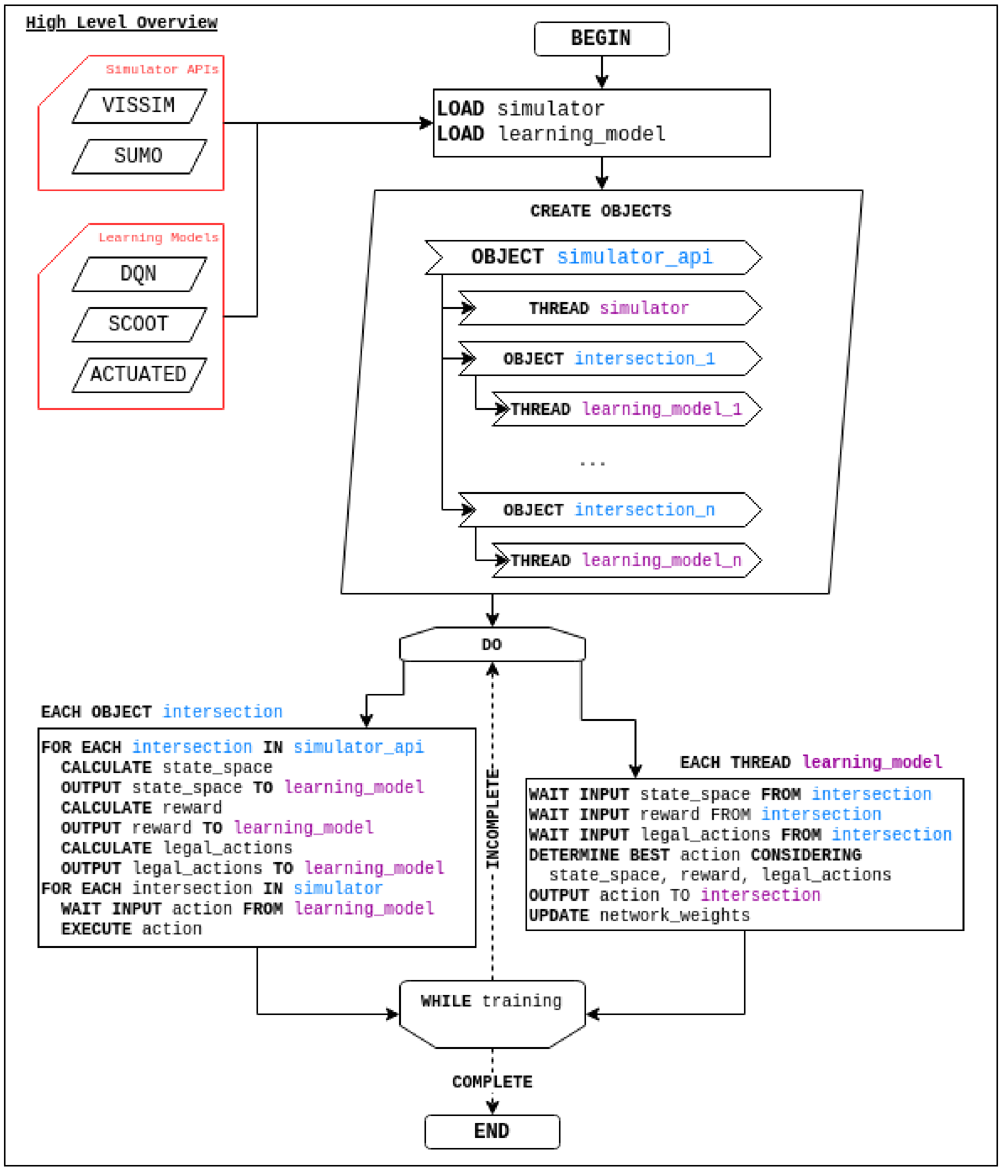
The loaded modules each create objects that can be interacted with, and each intersection's model is run as a separate thread. The process of learning or signal control is then done. In some cases the calculations are skipped if not relevant to the learning model (e.g. the Actuated and SCOOT models don't request the *reward* value). Since the simulation itself is controlled in the main thread, synchronisation is required to ensure that all intersections are learning at the correct rate and receiving frames at the same time.

A.2 SIMULATOR API

The simulator API is an abstraction layer that translates inputs and outputs from the *main* and *learning model* into commands understood by the relevant simulator. An object-oriented style was used to develop the API and two Two abstraction layers were developed for this project, one for SUMO and one for VISSIM.

Abstraction layers run the simulator as an external thread and implement the following functions:

- *A reset function*. This function returns the simulation to an empty state. It also re-generates all the volumes (if needed) from a new random seed. After re-setting the network, it also runs the simulation for a pre-defined warm-up period.
- *An advance function*. This function advances the simulation and then collects all the data required for input from the simulator. This function is usually the



```
FOR EACH intersection IN simulator_api
  CALCULATE state_space
  OUTPUT state_space TO learning_model
  CALCULATE reward
  OUTPUT reward TO learning_model
  CALCULATE legal_actions
  OUTPUT legal_actions TO learning_model
FOR EACH intersection IN simulator
  WAIT INPUT action FROM learning_model
  EXECUTE action
```

```
WAIT INPUT state_space FROM intersection
WAIT INPUT reward FROM intersection
WAIT INPUT legal_actions FROM intersection
DETERMINE BEST action CONSIDERING
  state_space, reward, legal_actions
OUTPUT action TO intersection
UPDATE network_weights
```

Figure A.1: SUMO Simulated Corridor

most time-intensive operation in a time step (particularly with VISSIM). Queue lengths are estimated directly from vehicle locations and speeds. This function also calls a number of other private functions that interact with the Network Performance object to update data that is used in the DRL and SCOOT models.

- *An action function*. This function implements the actions described in Section 4.3.1.2. It interacts with a separate Ring and Barrier abstraction object that controls and keeps track of all the phases
- *A permitted actions function*. This function checks which actions are valid at any given time based on the timing rules in the Ring and Barrier abstraction.

Actions that implement with the simulator translate these calls for each specific simulator.

To support the abstraction layer, there are a number of critical objects that are also used to abstract part of the simulation's operation. These include the following:

- *A network performance* object that keeps track of high-level data, including the current step, the number of runs, and queue lengths. This object directly keeps track of key elements of the reward function used in the DRL model, including the number of vehicles discharged and the current queue length. This object is usually included as a sub-object of the Network Data
- *A network data* object that has functions that keeps track of raw data, such as raw vehicle positions, signals, and links. This object also has functions that allow conversion of the raw data, such as functions to estimate the queue length for a particular link used in the Network Performance object
- *A signal head* object that keeps track of the current state of each phase's signal (e.g. whether it's green, red, etc).
- *A ring-and-barrier* object that emulates ring-and-barrier operation. This object includes tracking for the current phase, plan, minimum green, maximum green, etc.

A.3 BLUETOOTH AND WIFI PLATFORM

The Bluetooth and WiFi simulator consists of two different platforms. The Monte Carlo simulator is a simple Python program while the code interacting with the simulator platform interfaces with the simulator API described in Section A.2. This code functions as a *learning model* and fully interacts with the API.

A.3.1 *Monte Carlo Simulator*

This code runs exclusively in Python and loads pre-defined volume and roadway configuration data data from tables

Algorithm 1 Monte Carlo Simulator

```
1: procedure ESTIMATE
2:     $volumes \leftarrow$ volume.csv
3:     $LaneStructure \leftarrow$ lanes.csv

       #Defined at runtime
4:     $MarkovRuns$
5:     $SimulationTimeLength$
6:     $Error_{\sigma}$
7:     $Error_{\mu}$
8:     $OptimiseCycle$                                        #Boolean
9:     $MinimumGreen5$
10:    $MaximumGreen$
11:    $MaximumCycle$
12:    $MinimumCycle$
13:    $Saturation$

       #Arrays to hold the output for each run
14:    $Greens$ =Array(Array(Array()))                        #each run, second, phase
15:    $Cycles$ =Array(Array())                               #each run, second
16:    for $run_i \in MarkovRuns$ do
17:        for $time_i \in SimulationTimeLength$ do
18:            $Y_{total,time_i} = 0$
19:            for $phase_i \in LaneStructure$ do
               #Interpolate and get volume for a lane and time
20:                $volume_i \leftarrow volumes(time_i, phase_i)$
21:                $delay_{i,hcm} \leftarrow$HCMDelay($volume_i$, $Greens$, $Cycles$)
22:                $delay_{i,run} = delay_{i,hcm}+$ErrorNorm($Error_{\mu}$, $Error_{\sigma}$)
23:                $X_{est} \leftarrow$GoldenSection($delay_{i,run}$, $phase_i$, $Greens$, $Cycles$)
24:                $Y_{est} \leftarrow X_{est} \times \frac{Greens_{phase_i}}{Cycles}$)
25:                $Y_{total} = Y_{total} + Y_{est}$
26:            end for
27:            if $OptimiseCycle$ then
28:                $Cycles_{run_i,time_i}$ =OptimiseCycle($Y_{total}$)
29:            end if
30:            for $phase_i \in LaneStructure$ do
31:                $Greens_{run_i,time_i,lane_i} \leftarrow$ HCMGreen($Y_{est,phase_i}$, $Y_{total}$)
32:            end for
33:        end for
34:    end for
35: end procedure
```

The *HCMDelay* and *HCMGreen* functions implement the HCM proceedure outlined in Section 2.2.3 and the HCM2010 directly. This includes the ability to do capacity estimation using a base saturation flow rate and adjustments for permissive left turns. The *ErrorNorm* function applies an error value distributed normally with standard deviation $\mathtt{Error}_\sigma$ and mean $\mathtt{Error}_\mu$. In all test cases $\mathtt{Error}_\mu$ was defined as zero.

The *GoldenSection* function implements a Golden Section Search. The algorithm uses the HCM equations to compute the delay for a given value of X, though the calculation of d_2 is done with a pre-calculated capacity value. The process proceeds according to the following process:

A.4 LEARNING MODEL SPECIFICATION

The learning model specification is a custom API guideline used to generate models that interact with

A.4.1 *DRL Model*

The DRL model is implemented using the Tensorflow library. Tensorflow simplifies many aspects of neural network design by providing abstractions for layers, filters, and training algorithms that would otherwise be complicated to code from scratch. It is the basis of other deep learning libraries that can be used in Python, such as Keras.

A.4.1.1 *Network Initialisation*

Network initialisation is done by creating a Tensorflow network according the pre-specified size.

A.4.1.2 *Network Training*

Network training proceeds after the network is initialised. Pseudocode for the training is shown in Algorithm 5 The training process can run for a defined number of steps or can optionally run permanently in the "train" phase (until manually stopped). A number of settings must be defined, many of which are explored in this research. Training proceeds by first obtaining the current state of the simulation, choosing an action, executing the action, obtaining the reward, and obtaining the successor state. These are all added to a replay buffer that holds the past 10,000 frames.

The replay buffer is a queue-like object where experiences are added to the front and older experiences are pushed out from the back. A minibatch system is used to provide experiences that are used to train the model. Minibatches have been shown to provide more stable learning and reduce erratic behaviour in the model by allowing the agent to re-experience past actions. By default a minibatch with 32 frames is drawn from a replay buffer which is then used to train and update the model's weights. Prior to training, the agent must "observe" the network to populate this buffer. The minimum observation time must be equal to the buffer size, but larger observation

Algorithm 2 Monte Carlo Simulator

```
1:  function GoldenSection($delay_{i,run}$, $phase_i$, Greens, Cycles)
2:      Tolerance = 0.0001                    #Stops when calculated delay is within this range
3:      $X_A = 0$
4:      $delay_A \leftarrow$HCMDelayX($X_A$, Greens, Cycles)        #Check delay from no volume
5:      $\phi = \frac{\sqrt{5}+1}{2}$                                 #The golden ratio
6:      if $delay_{i,run} < delay_A$ then
7:          return $X_A$
8:      end if
9:      $Deviation_A = |delay_A - delay_i|$
10:     $X_B = 2$
        #Check delay from a very over-saturated value
11:     $delay_B \leftarrow$HCMDelayX($X_B$, Greens, Cycles)
12:     if $delay_{i,run} > delay_B$ then
13:         return $X_B$
14:     end if
15:     $Deviation_B = |delay_B - delay_i|$
16:     $X_C = X_B - \frac{X_B - X_A}{\phi}$
17:     $delay_C \leftarrow$HCMDelayX($X_C$, Greens, Cycles)
18:     $Deviation_C = |delay_C - delay_i|$
19:     $X_D = X_A + \frac{X_B - X_A}{\phi}$
20:     $delay_D \leftarrow$HCMDelayX($X_D$, Greens, Cycles)
21:     $Deviation_D = |delay_D - delay_i|$
22:     while Deviation > Tolerance do
23:         if $Deviation_C < Deviation_D$ then
24:             $X_B = X_D$
25:             $Deviation_B = Deviation_D$
26:             $Delay_B = Delay_D$
27:         else
28:             $X_A = X_C$
29:             $Deviation_A = Deviation_C$
30:             $Delay_A = Delay_C$
31:         end if
32:         $X_C = X_B - \frac{X_B - X_A}{\phi}$
33:         $delay_C \leftarrow$HCMDelayX($X_C$, Greens, Cycles)
34:         $Deviation_C = |delay_C - delay_i|$
35:         $X_D = X_A + \frac{X_B - X_A}{\phi}$
36:         $delay_D \leftarrow$HCMDelayX($X_D$, Greens, Cycles)
37:         $Deviation_D = |delay_D - delay_i|$
38:     end while
39:     return $X_D$
40: end function
```

Algorithm 3 DRL - Network Initialisation Functions

```
 1: function WEIGHTS($shape$)
 2:     $weights \leftarrow tensorflow.truncated_normal(shape, mean = 0, stddev = 0.01)$
 3:     return $weights$
 4: end function

 5: function BIAS($shape$)
 6:     $bias \leftarrow tensorflow.constant(shape, value = 0.01)$
 7:     return $bias$
 8: end function

 9: function CONVOLUTIONALLAYER($input, filters, stride$)
10:     $padding :=$ "same"
11:     $Conv \leftarrow tensorflow.conv2d(shape, filters, stride, padding)$
12: end function

13: function MAXPOOLLAYER($input$)
14:     $kernel\_size := [1, 2, 2, 1]$                    #Window Size is 2x2
15:     $strides = [1, 2, 2, 1]$                          #Steps around in blocks of 2
16:     $padding :=$ "same"
17: end function
```

times may allow more variability in the initial experience buffer by reducing the effect of the warm-up period.

A.4.2 *SCOOT Emulator*

The SCOOT emulator is coded in Python as a class compatible with the basic *learning model* unit. The module is divided into three key components: the split optimiser, cycle optimiser, and the offset optimiser. All optimisers rely on an implementation of the Robertson model to predict downstream flows Since the simulation API implemented for the DRL model supports two rings, two-rings are also used for the SCOOT emulation.

A.4.2.1 *Robertson Model*

Robertson's model is the fundamental model applied to generate the Cycle Flow Profiles that SCOOT uses. This model is a geometric model that applies a transformation to volumes on the upstream to predict downstream volumes. The basic outline of the algorithm is shown in Algorithm 6. Output is binned at a pre-defined resolution of 15s for all emulation in this work, but may be aggregated or interpolated to match green or cycle starting and ending times.

Algorithm 4 DRL - Network Initialisation

```
 1: procedure NetworkCreation
    #Initialise the weights
 2:     $\mathrm{Action_{count}} = 4$                                   #Final output vector size
 3:     $\mathrm{Matrix_{dim}} = 84$
 4:     $weights_{conv1} \leftarrow \mathrm{Weights}([8, 8, 4, 32])$            #Rank 4 Tensor shape
 5:     $weights_{conv2} \leftarrow \mathrm{Weights}([4, 4, 32, 64])$
 6:     $weights_{conv3} \leftarrow \mathrm{Weights}([3, 3, 64, 64])$
 7:     $weights_{fc1} \leftarrow \mathrm{Weights}([1600, 512])$             #2D Fully connected weights
 8:     $weights_{fc2} \leftarrow \mathrm{Weights}([512, \mathrm{Action_{count}}])$           #Output weights
        #Initialise the Bias
 9:     $bias_{conv1} \leftarrow \mathrm{Bias}([32])$
10:     $bias_{conv2} \leftarrow \mathrm{Bias}([64])$
11:     $bias_{conv3} \leftarrow \mathrm{Bias}([64])$
12:     $bias_{fc1} \leftarrow \mathrm{Bias}([512])$
13:     $bias_{fc2} \leftarrow \mathrm{Bias}([\mathrm{Action_{count}}])$

        #Initialise the layers using ReLU for convolutional layers
14:     $layer_{input} := [\mathrm{None}, \mathrm{Matrix_{dim}}, \mathrm{Matrix_{dim}}, 4]$       #Batch of the last 4 states
15:     $layer_{conv1} \leftarrow tensorflow.relu(\mathrm{ConvolutionalLayer}(layer_{input}, weights_{conv1}, 4)$
    $+bias_{conv1}$
16:     $layer_{conv2} \leftarrow tensorflow.relu(\mathrm{ConvolutionalLayer}(layer_{conv1}, weights_{conv2}, 2)$
    $+bias_{conv2}$
17:     $layer_{conv3} \leftarrow tensorflow.relu(\mathrm{ConvolutionalLayer}(layer_{conv3}, weights_{conv3}, 1)$
    $+bias_{conv3}$
        #Fully connected layers use matrix multiply
18:     $layer_{fc1} \leftarrow tensorflow.relu(tensorflow.matmul(layer_{conv3}, weights_{fc1})) +$
    $bias_{fc1}$
19:     $layer_{out} \leftarrow tensorflow.relu(tensorflow.matmul(layer_{fc1}, weights_{fc2})) +$
    $bias_{fc2}$
20: end procedure
```

Algorithm 5 DRL - Network Training

```
1: procedure TRAIN
2:     environment ← Simulator.API
3:     $\gamma$                                                    #Defined at runtime
4:     D :=Dequeue(1000)            #Replay memory, default maximum 10000 frames
5:     $Batch_{size} := 32$                                        #Minibatch size
6:     $\epsilon := \epsilon_{initial}$                            #Set at runtime, usually 1
7:     $\epsilon_{final}$                                          #Set at runtime, usually 0.01
8:     $time_{observation}$                                        #Set at runtime
9:     $time_{explore}$                                            #Set at runtime
10:    $time_{train}$                                              #Set at runtime, or indefinite
11:    $time_{all} = time_{observation} + time_{explore} + time_{train}$
12:    $state_t$ ← environment.state                               #Initial state
13:    $\eta$                                                      #Learning rate, defined at runtime
14:    trainer ←Tensorflow.AdamOptimiser($\eta$)
15:    while step < $time_{all}$ do
16:        state ← environment.state
17:        $actions_{all}$ ←Evaluate($state_t$)                     #Gives Q-values for actions
18:        random ←RandomNum(0, 1)
19:        $actions_{valid}$ =ValidAction(output) #Check in the API which actions are valid
20:        if random < $\epsilon$ then
21:            $action_{choice}$ =RandomSample($actions_{valid}$)
22:        else
23:            $action_{choice}$ =Max($actions_{valid}$)            #Select the max Q-value action
24:        end if
25:        if then$\epsilon > \epsilon_{final}$ & step > $time_{observation}$
26:            $\epsilon = \epsilon - \frac{\epsilon_{initial} - \epsilon_{final}}{time_{explore}}$    #Anneal $\epsilon$
27:        end if
28:        enviornment.DoAction($action_{choice}$)
29:        reward ← environment.Reward
30:        $state_{t+1}$ ← environment.state
31:        terminal ← environment.Terminal #Check if the simulation needs to restart
32:        D.add($state_t$, $actions_{all}$, reward, $state_{t+1}$, terminal)    #Excess discarded
33:        if step > $time_{observe}$ then                          #Train the model \
           #minibatch has elements state, actions, reward, $state_{t+1}$, terminal
34:            minibatch =RandomSample(D, $Batch_{size}$)
               #Calculate the rewards scaled with gamma for the batch
35:            for batch ∈ minibatch do
36:                if $batch_{terminal}$ then
37:                    $batch_{reward,calculated} = batch_{reward}$
38:                else
39:                    $batch_{reward,calculated} = batch_{reward} + \gamma *$Max(actions)
40:                end if
41:            end for
               #Run a training step and update the weights of the network
42:            trainer.Run(minibatch)
43:        end if
44:    end while
45: end procedure
```

Algorithm 6 SCOOT Emulator - Robertson Model

```
1: procedure ComputeRobertson(direction, controller, corridor, time)
     #Volume data is obtained from valid upstream scoot-enabled detectors
     #Volume data is binned per-15 seconds in this implementation
2:     $volume_{binned,up} \leftarrow corridor.detectors(controller, direction)$
3:     $bin_{size} := 15s$
4:     for $bin_i \in volume_{up}$ do
5:         $F = \frac{1}{1+\alpha t}, t = \beta T$
6:         $q_{down}(bin_i + t) := F \times q_{up}(bin_i) + (1 - F) \times q_{down}(bin_{i-1} + t)$
7:     end for
8:     return $q_{downstream,binned}$
9: end procedure
```

A.4.2.2 *Split Optimiser*

The split optimiser's main operation is divided by the direction of the corridor. The main operation is shown in Algorithm 7. By default, all directions are equally weighted and the optimiser attempts to make an adjustment that balances the degree of saturation (X) between all approaches in all rings of the controller. This is done with the process shown in Algorithm **??**. Calculation proceeds by using the Robertson model to estimate volume flow rates. Saturation flow rates must be pre-calculated and specified for all approaches and are used to estimate the degree of saturation from the green and cycle times. The algorithm makes a recommendation by first setting flags for the retard or advance decision to true. A check is then done against all approaches to see if the change produces a resulting saturation flow rate that is greater than (for the case of retard) or less than (for the case of advance) the corridor direction's current X value in the ring considered. If such a case is found then the flag condition is set to false. If both retard and advance are false, then the algorithm recommends keeping the current split time. This can occur due to the resolution of the check ($\pm 4s$), but it is not possible for both flags to be true. Finally, a check is also made on minimum and maximum greens, and a flag is set to false if it would result in any phase having an inappropriate green time.

A.4.2.3 *Cycle Optimiser*

The Cycle Optimiser runs at the end of the cycle which is defined relative to a reference phase. In this implementation the start of the corridor left turns is used as the reference cycle. The procedure is outlined in Algorithm 8. This algorithm works by checking each intersection's predicted maximum degree of saturation and choosing the adjustment that brings it closest to the target value (0.9 by default). Half and double cycles are not considered in this implementation. Similar to split adjustment, if a change would adjust the cycle below a maximum or minimum cycle the change is disallowed.

Algorithm 7 SCOOT Emulator - Split Optimisation

```
 1: procedure SplitOptimiser(corridor, controller, action)
 2:     upstreams ← corridor.upstream(controller) #Directions that can be optimised

        #Calculate the benefit of changing the split
 3:     recommendation ← Recommend(controller, corridor)
 4:     $greens_{total,approach}$ ← controller.greens
 5:     cycle ← controller.cycle
 6:     offset ← corridor.offset(controller, direction)
 7:     saturation ← controller.saturation                #Pre-specified at runtime
        #Comparisons are made against the dominant direction of the corridor
 8:     predictedvol ← Robertson(direction, controller, corridor )

 9:     $X_{direction,retard} := \frac{predictedvol}{saturation \times \frac{greens_{direction}-4}{cycle}}$
10:     $X_{direction,advance} := \frac{predictedvol}{saturation \times \frac{greens_{direction}+4}{cycle}}$
11:     $X_{retard} :=$ Max($X_{direction,retard}$)
12:     $X_{advance} :=$ Min($X_{direction,advance}$)


        #These Booleans indicate if retarding/advancing moves toward balancing all Xs
13:     $recommend_{retard}$ := true
14:     $recommend_{advance}$ := true

15:     for approach ∈ controller.intersection do
        #Compare the resulting X on the corridor to all other approaches
16:         if $approach_{direction} \neq$ direction then
17:             $approach_{vol}$ ← Robertson(direction, controller, corridor )
                #X for the approach will vary depending on the comparison
18:             $X_{approach,retard,advance} := \frac{predictedvol}{saturation \times \frac{greens_{approach}\pm 4}{cycle}}$
19:             if $X_{approach,retard} \geqslant X_{retard}$ then
20:                 $recommend_{retard}$ := false
21:             end if
22:             if $X_{approach,advance} \leqslant X_{advance}$ then
23:                 $recommend_{advance}$ := false
24:             end if
25:             $recommend_{retard}$ ←CheckMinMaxGreen(approach)
26:             $recommend_{advance}$ ←CheckMinMaxGreen(approach)
27:         end if
28:     end for
29:     return $recommend_{retard,advance,keep}$
30: end procedure
```

After deciding on and when evaluating an adjustment the algorithm implements the cycle change by proportionally increasing or decreasing green time to each phase.

Algorithm 8 SCOOT Emulator - Cycle Optimisation

```
1: procedure Cycle(corridor)
2:     $adjustments_{cycle} = (-32, -16, -8, -4, 0, 4, 8, 16, 32)$
3:     for controller ∈ corridor do
4:         cycle ← controller.cycle
5:         $X_{target} = 0.90$
6:         for direction ∈ controller do
7:             for change ∈ $adjustments_{cycle}$ do
8:                 if $change + cycle \geqslant cycle_{min}$ & $change + cycle \leqslant cycle_{max}$ then
9:                     q ← ComputeRobertson(direction, controller, corridor)
10:                    S ← controller.saturation                #Pre-specified at runtime
11:                    $X_{controller,direction,change} = \frac{q}{S \times \frac{green}{cycle+change}}$
12:                    $DiffX_{controller,direction,change} := X_{target} - X_{controller}$
13:                end if
14:            end for
15:        end for
16:    end for
17:    $change_{final}$ ← Max(change, DiffX)
18: end procedure
```

A.4.2.4 *Offset Optimisation*

The Offset Optimiser works by checking the queues produced by each candidate offset adjustment. Queues are estimated using the binned Robertson volumes in combination with a deterministic queueing model. The deterministic model assumes that no vehicles can discharge during red but that vehicles can discharge at the pre-defined saturation flow rate on green.

Offset adjustments are done every 5 minutes and implemented by proportionally adding or subtracting time to each phase to bring the offset in alignment with the new value by the next cycle. Offset adjustments can shorten the green time if the adjustment is small, otherwise offset adjustments are done by adding green to each phase.

Algorithm 9 SCOOT Emulator - Offset Optimisation

```
1:  procedure OFFSET(corridor)
    #Check whether a change in the offset improves the predicted arrivals
2:      $adjustments_{offset} := -4, 0, 4$
3:      for controller $\in$ corridor do
4:          for direction $\in$ (controller.directions $\in$ corridor do
5:              for change $\in$ $adjustments_{offset}$ do q $\leftarrow$
    ComputeRobertson(direction, controller, corridor) $green_{start,end}$ $\leftarrow$
    controller.greens + change queue $\leftarrow$ D/D/x(q, $green_{start,end}$
6:              end for
7:          end for
8:      end for
9:      $change_{final} \leftarrow$ Min(change, queue)
10: end procedure
```

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

`https://bitbucket.org/amiede/classicthesis/`

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

`http://postcards.miede.de/`

Final Version as of May 18, 2021 (`document` revision 2.0).