REINFORCEMENT LEARNING TRAFFIC SIGNAL CONTROL

DEEP REINFORCEMENT LEARNING ADAPTIVE TRAFFIC SIGNAL CONTROL

By WADE GENDERS, B.Eng.Soc, M.A.Sc (McMaster University)

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy.

McMaster University DOCTOR OF PHILOSOPHY (2018)
Hamilton, Ontario (Civil Engineering)

TITLE: Reinforcement Learning Traffic Signal Control

AUTHOR: Wade Genders, B.Eng.Soc, M.A.Sc (McMaster University)

SUPERVISOR: Dr. Saiedeh Razavi

NUMBER OF PAGES: xvii; 119

LAY ABSTRACT

Inefficient transportation systems negatively impact mobility, human health and the environment. The goal of this research is to mitigate these negative impacts by improving automated transportation control systems, specifically intersection traffic signal controllers. This research presents a system for developing adaptive traffic signal controllers that can efficiently scale to the size of cities by using machine learning and parallel computation techniques. The proposed system is validated by developing adaptive traffic signal controllers for 196 intersections in a simulation of the City of Luxembourg, Luxembourg, successfully reducing delay, queues, vehicle stopped time and travel time.

ABSTRACT

Sub-optimal automated transportation control systems incur high mobility, human health and environmental costs. With society reliant on its transportation systems for the movement of individuals, goods and services, minimizing these costs benefits many. Intersection traffic signal controllers are an important element of modern transportation systems that govern how vehicles traverse road infrastructure. Many types of traffic signal controllers exist; fixed time, actuated and adaptive. Adaptive traffic signal controllers seek to minimize transportation costs through dynamic control of the intersection. However, many existing adaptive traffic signal controllers rely on heuristic or expert knowledge and were not originally designed for scalability or for transportation's big data future.

This research addresses the aforementioned challenges by developing a scalable system for adaptive traffic signal control model development using deep reinforcement learning in traffic simulation. Traffic signal control can be modelled as a sequential decision-making problem; reinforcement learning can solve sequential decision-making problems by learning an optimal policy. Deep reinforcement learning makes use of deep neural networks, powerful function approximators which benefit from large amounts of data. Distributed, parallel computing techniques are used to provide scalability, with the proposed methods validated on a simulation of the City of Luxembourg, Luxembourg, consisting of 196 intersections.

This research contributes to the body of knowledge by successfully developing a scalable system for adaptive traffic signal control model development and validating it on the largest traffic microsimulator in the literature. The proposed system reduces delay, queues, vehicle stopped time and travel time compared to conventional traffic signal controllers. Findings from this research include that using reinforcement learning methods which explicitly develop the policy offers improved performance over purely value-based methods. The developed methods are expected to mitigate the problems caused by sub-optimal automated transportation signal controls systems, improving mobility and human health and reducing environmental costs.

PREFACE

Included Manuscripts

This thesis contains three manuscripts, as listed below:

- W. Genders and S. Razavi, "Asynchronous n-step q-learning adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, 2018. doi:10.1080/15472450.2018.1491003. *Accepted, In Press*
- W. Genders and S. Razavi, "Policy analysis of reinforcement learning adaptive traffic signal control," *ASCE Journal of Computing in Civil Engineering*, 2018. *4th round of revisions*, CPENG-2667R4
- W. Genders and S. Razavi, "Distributed deep deterministic policy gradients for adaptive traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2018. *Submitted 15 August 2018*, T-ITS-18-08-0805

Manuscript Details

Paper 1 Presented in Chapter 2

The work was completed between May 2015 and October 2017. The manuscript was submitted in October 2017 and accepted in June 2018. My contributions are:

- Development of the reinforcement learning adaptive traffic signal control proof-of-concept.
- Simulation implementation, experiments and analysis.
- Manuscript authorship and journal submission corresponding author.

Paper 2 Presented in Chapter 3

The work was completed between May 2015 and January 2018. The manuscript was submitted in January 2018. My contributions are:

- Development and extension of the reinforcement learning adaptive traffic signal control proof-of-concept.
- Simulation implementation, experiments and analysis.
- Manuscript authorship and journal submission corresponding author.

Paper 3 Presented in Chapter 4

The work was completed between May 2015 and August 2018. The manuscript was submitted in August 2018. My contributions are:

- Development of the distributed reinforcement learning adaptive traffic signal control architecture.
- Simulation experiments and analysis.
- Manuscript authorship and journal submission corresponding author.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Saiedeh Razavi, for her guidance and support during my studies. I'm especially grateful to my supervisor for the research freedom I was afforded which made my time fruitful and enjoyable - a Ph.D. student could ask for little more. Thank you to Dr. Rong Zheng and Dr. Antonio Páez for their time and efforts serving on my Ph.D. committee.

To my current and former lab mates, Shuming Du and Dr. Jun Wang, it has been a pleasure to work beside you and I'm honoured to call you my friends. Thanks to Shuo Feng for all the interesting conversations over lunch.

Thanks to Mike Justason for the hours of interesting (mathematical) conversation which preserved my sanity and for keeping me fed. Thanks to Dr. Brian Baetz for all the advice and support and for connecting Dr. Razavi and I all those years ago.

To my friends and family, especially my Mom and Dad, I'm lucky to have such a loving, supportive group of people in my life.

To my wife Virginia, your love and support sustained me during my education. Thank you for encouraging me when I struggled and for celebrating my successes - I love you.

"The only thing I know is that I know nothing."
-Socrates

"I was born not knowing and have had only a little time to change that here and there."
-Richard Feynman

Contents

LISTS OF FIGURES AND TABLES

List of Figures

List of Tables

LIST OF ABBREVIATIONS, INITIALISMS AND ACRONYMS

A3C	Asynchronous Advantage Actor-Critic
DDPG	Deep Deterministic Policy Gradients
DNN	Deep Neural Network
DQN	Deep Q-Network
FC	Fully Connected
LuST	Luxembourg SUMO Traffic Scenario
MDP	Markov Decision Process
MoE	Measure of Effectiveness
NEMA	National Electrical Manufacturers Association
NN	Neural Network
RBF	Radial Basis Function
ReLu	Rectified Linear Unit
RL	Reinforcement Learning
TSC	Traffic Signal Control/Controller

DECLARATION OF ACADEMIC ACHIEVEMENT

Wade Genders was the main contributor and first author for all manuscripts in this thesis. Any co-author contributions are detailed at the beginning of each chapter that includes a published manuscript.

1 INTRODUCTION

1.1 Problem Statement and Motivation

Transportation systems are fundamental to society - all users benefit from increased mobility at low cost. Urban transportation systems are governed by automated control systems, such as intersection traffic signal controllers (TSC), that strive for optimal control, attmpeting to maximize safety and mobility and minimize cost. Transportation systems experience dynamic and stochastic demand, requiring systems and control policies that can adapt and respond to change. Sub-optimal control in transportation systems creates large inefficiencies which incur significant costs due to widespread use. Intersection traffic signal controllers enacting sub-optimal decisions contribute to congestion, negatively affecting the environment, human health and mobility [7]. These costs can be quantified financially [8], 10^8 to 10^9 United States Dollars (USD) per annum for large populations and 10^3 USD for individuals [9, 10, 11, 12]. These costs manifest as delay to individuals and unnecessary fuel consumption/emission [13] and are incurred when vehicles accelerate, decelerate and idle at intersections or when vehicles take longer travel times than necessary [14]. The transportation sector contributes 24% of global carbon dioxide emissions, with approximately three-quarters emitted by road traffic [15]. Annually, air pollution is responsible for over three million fatalities globally [16] with emissions from land transportation contributing one out of three fatalities from air pollution in North America [17]. Individuals living in metropolitan areas consistently exposed to traffic related noise and air pollution have statistically significant increases in coronary heart disease mortality [18]. With transportation systems used universally throughout society, improvements can yield significant benefits from cumulative effects. Sub-optimal TSC are a result of their control policy, which is often governed by simple or heuristic logic (e.g., fixed time TSC) that does not consider current traffic conditions. It would be desirable to have a TSC with *adaptive* capabilities, that predicates its behaviour on current traffic conditions.

However, it can be challenging to develop optimal control policies for dynamic, stochastic and high dimensional environments such as traffic intersections [19]. Vehicle traffic at intersections is dynamic, changing demand from peak to free-flow throughout the day. Intersection dynamics exhibit stochasticity, both from inherent stochasticity (e.g., driver behaviour and idiosyncrasies) and sensor aliasing. The optimal traffic signal control policy must be able to adapt to these changes and uncertainties.

Intelligent transportation systems offer solutions and improvements for transportation problems using computational intelligence. Much research has been devoted to studying and developing adaptive traffic signal controllers. An adaptive traffic signal controller,

> ...uses detection data and algorithms to adjust signal timing parameters for current conditions...is able to adjust various timing parameters (within certain constraints) based on what the traffic requires. Over time, traffic evolves - whether quickly in minutes or slowly in days, months, or years - and renders signal timing plans increasingly less effective. Traditional signal operations are often unable to adjust to this variability, resulting in degradation of traffic performance. [20, p. 9-11].

Adaptive TSC attempt to improve upon traditional TSC by implementing dynamic phase durations and acyclic sequencing. Numerous techniques have been used to develop adaptive traffic signal controllers; dynamic programming [21, 22], fuzzy systems [23, 24] evolutionary algorithms [25, 26, 27]. Adaptive systems can use a combination of sensor observations, models and memory to make traffic signal control decisions. Some of these adaptive techniques have seen field deployment; SCOOT [28], OPAC [29], SCATS [30], RHODES [31] and ACS-Lite [32]. However, many of the aforementioned methods rely on models and/or expert knowledge. This can be problematic in cases where models do not replicate reality or if expert knowledge is incorrect. These techniques can also become infeasible as the quantity or dimensionality of the data increases. With the introduction of connected and autonomous vehicles along with additional traffic sensors (e.g., video cameras, radar), transportation systems are quickly becoming domains of big data [33]. Traditional control systems were not designed to utilize data at this scale, necessitating new solutions. Adaptive TSC solutions which overcome these problems are desirable.

1.2 Research Objectives

To address the aforementioned problems, the primary research objective of thesis is to develop an adaptive traffic signal controller for optimal traffic signal control which overcome the limitations of conventional traffic signal controllers. A traffic signal controller that can adapt to the dynamic and stochastic nature of traffic will reap improvements by reducing congestion and decreasing costs from inefficiencies. Since transportation systems exhibit large numbers of users, any improvements or efficiencies benefit many. Developing improved automated control systems for key transportation infrastructure, such as traffic signal controllers, is desirable to reduce mobility, environmental and health costs. Sub-objectives which will aid in achieving the primary objective include answering the following questions:

- What decisions should an adaptive traffic signal controller make?
- How can the solution be made scalable?

1.3 Research Outline

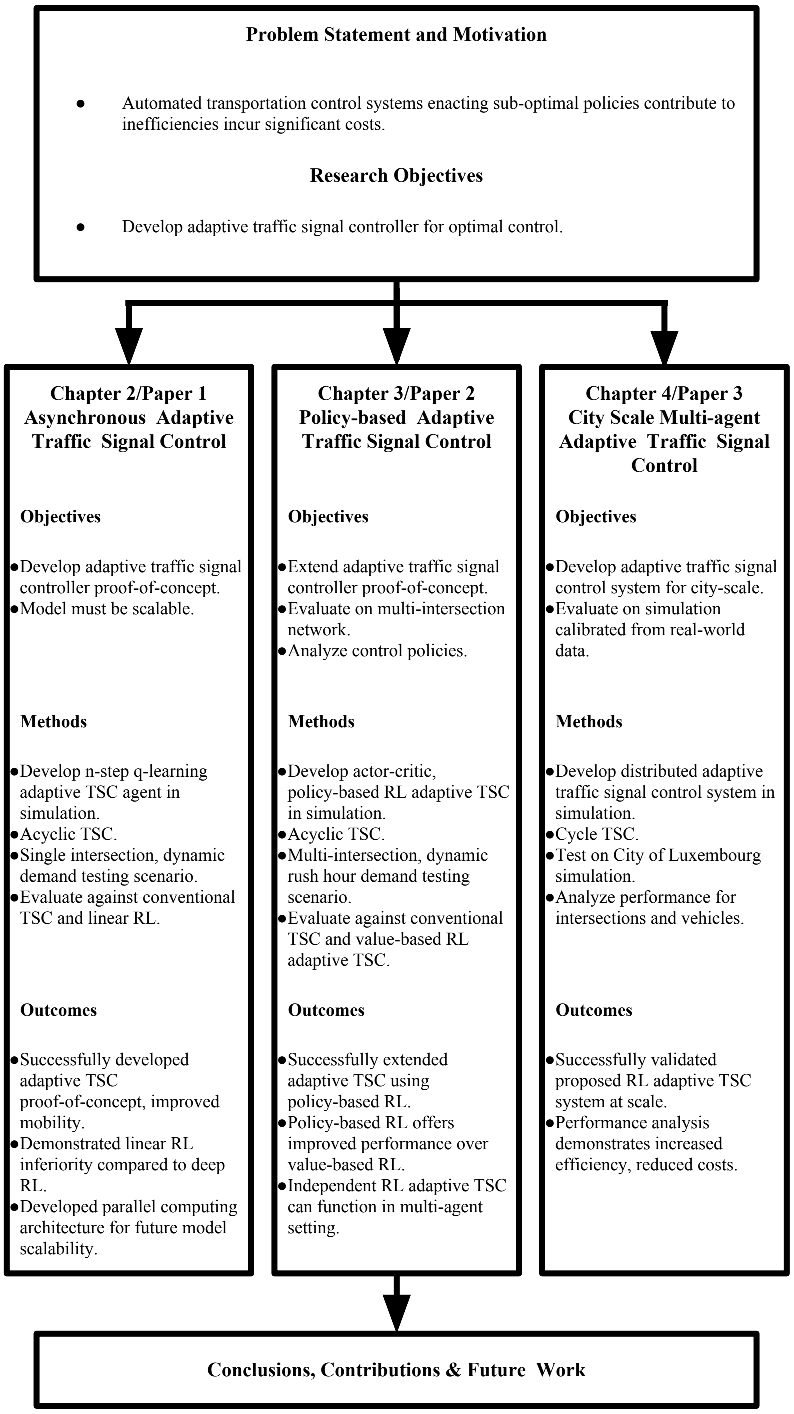

Problem Statement and Motivation
• Automated transportation control systems enacting sub-optimal policies contribute to inefficiencies incur significant costs.
Research Objectives
• Develop adaptive traffic signal controller for optimal control.
Chapter 2/Paper 1 Asynchronous Adaptive Traffic Signal Control
Objectives
•Develop adaptive traffic signal controller proof-of-concept.
•Model must be scalable.
Methods
•Develop n-step q-learning adaptive TSC agent in simulation.
•Acyclic TSC.
•Single intersection, dynamic demand testing scenario.
•Evaluate against conventional TSC and linear RL.
Outcomes
•Successfully developed adaptive TSC proof-of-concept, improved mobility.
•Demonstrated linear RL inferiority compared to deep RL.
•Developed parallel computing architecture for future model scalability.
Chapter 3/Paper 2 Policy-based Adaptive Traffic Signal Control
Objectives
•Extend adaptive traffic signal controller proof-of-concept.
•Evaluate on multi-intersection network.
•Analyze control policies.
Methods
•Develop actor-critic, policy-based RL adaptive TSC in simulation.
•Acyclic TSC.
•Multi-intersection, dynamic rush hour demand testing scenario.
•Evaluate against conventional TSC and value-based RL adaptive TSC.
Outcomes
•Successfully extended adaptive TSC using policy-based RL.
•Policy-based RL offers improved performance over value-based RL.
•Independent RL adaptive TSC can function in multi-agent setting.
Chapter 4/Paper 3 City Scale Multi-agent Adaptive Traffic Signal Control
Objectives
•Develop adaptive traffic signal control system for city-scale.
•Evaluate on simulation calibrated from real-world data.
Methods
•Develop distributed adaptive traffic signal control system in simulation.
•Cycle TSC.
•Test on City of Luxembourg simulation.
•Analyze performance for intersections and vehicles.
Outcomes
•Successfully validated proposed RL adaptive TSC system at scale.
•Performance analysis demonstrates increased efficiency, reduced costs.
Conclusions, Contributions & Future Work

Figure 1: Research Outline

1.4 Thesis Organization

This thesis is composed of the following chapters:

Chapter 1 provides an introduction to the problem along with research motivations, subsequent research objectives and research outline to achieve the objectives.

Section 1.5 provides a background on important topics used in this thesis; Traffic Signal Control, Machine Learning, Markov Decision Processes, Reinforcement Learning and Neural Networks. Readers familiar with these topics can skip this chapter.

Section 1.6 provides a detailed methodology for achieving the research objectives.

Chapter 2 develops the first proof-of-concept solution, developing a reinforcement learning, adaptive TSC for an isolated intersection in simulation. This chapter is composed of a published journal article.

Chapter 3 extends the proof-of-concept by developing an adaptive TSC using different reinforcement learning techniques. Instead of using value-based reinforcement learning, policy based methods are used to achieve higher performance compared to Chapter 2. Simulation experiments are conducted in a multi-intersection setting and the developed reinforcement learning policies analyzed to better understand the adaptive TSC.

Chapter 4 uses knowledge generated from preceding chapters to develop a scalable system for adaptive TSC and validate it in city scale traffic simulation. Results show the proposed system achieves the research objectives, optimizing transportation by minimizing costs, reducing vehicle delay and queues, stopped and travel time.

Chapter 5 concludes the thesis by summarizing the work presented, its main contributions, limitations and areas for future work.

Appendices A and B provide supplemental research conducted during the course of the thesis.

1.5 Background

1.5.1 Traffic Signal Control

Modern transportation systems rely on various automated control systems. Road intersections are ubiquitous in urbanized areas and are predominantly governed by traffic signal controllers which use tuples of coloured lights known as phases to control intersection user movements.

Fundamentally, a traffic signal controller must enact two control decisions in sequence: selecting the current phase from among a finite set to enact, and for how long in duration? A traffic signal controller's purpose is to serve intersection users (e.g., vehicles, pedestrians) in safely crossing the intersection with different phases. If the traffic signal controller implements phases in a cycle, phases displayed in an ordered sequence, then the answer to the first control decision is simply to consult the cycle. If a cycle is used, the duration of the phase, the second control decision, still must be chosen. Various classes of traffic signal controllers provide a myriad of answers to these decisions.

The simplest class is a fixed time traffic signal controller, which implements phases in a cycle with fixed durations. Phase durations are often calculated based on historical or estimated based on traffic metrics (e.g., vehicle flow, density). However, as the disparity between estimation and reality increases, fixed phase durations become increasingly sub-optimal.

1.5.2 Machine Learning

Although some question its longevity, Moore's Law [34] has continued to produce cheaper, smaller and more efficient computer hardware, making computers ubiquitous. Transportation systems have benefited from Moore's law, with increased automation and data generation from sensors. Consequently, because transportation is pervasive, observing transportation systems yields ample data. Machine learning offers an ensemble of mathematical techniques that benefit from large quantities of data to develop solutions to problems.

Machine Learning allows computers to attain knowledge from data to solve problems, often in high dimensional problems which overwhelm human capabilities. Computers are particularly good at fast computation and storing large quantities of data. These two qualities can be leveraged to solve some problems that humans find difficult or solve sub-optimally.

Intersection traffic signal control can be modelled as sequential decision-making problem and machine learning can learn how to make the optimal decisions given sufficient data.

1.5.3 Markov Decision Processes

Decision-making can be theoretically modeled by Markov Decision Processes (MDP) [35]. An MDP can be used to describe sequential problems where outcomes are partly stochastic and partly influenced by a decision maker who can act. Mathematically, an MDP is defined by a tuple $< S, A, R, T, \gamma >$; consisting of a state space S, an action space A, a reward function $R : S \times A \times S \to \mathbb{R}$, a State Action Transition function $T : S \times A \times S \to \mathbb{R}$ and a discount factor $\gamma \in [0, 1)$. Beyond these sets and functions, an MDP assumes the Markov property.

Beginning at time t, assume we encounter a sequence of states, potentially infinite, and define it as the state history $S_H = s_t, s_{t+1}, ..., s_{t+\infty}$. If this history exhibits the Markov property, then $\mathbb{P}(s_{t+1}|s_H) = \mathbb{P}(s_{t+1}|s_t)$, meaning information about future states is predicated entirely on the present state, also described as memoryless. This property can be used to simplify calculation and aid in optimization. A policy π is a state-action function $\pi : S \times A \to [0, 1]$, describing action selection probabilities conditioned on the state $\pi(s, a) = \mathbb{P}[a_\pi = a | s_\pi = s]$. Actions influence future states, actions and rewards. Reward R quantitatively describes success or failure relative to an abstract goal and is ultimately influenced by states and actions. The reward can be used to compare and improve policies. Different states, actions and policies are said to be optimal if they lead to maximum reward.

1.5.4 Reinforcement Learning

Reinforcement Learning is a type of machine learning which uses noisy feedback and past experience to solve problems [36]. A reinforcement learning agent acts in an environment to achieve a goal, developing behaviour to solve sequential decision-making problems. Reinforcement learning relies on MDP for modelling sequential decision-making problems, however different classes of algorithms make different mathematical assumptions. Model-based reinforcement learning explicitly models the state action transition function T and Reward function R. These are built from dynamic programming methods [37]. Model-free reinforcement learning techniques do not model a state action transition function or reward function and instead rely exclusively on environment samples. Both of these methods seek to develop policies which achieve the maximum reward, known as optimal policies.

Reinforcement learning offers a variety of algorithms to develop agents for environments (i.e., MDP). However, they all share a common goal of developing a policy to maximize reward. The concept of value in reinforcement learning represents the notion of long term reward [36]. A reinforcement learning agent can estimate values of states and actions to develop the optimal policy. The agent uses its experiences with the environment to calculate and improve value estimates. Since the environment is unknown and potentially stochastic, the agent uses previous success coupled with current observations to achieve optimal behaviour.

To make decisions in a MDP, an *agent* must *act*. Consider an agent interacting with an MDP through a policy π in discrete time intervals. Desirable agents interact with an environment (i.e., MDP) and receive high amounts of reward. A policy is a function from states to actions $\pi : S \rightarrow A$, where the agent first observes the state s_t and then chooses an action a_t based on the state. The agent's action transitions the MDP to a next state s_{t+1} and the agent receives a reward r_t. A tuple representing one interaction between the agent and an MDP is known as an experience $e_t = (s_t, a_t, r_t, s_{t+1})$.

Many successful reinforcement learning algorithms exist that use experiences in different ways to develop optimal policies; Q-learning [38], SARSA [39], TD-λ [36] policy based methods; REINFORCE [40], natural gradients [41]. Optimal policies can be developed for low dimensional problems

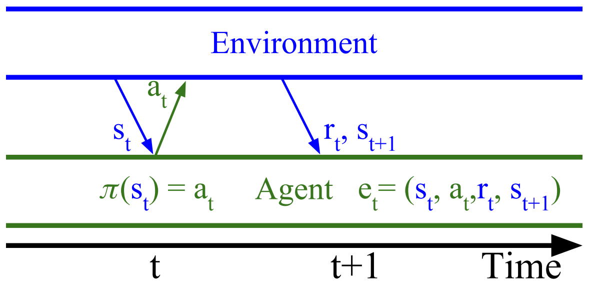


Figure 2: At time $t + 1$ an experience tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ is generated from the interaction of the agent with the environment. The agent observes the state s_t of the environment s_{Env} which may be partially $s_t \subset s_{Env}$ or fully observable $s_t = s_{Env}$. The agent uses the policy to select an action $\pi(s_t) = a_t$; policies can be stochastic or deterministic. The environment transitions and returns feedback for the agent in the form of a quantitative reward r_t and a new state s_{t+1}. The agent uses experiences to learn, to move towards the optimal policy π^*.

with tabular function approximation, however as dimensionality increases, tabular methods become infeasible. Linear function approximation can improve upon tabular solutions, but may still have insufficient model capacity. Many function approximation methods have been developed beyond tabular and linear, both parametric and non-parametric. Neural network function approximation has been used to develop deep reinforcement learning agents with success.

1.5.5 Neural Network Function Approximation

Neural Networks are powerful function approximators. Neural networks use computational neurons and function composition to model a wide family of functions [42]. A comparison with linear regression, a common function approximator, is provided for comprehension, displayed in Figure 3. It is useful to consider the similarities and differences between these methods if unfamiliar with neural networks. Mathematical notation in this section includes x scalars, $\vec{x}$ vectors, X matrices and $\boldsymbol{X}$ tensors.

Both linear regression and neural networks are parametric methods, using a set of parameters θ to to develop a function approximation of a target function $f(x) \approx f(x;\theta)$. Linear regression produces a single output from the dot product of an input vector and a parameter vector $f(\vec{x};\vec{\theta}) = \vec{x} \cdot \vec{\theta}$. The input vector $\vec{x} \in X$ is often considered a sample from some function. Each input vector has a corresponding output from a label set $y \in \vec{y}$, often referred to as the 'true' output, creating a data tuple $(\vec{x}, y)$. This is often data observed from some phenomena or experiment. Linear

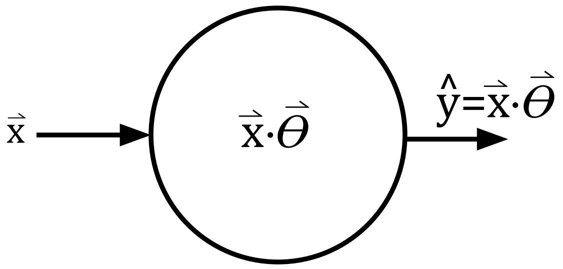

$\vec{x}$
$\vec{x} \cdot \vec{\theta}$
$\hat{y} = \vec{x} \cdot \vec{\theta}$

Figure 3: Computational neuron without activation function. A computational neuron computes the dot product $\vec{x} \cdot \vec{\theta}$ between its parameter vector $\vec{\theta}$ and an input vector $\vec{x}$. From a geometric perspective, a neuron's computation can be reformulated as the cosine similarity between the input and parameter vectors $cos(\phi) = \frac{\vec{x} \cdot \vec{\theta}}{\|\vec{x}\| \|\vec{\theta}\|}$. A single neuron is equivalent to linear regression.

regression provides a framework to model the relationship between the function input and output from pairs drawn from a dataset with a linear combination of the input and parameter vectors. This linear function is parameterized by its parameter vector $\vec{\theta}$. Like any machine learning algorithm, linear regression ultimately seeks to model the relationship of the data well (i.e., the approximation should be accurate). The approximation's quality is measured using an error (or loss or cost) function. The accuracy of the approximation depends on the parameters. Linear regression optimizes the parameters by minimizing the error between linear regression's estimate of the correct output $\hat{y} = f(\vec{x}, \theta)$ and the target output y. Linear regression uses the ordinary least squares error function, defined in Equation 1.

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2 \tag{1}$$

Optimal parameters minimize the given loss function and produce accurate approximations.

To represent a neural network, a parameter tensor is used $\mathbf{\Theta}$. Each layer is represented as a parameters matrix $\Theta \in \mathbf{\Theta}$, with individual neuron parameters $\vec{\theta} \in \Theta$. Neural networks have many neurons per layer and multiple layers, this is represented as function composition $f(\vec{x}; \mathbf{\Theta}) = f(...f(f(\vec{x}; \Theta); \Theta)...; \Theta)$. Common error functions for optimizing neural network parameters are the mean squared error and cross-entropy, defined in Equation 2.

$$L(\vec{y}, \vec{\hat{y}}) = -\sum \vec{y} \log(\vec{\hat{y}}) \tag{2}$$

Neural networks build from linear regression by being capable of modelling a wider family of functions. These additional capabilities are afforded through function composition and activation

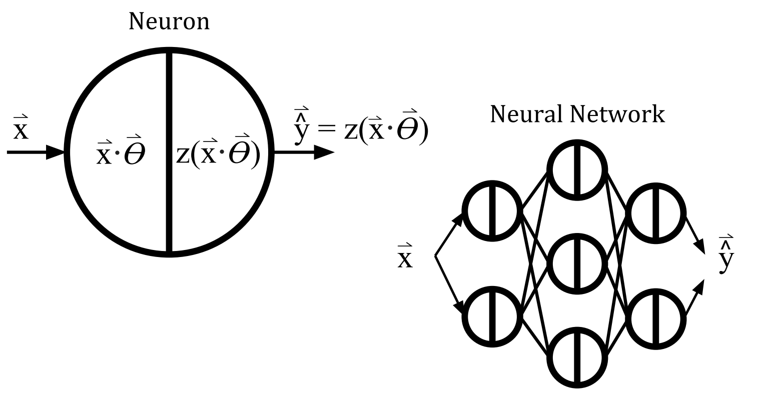

Neuron
$\vec{x}$
$\vec{x}\cdot\vec{\Theta}$
$z(\vec{x}\cdot\vec{\Theta})$
$\hat{\vec{y}} = z(\vec{x}\cdot\vec{\Theta})$
Neural Network
$\vec{x}$
$\hat{\vec{y}}$

Figure 4: A neural network is composed of layers of neurons with activation functions $z(\vec{x} \cdot \vec{\Theta})$ after the dot product. Function composition is accomplished by each hidden layer's output becoming the input vector for the next layer. Different activation functions are utilized depending on the problem, common examples include linear, sigmoid, rectified linear (ReLu), softmax and hyperbolic tangent.

functions. Function composition allows for learning hierarchical representations of the input data, learning high level representations of data from lower, more basic representations. Activation functions further transform the data, yielding additional model capacity. To train the hidden layer parameters, the backpropagation algorithm is used [43, 44]. Backpropagation is a gradient-based optimization technique. Gradient-based optimization uses a function's derivative to find its optimal values, gradient descent (i.e., minimize function) defined in Equation 3.

$$x' = x - \alpha f'(x) \tag{3}$$

Where x' represents the next iteration taking a gradient step $f'(x)$ scaled by α starting from x. Repeatedly applying this gradient step approaches a function minimum. However, since this is a first order optimization algorithm, it is unknown whether a global or local minimum.

Neural networks are popular in machine learning because they have been responsible for many recent machine learning successes thanks to their function approximation capabilities. Function approximation is useful in reinforcement learning, as value and policy functions must often be estimated from samples.

Deep artificial neural networks use multiple hidden layers and weight connectivities (e.g., fully connected, convolutional, recurrent) to increases their function approximation capabilities. Deep reinforcement learning combines the powerful function approximation of deep neural networks with reinforcement learning algorithms, yielding agents that can perform complex tasks, sometimes better than humans, such as various video games [45, 46, 47, 48, 49], the game of GO [50, 51], image recognition [52] and robotics [53].

1.6 Research Methodology

The research objectives were achieved by using simulations tools, computational methods and quantitative metrics.

Like other safety critical systems, traffic signal controllers are developed using simulation tools because *in situ* development is dangerous - any failure during development would be disastrous. Simulations offer a safe environment to develop, allowing researchers to thoroughly verify and validate their work before real-world deployment. Many categories of traffic simulator exist. Macroscopic simulators model traffic at a high-level, describing vehicles in aggregate with density, speed and flow. Mesoscopic simulators provide more resolution by aggregating vehicles in small groups or platoons. Microscopic simulators provide the highest resolution, modelling vehicles as individual entities interacting. Macrosopic models require the least amount of computation at the cost of significant aliasing. Microscopic models offer the most accurate model of traffic at the cost of high computation. This research uses microscopic traffic simulation because it provides the most accurate model of real-world traffic. Individual vehicle dynamics are important when studying traffic signal controllers; macroscopic and mesocopic simulations provide an insufficiently detailed model, omitting these dynamics. Microscopic traffic simulations are composed of network geometry and vehicle dynamics models.

Network geometry includes all vehicle infrastructure (e.g., roads, intersections, traffic signal controllers). Vehicle dynamics are controlled by various sub-models. Vehicles are generated into the network and assigned a trip, a route from an origin to a destination in the network. Various routing algorithms are used to develop vehicle trips from origin to destination (e.g., Dijkstra). Vehicles use a car-following model to describe a vehicle's position and its derivatives with respect to other vehicles [54, 55]. The lane changing and gap acceptance models describe how a vehicle safely changes lanes when an appropriate gap exists. This thesis uses the microscopic traffic software Simulation of Urban Mobility (SUMO) [4]. SUMO offers ample tools to model custom or real-world network geometry, vehicle demand and traffic signal controllers. Like many software simulations, SUMO relies on pseudorandom number generation [56] for stochasticity.

To develop an adaptive traffic signal controller in simulation, machine learning methods are used. Machine learning, as a subfield of artificial intelligence, allows computers to solve problems by using data. Since transportation is quickly becoming a domain of big data, machine learning is appealing as it is a data driven problem solving method. Reinforcement learning, a type of machine learning, is an abstract family of techniques to develop goal-oriented behaviour in uncertain, sequential decision-making problems. A reinforcement learning agent interacts with an environment, learning a behaviour policy to achieve a pre-defined goal. Since traffic signal control can be modelled as a sequential decision-making problem, reinforcement learning is an obvious candidate solution.

Ample literature detailing reinforcement learning, adaptive TSC exists, displayed in Table 1. Initial research demonstrated positive proof-of-concept results for adaptive TSC using reinforcement learning [57, 58, 59, 60]. However, these initial works made many simplifications (e.g., vehicle dynamics, intersection control models) that obliged additional research. Subsequent research has attempted to improve beyond initial efforts, using more powerful learning algorithms and realistic simulation tools [61, 62, 63, 64, 65]. Still, these efforts lacked in areas such as function approximation and scale. Recently, deep reinforcement learning techniques, combining reinforcement learning with deep neural network function approximation, have been applied for adaptive traffic signal control with varying degrees of success [66, 67, 68, 69]. Deep reinforcement learning provides powerful function approximation at the cost of model complexity, potentially difficult development and model instability. Although considerable amounts of research attest to the simulated success of reinforcement learning adaptive TSC, few of these systems been deployed in the field.

Table 1: Reinforcement Learning Adaptive Traffic Signal Control related work.

Research	Method	Notes
[70]	Genetic RL	3x3 grid network, local RL, global genetic algorithm optimization
[57]	SARSA[1]	4x4 grid network, discrete state representation
[59]	Model-based RL	2x3 grid network, co-learning, agent communication
[60]	Q-learning	Isolated intersection, CMAC[2] function approximation
[71]	NeuroFuzzy	Isolated intersection, NN function approximation

[1] State-Action-Reward-State-Action (SARSA).

[2] Cerebellar Model Articulation Controller (CMAC).

Traditional, quantitative transportation performance indicators and measures of effectiveness are used for evaluating the proposed research [20]. These include mobility metrics such as vehicle travel and stopped time, and intersection metrics such as vehicle queue and delay. These metrics are used to compare and validate the proposed research to traditional solutions, such as fixed time, Webster's [72] and actuated TSC.

The methodology can be separated into two phases. First, developing and verifying proof-of-concept solutions and second, validating the solutions at scale.

The first phase of this research develops a proof-of-concept reinforcement learning, adaptive traffic signal controller in simple (i.e., isolated intersections) simulations. The answers to the specific research questions stated as part of the objectives will be developed in this phase. Reinforcement learning offers many methods, verifying which ones are best suited for adaptive traffic signal control requires careful analysis.

The second phase uses insights developed in the first phase to develop a scalable solution that will be validated under realistic simulation conditions (i.e., city scale). Techniques must be sought and developed that can transfer the findings from the first phase and make them useful in a scalable manner.

Using this systematic approach, an adaptive traffic signal control system was developed to optimize transportation, improve mobility and minimize costs.

2 Asynchronous Adaptive Traffic Signal Control

2.1 Introduction

Adaptive traffic signal controllers can improve transportation efficiency and reduce costs through optimal control. Reinforcement learning can be used for optimal adaptive traffic signal control, however existing methods can be computationally expensive and are not designed for scalability. In this chapter, n-step Q-learning is combined with parallel, asynchronous computation to develop a proof-of-concept acyclic adaptive traffic signal controller to overcome these challenges/gaps. In simulation, the proposed adaptive controller is developed and evaluated on an isolated intersection and compared against conventional traffic signal controllers and linear adaptive traffic signal controllers. The publication included in this chapter is:

W. Genders and S. Razavi, “Asynchronous n-step q-learning adaptive traffic signal control,” *Journal of Intelligent Transportation Systems*, 2018. doi:10.1080/15472450.2018.1491003. *Accepted, In Press*

The co-author’s contributions to the above work include:

- Financial and technical supervision of the study presented in this work.
- Manuscript editing.

2.2 Asynchronous n-Step Q-learning Adaptive Traffic Signal Control

2.3 Abstract

Ensuring transportation systems are efficient is a priority for modern society. Intersection traffic signal control can be modeled as a sequential decision-making problem. To learn how to make the best decisions, we apply reinforcement learning techniques with function approximation to train an adaptive traffic signal controller. We use the asynchronous n-step Q-learning algorithm with a two hidden layer artificial neural network as our reinforcement learning agent. A dynamic, stochastic rush hour simulation is developed to test the agent's performance. Compared against traditional loop detector Actuated and Linear Q-learning traffic signal control methods, our reinforcement learning model develops a superior control policy, reducing mean total delay by up 40% without compromising throughput. However, we find our proposed model slightly increases delay for left turning vehicles compared to the Actuated controller, as a consequence of the reward function, highlighting the need for an appropriate reward function which truly develops the desired policy.

2.4 Introduction

Society relies on its many transportation systems for the movement of individuals, goods and services. Ensuring vehicles can move efficiently from their origin to destination is desirable by all. However, increasing population, and subsequent vehicle ownership, has increased the demand of road infrastructure often beyond its capacity, resulting in congestion, travel delays and unnecessary vehicle emissions. To address this problem, two types of solutions are possible. The first is to increase capacity by expanding road infrastructure, however this can be expensive, protracted and decrease capacity in the short term (i.e., parts of the existing infrastructure are often restricted to traffic while construction occurs). The second solution is to increase the efficiency of existing infrastructure and the systems that govern them, such as traffic signal controllers (TSC). We advocate this second solution, by utilizing innovations from the domain of artificial intelligence [45, 46] to develop an adaptive traffic signal controller.

Developing a traffic signal controller means attempting to solve the traffic signal control problem (i.e., deciding what traffic phases should be enacted at any given time). Many solutions have been proposed to solve the traffic signal control problem, differing in complexity and assumptions made. The traffic signal control problem can be modeled as a sequential decision-making problem. Reinforcement learning, a machine learning paradigm, provides a framework for developing solutions to sequential decision-making problems. A reinforcement learning agent learns how to act (i.e., control the intersection phases) in an uncertain environment (i.e., stochastic vehicle dynamics) to achieve a goal (i.e., maximize a numerical reward signal, such as throughput). Rewards r are quantitative measures of how well the agent is achieving its goal. Described mathematically, given a discount factor $\gamma \in [0, 1)$, the agent seeks to maximize the sum of discounted future rewards $G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$. Formally, the agent develops a policy π which maps states to actions $\pi(s) = a$, in an attempt to maximize the sum of discounted future rewards. To achieve optimal control, we develop an optimal state-action policy π^*. We use valued-based reinforcement learning to estimate an action-value function, which tells us how "good" any action is in a given state based on the agent's prior experiences. We derive the optimal policy by developing an optimal action-value function $Q^*(s, a)$ defined in Equ. (4).

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\Big[G_t | s = s_t, a = a_t, \pi\Big] \tag{4}$$

Q-learning [38], a type of temporal difference learning [73], is used to estimate the optimal action-value function through repeated environment interactions, composed of the agent, at time t, observing the state of the environment s_t, choosing an action a_t, receiving feedback in the form of a reward r_t and observing a new state s_{t+1}. Accurately estimating an action's value allows the agent to select the action which will yield the most long-term reward.

However, traditional reinforcement learning can suffer from the curse of dimensionality, where developing tabular solutions becomes intractable due to a combinatorial explosion of the state-action space. This problem can be alleviated by combining traditional reinforcement learning with function approximators and supervised learning techniques. We approximate the optimal action-value function using an artificial neural network, defined in Equ. (5).

$$Q^*(s, a) \approx Q(s, a; \theta) \tag{5}$$

We develop the parameters θ (weights between neurons) of the neural network action-value function using a variant of the deep Q-network (DQN) algorithm [45], asynchronous n-step Q-learning [46]. Once sufficiently approximated, the optimal policy can be derived by selecting the action with the highest value using the developed action-value function, defined in Equ. (6).

$$\pi^*(s) = \text{argmax}_a Q(s, a; \theta) \tag{6}$$

Using the microtraffic simulator SUMO [4] as the environment, we ultimately produce an agent that can control the traffic phases, termed an n-step Q-network traffic signal controller (nQN-TSC), illustrated in Figure 5.

Other researchers have long noted reinforcement learning's potential in solving the traffic signal control problem and have applied it with varying degrees of success [74, 75, 64]. Recent deep

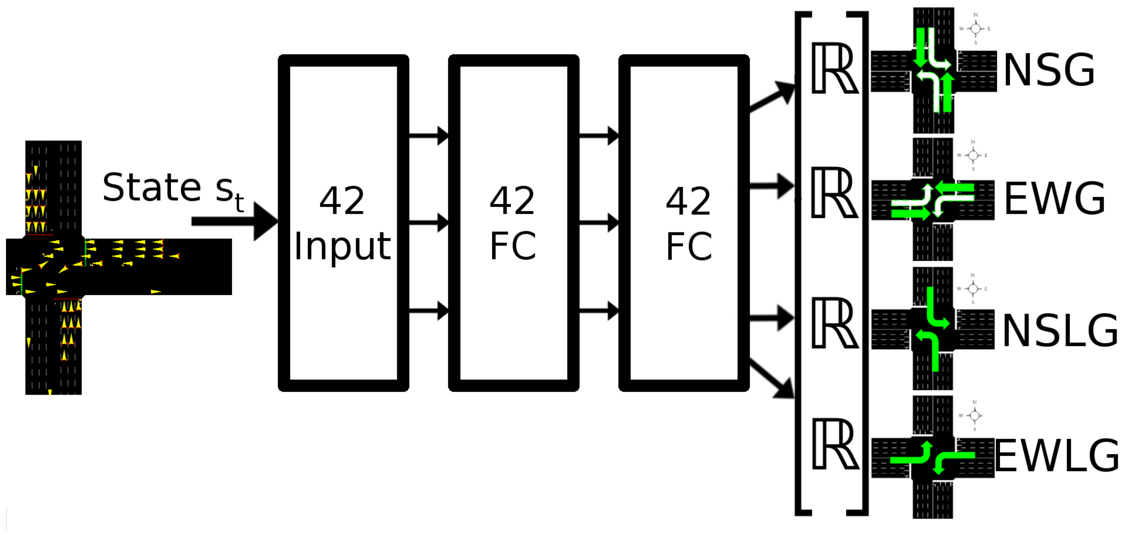


Figure 5: nQN-TSC model. First, the density and queue state s_t is observed by the agent, which is used as input to the neural network (Fully Connected (FC) layer of 42 neurons connected to another FC layer of 42 neurons), which produces a column vector of real numbers, each representing a different actions' value. Solid green arrows represent protected movements and white arrows represent permissive movements.

reinforcement learning techniques, successful in other domains, have provided further improvements over traditional reinforcement learning, traffic signal control models [66, 68, 67, 76, 77]. However, we observed subtle deficiencies in previous work which we address in this paper; specifically, the traffic signal control problem has been modeled too simply (discussed in the Literature Review) and must be confronted authentically. We argue these simplifications have created uncertainty as to whether reinforcement learning can truly be a contending solution for the traffic signal control problem compared against established solutions.

2.5 Literature Review

Intersection traffic signal control methods have been studied for decades. Traditionally, traffic signal phases are displayed in an ordered sequence to produce a cycle. If the phase durations and sequence are static, a fixed cycle traffic signal controller is produced and can be classified as a non-adaptive form of traffic signal control. Non-adaptive signal control methods are stable and relatively simple to create, yet may not be optimal as they are not influenced by current traffic conditions. With the advent of sensors (e.g., loop detectors, radar, cameras) at intersections, adaptive traffic signal controllers were possible. Adaptive traffic signal controllers can vary phase duration or phase sequence, producing dynamic phase durations and acyclic phase sequences. Adaptive methods can potentially improve performance as they react to current traffic conditions, but can be unstable, complex and difficult to develop.

We propose reinforcement learning as the appropriate method for traffic signal control because it offers specific advantages compared to other solutions. First, reinforcement learning utilizes the structure of the considered problem, observing what actions in certain states achieve reward [36]. This is in contrast to other optimization techniques, such as evolutionary methods (e.g., genetic algorithms [78]), which ignore the information yielded from individual environment interactions. Second, Q-learning is a model-free reinforcement learning algorithm, requiring no model of the environment once it has been sufficiently trained. Contrasted with many traffic signal control optimization systems (e.g., SCOOT [28], SCATS [30], RHODES [31]) which require traffic models, or control theory approaches [79, 80, 81] which use other models, such as backpressure, model-free reinforcement learning makes no assumptions about the model as no model is required. This arguably makes reinforcement learning more robust to traffic dynamics; model-based methods require their models accurately reflect reality. As the disparity between the model and reality increases, the performance of the system suffers. Model-free reinforcement learning is parsimonious compared to model-based methods, requiring less information to function.

Significant research has been conducted using reinforcement learning for traffic signal control. Early efforts were limited by simple simulations and a lack of computational power [57, 59, 82, 60]. Beginning in the early 2000's, continuous improvements in both of these areas have created a variety of simulation tools that are increasingly complex and realistic. Traffic microsimulators are the most popular tool used by traffic researchers, as they model individual vehicles as distinct entities and can reproduce real-world traffic behavior such as shockwaves. Research conducted has differed in reinforcement learning type, state space definition, action space definition, reward definition, simulator, traffic network geometry and vehicle generation model and can be separated into traditional reinforcement learning and deep reinforcement learning. Previous traditional reinforcement learning traffic signal control research efforts have defined the state space as an aggregate traffic statistic, the number of queued vehicles [59, 60, 83, 84, 85] and traffic flow [63, 74] the most popular. The action space has been defined as all available signal phases [63, 64] or restricted to green phases only [74, 83, 84]. The most common reward definitions are functions of delay [63, 64] and queued vehicles [74, 83, 84]. For a comprehensive review of traditional reinforcement learning traffic signal control research, the reader is referred to [86] and [87]. Bayesian reinforcement learning has also been demonstrated effective for adaptive traffic signal control [88, 89, 90, 65].

Deep reinforcement learning techniques were first applied to traffic signal control via the DQN in

[66]. The authors followed the example in [45] closely, defining the dimensions of the state space to be the same dimensions as in the Atari environment, meaning only a fraction of any state observation contained relevant information (which the authors identified and acknowledged as problematic). A uniform distribution was used to generate vehicles into the network, which is likely too simple a model of vehicles arriving at an intersection. Also, only green phases were modeled, meaning no yellow change or red clearance phases were used between green phases.

Subsequent work by [68] expanded on the ideas in [66], with the inclusion of vehicle speed and signal phases in the state observation and one second yellow change phases, Double Q-learning and limited experiments with multiple intersections. Both authors noted that while the agent's performance improved with training, there were concerns with the agent's learned policy, such as quickly oscillating phases and the inability to develop stable, long-term policies.

Research by [67] deviated from previous work by using deep stacked autoencoders to approximate the optimal action-value function. A disadvantage of stacked autoencoders is that each layer must be trained individually until convergence and then stacked on top one another, as opposed to DQN, which can be trained quicker in an end-to-end manner. The state observation was the number of queued vehicles in each lane over the past four seconds. The authors tested their agent on an isolated four way, two lane intersection where vehicles could not turn left or right, only travel through. This simplified the action space to two green phases. They also mandated a minimum green time of 15 seconds and did not model any yellow change or red clearance phases. The traffic demand for each intersection approach was $[100, 2\,000]$ veh/hr generated 'randomly', without detailing the specific probability distribution.

Research referenced thus far used valued-based reinforcement learning, which estimate value functions to develop the traffic signal control policy. Policy based methods are another type of reinforcement learning which explicitly parameterize the policy instead of using a value function to derive the policy. A traffic signal control policy was parameterized in [76] and [77]. Deep policy gradient was used in [76] to control dozens of traffic signal controllers in a large traffic network with impressive results. However, the model developed modified an existing traffic signal control policy instead of deriving its own in a truly autonomous manner. It seems this was a concession made so that multiple intersections could be controlled with the same agent, however it makes the implicit assumption that the optimal policy for each intersection is the same, which is unlikely. Regardless, the size and scope of the experiments which the agent was subjected, and its subsequent performance, is strong evidence for further work using policy-based reinforcement learning for traffic signal control.

2.6 Contribution

Considering the previous work as detailed, we identify the following problems and our contributions.

First, prior research has modeled the traffic signal problem with unrealistic simplifications. Specifically, either completely omitting yellow change and red clearance phases [66, 67], or including them for short durations [68]. Acknowledging that yellow and red phase durations can depend on many factors (e.g., intersection geometry and traffic laws), we contend that omitting yellow and red phases or including them for unrealistically short durations simplifies the traffic signal control problem such that any results are questionable. If reinforcement learning methods are to provide a solution to the traffic signal control problem, the modelling must include all relevant aspects as would be present in practice. We contribute a realistic model of the traffic signal control problem, with yellow and red phase durations of four seconds. Although the inclusion of these additional signals may seem slight, it changes how the agent's actions are implemented.

Second, we contend the only actions worth choosing by the agent are the green phases, as these are the actions which ultimately achieve any rational traffic signal control goal (i.e., maximize throughput, minimize delay or queue), assuming safety concerns have been satisfied. However, if green phases are the only actions available to the agent, depending on the current phase, some actions can not always be enacted immediately - yellow change and red clearance phases may be necessary. For example, there exist action sequences which are unsafe to transition from immediately, meaning the traffic signal phase can not immediately change to the agent's chosen action for safety reasons (i.e., the movements constituting these phases conflict). Some action sequences may require the traffic signal to enact a yellow change phase, instructing vehicles to slow down and prepare to stop, and then a red clearance phase, instructing vehicles to stop. Note that the agent does not explicitly select these transition traffic phases as actions, they are implicit when selecting actions that conflict with the previous action. Therefore, yellow change and red clearance phases are only included between sequences of conflicting actions.

Third, prior research has generated traffic using simple [66, 68], and likely unrealistic, models (e.g., uniform distribution vehicle generation). For reinforcement learning traffic signal control to be a contender in practice, it must be able to control intersections at any point on the demand spectrum, up to saturation demands. We train and test our agent under a dynamic rush hour demand scenario, authentically modelling the challenging environment as would be present in practice.

2.7 Proposed Model

Attempting to solve the traffic signal control problem using reinforcement learning requires a formulation of the problem using concepts from Markov Decision Processes, specifically, defining a state space S, an action space A and a reward R.

2.7.1 State Space

The state observation is composed of four elements. The first two elements are aggregate statistics of the intersection traffic - density and queue of each lane approaching the intersection. The density, $s_{d,l}$ is defined in Equ. (7) and the queue, $s_{q,l}$, is defined in Equ. (8):

$$s_{d,l} = \frac{|V_l|}{c_l} \tag{7}$$

Where V_l represents the set of vehicles on lane l and c_l represents the capacity of lane l.

$$s_{q,l} = \frac{|V_{q,l}|}{c_l} \tag{8}$$

Where $V_{q,l}$ represents the set of queued (i.e., stationary) vehicles on lane l.

The third and fourth elements encode information about the current traffic signal phase. These elements are a one-hot vector encoding the current traffic signal phase and a real-valued scalar encoding the time spent in the current phase. A one-hot vector is a binary valued, categorical encoding of qualitative properties. Therefore, the state space at an intersection with L lanes and a set of traffic phases P is formally defined as $S \in \left(\mathbb{R}^L \times \mathbb{R}^L \times \mathbb{B}^{|P|} \times \mathbb{R}\right)$. At time t, the agent observes the traffic state as $s_t \in S$.

Technologies over the last decade have made gathering information required for the proposed state space possible. Video cameras [91] are becoming more common as sensor devices at intersections and vehicles with wireless communication capabilities (i.e., Connected Vehicles [92]) are expected to be deployed in the near future.

2.7.2 Action Space

The actions available to the agent are the green phases that the intersection traffic signal controller can enact. Although traditionally denoted numerically with natural numbers starting at 1 (i.e., National Electrical Manufacturers Association (NEMA) standard), in this research, to reduce ambiguity, we denote phases by their compass directions and movement priority. Beginning with the four compass directions, North (N), South (S), East (E) and West (W) and combining the parallel

Table 2: Traffic Signal Phase Information

			Turning Movements		
Action	NEMA Phases	Compass Directions	Left	Through	Right
NSG	2, 6	North, South	Permissive	Protected	Permissive
NSLG	1, 5	North, South	Protected	Prohibited	Permissive
EWG	4, 8	East, West	Permissive	Protected	Permissive
EWLG	3, 7	East, West	Protected	Prohibited	Permissive

Table 3: Traffic Signal Phase Action Transitions

		Selected Action a_t			
		NSG	EWG	NSLG	EWLG
Current Traffic Signal Phase	NSG	-	{NSY, R}	{NSLY}	{NSY, R}
	EWG	{EWY, R}	-	{EWY, R}	{EWLY}
	NSLG	-	{NSY, R}	-	{NSY, R}
	EWLG	{EWY, R}	-	{EWY. R}	-

directions, along with the two different lane movements which allow vehicles to traverse the intersection, through green (G) and advanced left green (LG), the four possible actions are North-South Green (NSG), East-West Green (EWG), North-South Advance Left Green (NSLG) and East-West Advance Left Green (EWLG). For any of the action phases, all other compass direction traffic movements are prohibited (i.e., East-West Green phase imply all North-South signals are red), except for right turns, which are permissive (i.e., right on red). Formally the set of all possible actions A is defined as $A = \{\text{NSG, EWG, NSLG, EWLG}\}$, with additional information available in Table 2. Therefore, at time t, the agent chooses an action a_t, where $a_t \in A$.

However, when an agent chooses an action, it may not be immediately enacted. To ensure safe control of the intersection, additional phases may precede the chosen action. Instead of immediately transitioning from the current traffic signal phase to the selected action, a sequence of yellow change and red clearance phases may be required, dependent on the current phase and chosen action. All possible action transition sequences to transition from the current traffic phase to the chosen action phase are shown in Table 3. Note the addition of the North-South Yellow (NSY), East-West Yellow (EWY), North-South Advance Left Yellow (NSLY), East-West Advance Left Yellow (EWLY) change and red clearance (R) phases, which cannot be chosen explicitly as actions, but are part of some phase transition sequences. The set of all traffic signal phases, actions and transition phases, is defined as $P = \{\text{NSG, EWG, NSLG, EWLG, NSY, EWY, NSLY, EWLY, R}\}$.

2.7.3 Reward

The final element of reinforcement learning, after the agent has observed the state of the environment s_t, chosen an action a_t, and performed it, is receiving the reward r_t. Reward is one element that differentiates reinforcement learning from other types of machine learning; the agent seeks to develop a policy which maximizes the sum of future discounted rewards. Compared to supervised learning, in which correct actions are given by instruction, reinforcement learning has the agent evaluate actions by interacting with the environment and receiving reward.

In the context of traffic signal control, various rewards have been successfully used. Examples of rewards used are functions based on queue, delay and throughput [86], however there is yet no consensus on which is best.

We define the reward r_t as a function of the number of queued vehicles (9).

$$r_t = -(|V_q^t|^2) \tag{9}$$

Where V_q^t is the set of queued vehicles at the intersection at time t. The number of queued vehicles is squared to increasingly penalize actions that lead to longer queues. Since reinforcement learning agents seek to maximize reward, the negative number of queued vehicles squared is used, rewarding the agent for minimizing the number of queued vehicles.

2.7.4 Agent

In reinforcement learning, the agent is the entity that learns by interacting with the environment. There is a single agent governing the traffic signal controller, exclusively responsible for selecting the next green traffic phase from the set of possible green phases A. We model the agent controlling the traffic signals similar to a DQN [45]. A DQN is a deep artificial neural network trained using Q-learning, a valued-based reinforcement learning algorithm. Artificial neural networks are mathematical functions inspired by biological neural networks (i.e., brains) that are appealing for their function approximation capabilities. Many problems in machine learning can suffer from the "curse of dimensionality", which is when the dimensionality of the data increases, the training and computational resources required grow exponentially. Artificial neural networks have the capability to generalize from what they have learned, weakening the problems posed by the curse of dimensionality.

Artificial neural networks are defined by their architecture and weight parameters θ. Typically, neural networks are composed of at least one hidden layer of computational neurons. Additional hidden layers in a network allow it to develop high level representations of its input data through multiple layers of function composition, potentially increasing performance. The nQN-TSC is implemented as a two hidden layer artificial neural network. The input to the nQN-TSC is the state observation s_t outlined in 2.7.1. The two hidden layers are fully connected with 42 neurons and are each followed by rectified linear activation units (ReLu). The output layer is composed of four neurons with linear activation units, each representing the action-value of a different action (i.e., green phase).

The optimal policy π^* is achieved by using the neural network to approximate the optimal action-value function by changing the parameters θ through training. The action-value function maps states to action utilities (i.e., what is the value of each action from a given state). Values represent long-term reward. If an action has a high value (and is accurately estimated), enacting it means reaping future reward. Choosing actions with the highest value will yield the optimal policy.

The specific reinforcement learning algorithm used in this research is asynchronous n-step Q-learning [46], a variant of DQN [45]. Asynchronous n-step Q-learning differs from DQN in two fundamental ways. First, instead of a single agent interacting with a single environment, the algorithm leverages modern multi-core computer processors to simulate multiple agent-environment pairs in parallel, each agent-environment pair on a CPU thread maintaining their own local parameters θ'. The agents collect parameter updates $d\theta$, using their local parameter set θ', that are used to asynchronously update a global parameter set θ. Each agent periodically copies the global parameter set to its local parameter set every I actions. Utilizing multiple agents improves learning, as each agent is likely experiencing different environment states, taking different actions, and receiving different rewards. Second, instead of updating the parameters θ after each action, the agent executes n actions and uses data from all n actions to improve reward (and value) estimates. The asynchronous n-step Q-learning algorithm pseudo code is presented in Algorithm 1 and the model hyperparameters are presented in Table 4. Note the inclusion of reward normalization by the maximum reward r_{max} experienced thus far across all agents.

The RMSProp [93] optimization algorithm is used to train the network with an initial learning rate α of 0.001.

The agent employs an action repeat of 10 for improved stability (i.e., actions/green phases have a duration of 10 seconds). Yellow change and red clearance phases have a duration of four seconds. If at least one vehicle is on an incoming lane to the intersection, the agent must select an action. However, if no vehicles are present on incoming lanes, this is considered the terminal state $s_{terminal}$, the agent does not select an action and the traffic phase defaults to the red clearance (R) phase.

To select actions during training, we implement the simple, yet effective, ϵ-greedy exploration policy, which selects a random action (explore) with a probability ϵ and selects the action with the highest value (exploit) with a probability 1-ϵ. The value of ϵ decreases as training progresses according to (10).

$$\epsilon_m = 1.0 - \frac{m}{M} \tag{10}$$

Where m is the current number of completed actions and M is the total number of actions. Initially, $m = 0$, meaning the agent exclusively explores, however, as training progresses, the agent increasingly exploits what it has learned, until it exclusively exploits.

2.8 Experimental Setup and Training

All experiments were conducted using the traffic microsimulator SUMO v0.29 [4]. SUMO provides a Python application programming interface (API), by which custom functionality can be implemented in the traffic simulation. We used the SUMO Python API and custom code to implement all experiment code. The nQN-TSC was implemented using Tensorflow v1.2.1 [94] with additional code from [95]. Additional optimized functionality was provided by NumPy and SciPy libraries [96]. The simulations were executed using eight parallel agents on a desktop computer with an i7 − 6700 CPU, 16 GB of RAM running Ubuntu 16. To train the nQN-TSC for $M = 1.5 \times 10^6$ actions requires approximately 6 hours of wall time.

The intersection geometry is four lanes approaching the intersection from every compass directions (i.e., North, South, East and West) connected to four outgoing lanes from the intersection. The traffic movements for each approach are as follows: the inner lane is left turn only, the two middle lanes are through lanes and the outer lane is through and right turning. All lanes are 300 meters in length, from the vehicle origin to the intersection stop line. The state observations are derived from the stop bar to 150 meters away from the intersection (i.e., only vehicles 150 meters or closer to the intersection are considered in the state). A visual representation can be seen in Fig 6.

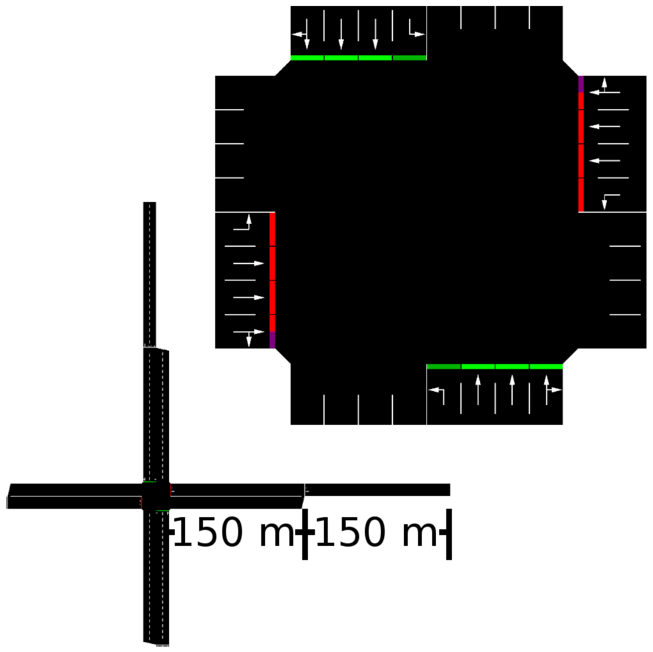


Figure 6: Simulation model of intersection, arrows represent lane movements.

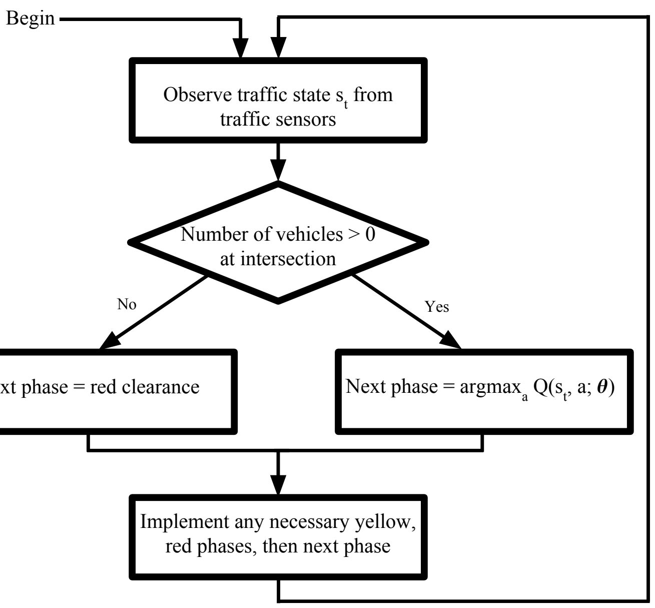

Begin
Observe traffic state s_t from traffic sensors
Number of vehicles > 0 at intersection
No
Yes
Next phase = red clearance
Next phase = $\text{argmax}_a\, Q(s_t, a; \boldsymbol{\theta})$
Implement any necessary yellow, red phases, then next phase

Figure 7: Block diagram describing main steps for real world deployment. The neural network $Q(s_t, a; \theta)$ is first developed in simulation and then used to determine the next phase using traffic sensor data.

Algorithm 1: nQN-TSC Parallel Training pseudo code

```
$m \leftarrow 0$, Initialize $\theta, \theta^{'}$
**While** $m < M$
  $t \leftarrow 0$, Generate vehicle demands
  **While** $t < T$
    Reset parameter gradients $d\theta \leftarrow 0$
    Set thread parameters to global $\theta' = \theta$
    $t_{start} = t$
    Observe state $s_t$
    **While** $s_t \neq s_{terminal}$ **and** $t - t_{start} \neq t_{max}$
      **If** $\text{Unif}(0, 1.0) > \epsilon_m$
        $a_t = \text{argmax}_a Q(s_t, a; \theta')$
      **Else**
        $a_t = \text{Unif}(A)$
      Receive reward $r_t$ and observe new state $s_{t+1}$
      **If** $r_t > r_{max}$
        $r_{max} = r_t$
      $r_t = \frac{r_t}{r_{max}}$
      $s_t = s_{t+1}$
      $t \leftarrow t + 1$
      $m \leftarrow m + 1$
      **If** $m \text{ mod} I == 0$
        $\theta' \leftarrow \theta$
    **If** $s_t == s_{terminal}$
      $G = 0$
    **Else**
      $G = \max_a Q(s_t, a; \theta')$
    **For** $i \in \{t-1, t-2, ..., t_{start}\}$
      $G \leftarrow r_i + \gamma G$
      Collect gradients $d\theta \leftarrow d\theta + \frac{\partial (G - Q(s_t, a; \theta'))^2}{\partial \theta'}$
    Asynchronously update $\theta$ with $d\theta$
```

For training the nQN-TSC, we subject it to different traffic simulations. We seek to model the dynamic nature of traffic at an intersection, which changes over time. Therefore, we model a rush hour or peak traffic demand scenario. Initially, the traffic demand is at a minimum, then it begins

Table 4: Model Hyperparameters

Variable	Parameter	Value
$\|s_t\|$	State Cardinality	42
c_l	Lane Capacity	20 veh
T	Traffic Simulation Length	7200 s
t_{max}	Max Experience Sequence/n-step	4
M	Total Training Actions	1.5×10^6
I	Update Interval	7500
γ	Discount Factor	0.99

increasing until it reaches a maximum, where it remains for an hour, after which it begins decreasing and then returns to a constant minimum. This training procedure is devised to create a dynamic training environment. We seek to make solving the traffic signal control problem in simulation as challenging as would be faced in practice - dynamic and stochastic. To ensure these qualities, we randomly translate the traffic demand in time at the beginning of each training simulation, displayed in Figure 8. The demands are translated to vary training and to ensure the agent doesn't overfit the training data. Vehicles are generated using a negative exponential distribution, where the rate parameter λ is used to sample from a normal distribution $\mathcal{N}(\lambda, \frac{\lambda}{10})$ to stochastically generate vehicles. This procedure is used to ensure training simulations are similar but not the same, exposing the agent to a diversity of traffic states. Training is completed after the agent has executed $M = 1.5 \times 10^6$ actions. A block diagram describing the use of the proposed model after training is displayed in Figure 7.

Vehicles are generated into the network from each compass direction with uniform probability. Vehicle routing is proportional to turning movement capacity. Vehicles are randomly assigned a route with a probability proportional to the route's lane capacity (i.e., through movements are more likely than turning movements as there are more lanes from which through movements are possible). Given the lane movements defined earlier in the section, The probability a vehicle assigned a through movement is $\frac{3}{5}$ and $\frac{1}{5}$ each for left and right turning movements.

2.9 Results and Discussion

To evaluate the performance of the nQN-TSC after training, untranslated rush hour demands are used and its performance is compared against three other TSC methods; Random, Actuated and Linear Q-learning.

The Random TSC method enacts each action with a uniform probability (i.e., $\pi(s_t)_{Rand} =$

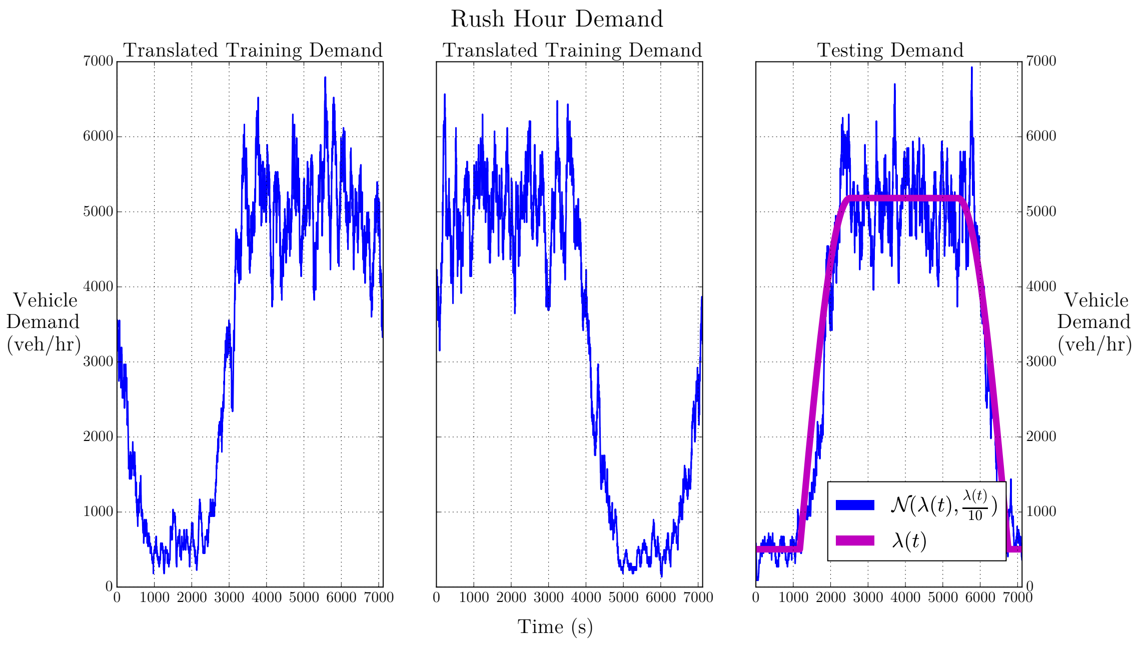


Figure 8: Examples of randomly generated simulation rush hour traffic demand. The demands are randomly translated in time for use during training while the untranslated rush hour demand is used during testing.

Unif(A)). A random action policy is extremely naive with respect to optimality, however, it provides a baseline from which to compare the progress of nQN-TSC.

The actuated method is a fixed phase sequence, dynamic phase duration TSC. Sensors are used to modify the green phase durations, making this method adaptive. Each green phase has a minimum duration (10 seconds), after which a timer begins decrementing from a gap-out time (5 seconds). If a vehicle is sensed in a green phase lane (i.e., the lane currently has a permissive or protected movement given the current phase) while the timer is decrementing, the timer is reset and the current phase is temporarily extended. Only a finite number of extensions can occur, limited by a maximum extension time (40 seconds). The sensors modeled in simulation are loop detectors (one per lane, 50 meters from the stop line), which trigger when a vehicle traverses them.

The Linear Q-learning method is similar to the proposed nQN-TSC, selecting the next phase in an *ad-hoc* manner, except that it uses a linear combination of features instead of a neural network for function approximation. A linear combination of features function approximator can be thought of as a neural network without hidden layers and activation functions. Each action has its own value function, as a linear combination of features can not approximate all action-values simultaneously. The features used by Linear Q-learning are the same as the input layer to the nQN-TSC, a combination of the lane queue, density and phase information. The Linear Q-learning method does not use asynchronous learning (i.e., only one actor and one environment) or n-step returns, instead computing the traditional action-value target update (i.e., 1-step return). Gradient descent is used to train the Linear Q-learning action-value functions using the same model hyperparameters as the nQN-TSC. To train the Linear Q-learning method requires approximately 24 hours of wall time.

Total vehicle delay D (11) and throughput F (12) traffic data are collected for comparing TSC methods.

$$D = \sum_{t=0}^{T} \sum_{v \in V^{t}_{\text{arrive}}} d_v \tag{11}$$

Where T is the total number of simulation steps, V^{t}_{arrive} is the set of vehicles at time t that arrive at their destination (i.e., are removed from the simulation) and d_v is the delay experienced by vehicle v.

$$F = \sum_{t=0}^{T} |V^{t}_{\text{arrive}}| \tag{12}$$

Where V^{t}_{arrive} is the set of vehicles at time t that arrive at their destination (i.e., are removed from the simulation) and T is the total number of simulation steps.

To estimate relevant statistics, each TSC method is subjected to 100 two hour simulations at testing rush hour demands. Results from the testing scenarios are shown in Table 5 and Figure 9,

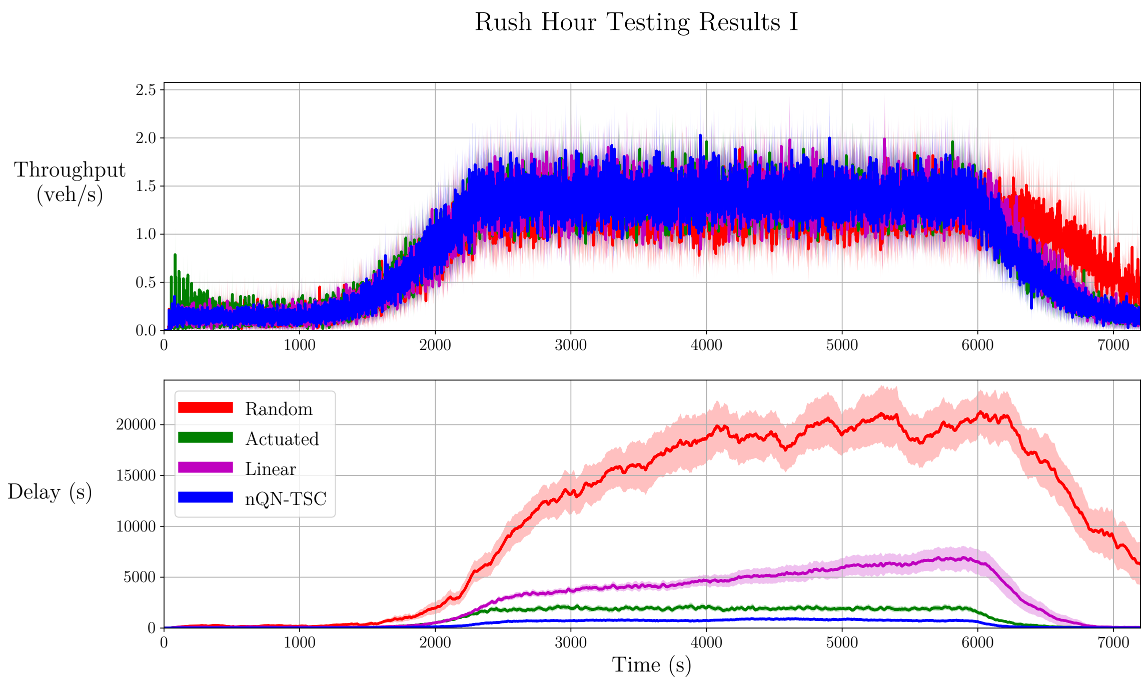


Figure 9: Performance of TSC methods under 100, two hour rush hour simulations. Lines represent the mean and the shaded area is the 99% confidence interval. Note that the throughput is influenced by the traffic demand. Traffic demand begins low, increases to a maximum, remains constant, and then decreases at the end of the testing scenario.

10 and 11.

As expected, the Random TSC achieves the worst performance, with the lowest throughput and highest delay.

Comparing the Actuated, Linear and nQN-TSC, there is no significant difference in throughput, however the nQN-TSC achieves the lowest delay. Interestingly, Linear Q-learning produces higher delay than the Actuated TSC. This is likely evidence that the use of a linear combination of feature as a function approximator has insufficient capacity to model the action-value function for the traffic signal control problem. Although the nQN-TSC and Linear Q-learning methods share many hyperparameters, the additional layers and non-linear transformations of the neural network seem to allow it to more accurately estimate the optimal action-value function, leading to better performance.

We find it interesting that the same number of vehicles are traversing the intersection during the rush hour under the Actuated, Linear and nQN-TSC, but there are significant differences in delay. We conjecture there is a strong correlation between vehicle queues and delay, and since the nQN-

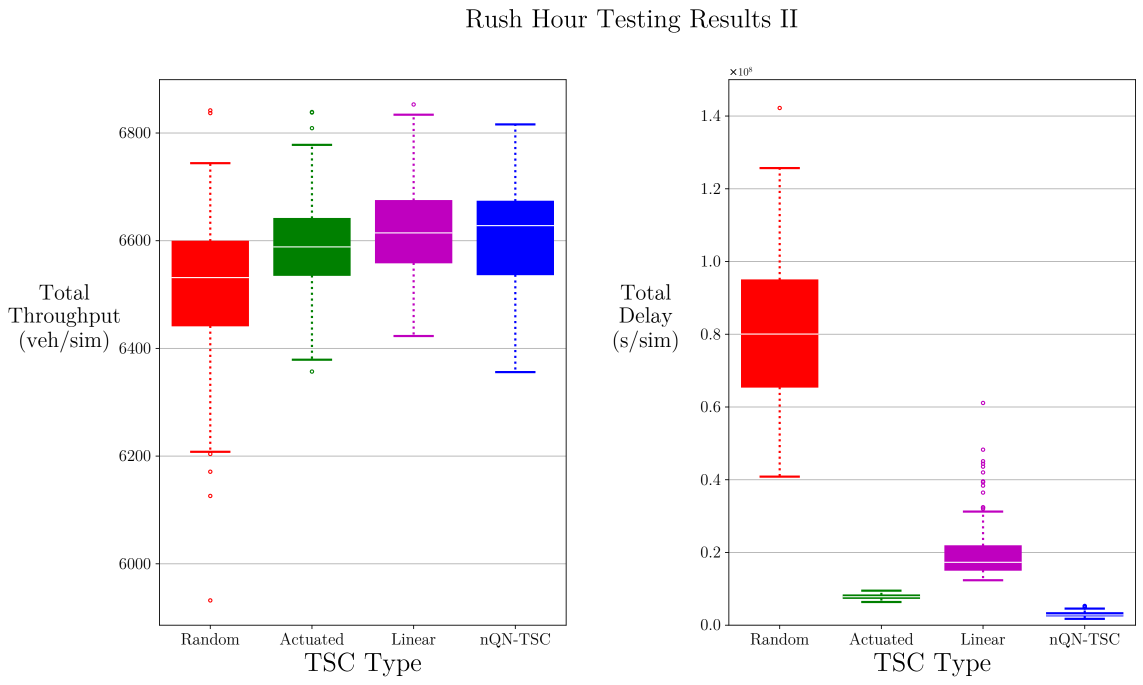


Figure 10: Total throughput and delay for individual simulation runs. Coloured rectangles represent the range between the first and third quartiles, solid white line inside the rectangle represents the median, dashed coloured lines represent 1.5 x interquartile range, coloured dots represent outliers.

TSC seeks to minimize the queue, it is also minimizes vehicle delay, as queued vehicles are delayed vehicles (i.e., both are stationary). The performance disparity can be attributed to the flexibility the nQN-TSC has in selecting the next green phase. The nQN-TSC does not need to adhere to a particular phase cycle, it operates acyclically. Every action (i.e., green phase) chosen is in an attempt to maximize reward. The Actuated TSC does not exhibit such goal-oriented behaviour, instead executing primitive logic that produces inferior performance compared to the nQN-TSC. Granted, the Actuated TSC does not have the benefit of training like the nQN-TSC, but its design precludes it from being of any benefit even if it was available. The information contained in the nQN-TSC's state representation is also much higher compared to the Actuated TSC, improving its action selection. The Actuated TSC's state representation is a bit for every lane and this low-resolution state likely aliases many different traffic states. These results provide evidence to support that adaptive traffic signal control can be accomplished with reinforcement learning and function approximation.

We also present individual vehicle delay results for each lane/turning movement in Figure 11. We observe that although in aggregate the nQN-TSC achieves the lowest total delay results, this observation doesn't exist for all lanes. For the through and right lanes, the nQN-TSC exhibits the lowest median delay, quartiles and outliers compared to the other TSC methods. However, observing the left lane delay, the Actuated method achieves lower delay than the nQN-TSC, specifically in regards to outliers, as the nQN-TSC has significant outliers which indicate some left turning vehicles have to wait a disproportionate amount of time to turn left compared to other vehicles. This result can be understood by considering the reward function, which seeks to minimize the squared number of queued vehicles. The intersection geometry used in this research has, in any single compass direction, three incoming lanes which allow through movements while only one lane for left turn movements. The nQN-TSC has discovered that selecting the actions/phases which give protected green signals to through movements (i.e., NSG and EWG) yield the highest reward. The nQN-TSC learns to favour these actions over the exclusive left turning actions/phases (i.e., NSLG, EWLG), because they yield less reward from having fewer queued vehicles potentially traversing the intersection when selected.

The nQN-TSC's behaviour is contrasted with the Actuated method, which implements a cycle and reliably enacts exclusive left turning phases, yielding lower delays for vehicles in left turning lanes. This is an example of a reinforcement learning agent developing a policy which seems optimal with respect to the reward, but with undesirable, unintended consequences. This behaviour could be corrected with a different reward function for the nQN-TSC, but perhaps at the expense of its desirable performance in the other metrics (e.g., total delay, through lane delay, right lane delay).

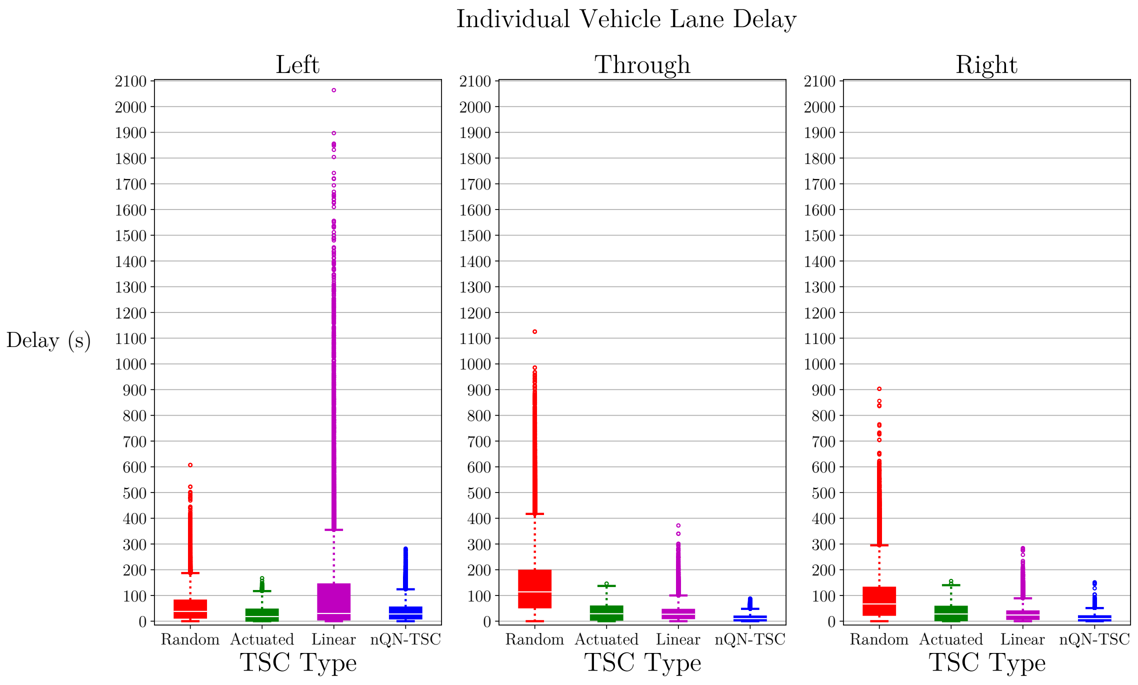


Figure 11: Individual vehicle delay by intersection lane movement and TSC type. Coloured rectangles represent the range between the first and third quartiles, solid white line inside the rectangle represents the median, dashed coloured lines represent 1.5 x interquartile range, coloured dots represent outliers. Performance of TSC methods under 100, two hour rush hour simulations.

This is an interesting result that we have not observed discussed in the TSC reinforcement learning literature and should be the focus of future research.

2.10 Conclusion and Future Work

We modeled an n-step Q-network traffic signal controller (nQN-TSC) trained to control phases at a traffic intersection to minimize vehicle queues. Compared to a loop detector Actuated TSC in a stochastic rush hour demand scenario, the modeled nQN-TSC achieve superior performance by reducing average total vehicle delay by 40%. This research provides evidence that valued-based reinforcement learning methods with function approximation can provide improved performance compared to established traffic signal control methods in stochastic environments, such as rush hour demands.

Beyond this research, many areas of future work are available. The function approximator used in this research was simple (i.e., two fully connected hidden layer neural network) compared

Table 5: Traffic Signal Control Rush Hour Results

	$(\hat{\mu}, \hat{\sigma})$	
Traffic Signal Control Method	Total Throughput (veh/sim)	Total Delay (s/sim)
Random	$(6\,508, 151)$	$(81 \times 10^6, 20 \times 10^6)$
Actuated	$(6\,594, 95)$	$(7.8 \times 10^6, 0.7 \times 10^6)$
Linear	$(6\,618, 91)$	$(20 \times 10^6, 9.1 \times 10^6)$
nQN-TSC	$(6\,609, 93)$	$(3.0 \times 10^6, 0.8 \times 10^6)$

to research in other domains; using convolutional or recurrent layers in the neural network will likely yield additional performance improvements. Future work can also investigate policy or actor-critic reinforcement learning algorithms, different state representations, pedestrians and multiple intersections. These ideas will be explored in future endeavors by the authors.

3 Policy-based Adaptive Traffic Signal Control

3.1 Introduction

Although n-step Q-learning was successfully used to develop an adaptive traffic signal controller in the previous chapter, reinforcement learning algorithms which directly develop a policy have demonstrated superior performance in other domains. This chapter extends the work from the previous chapter by developing an acyclic adaptive traffic signal controller using the asynchronous advantage actor-critic (A3C) algorithm. Policy-based reinforcement learning methods exhibit certain advantages over value-based methods (i.e., Q-learning), such as the ability to develop stochastic policies. The A3C adaptive traffic signal controller is compared against the asynchronous n-step Q-learning controller developed in the previous chapter and conventional traffic signal control methods. Although trained on an isolated intersection, all controllers are evaluated on a multi-intersection simulation network. The A3C controller demonstrates the best performance in a multi-intersection network despite independent training and execution. The learned policies of the various controllers are analyzed to understand the difference in performance. The submitted manuscript included in this chapter is:

W. Genders and S. Razavi, "Policy analysis of reinforcement learning adaptive traffic signal control," *ASCE Journal of Computing in Civil Engineering*, 2018. *4th round of revisions*, CPENG-2667R4

The co-author's contributions to the above work include:

- Financial and technical supervision of the study presented in this work.
- Manuscript editing.

3.2 Policy Analysis of Reinforcement Learning Adaptive Traffic Signal Control

3.3 Abstract

Previous research studies have successfully developed adaptive traffic signal controllers using reinforcement learning; however, few have focused on analyzing what specifically reinforcement learning does differently than other traffic signal control methods. This research proposes and develops two reinforcement learning adaptive traffic signal controllers, analyzes their learned policies and compares them to a Webster's controller. The asynchronous Q-learning and advantage actor-critic adaptive algorithms are used to develop reinforcement learning traffic signal controllers using neural network function approximation with two action spaces. Using an aggregate statistic state representation (i.e., vehicle queue and density), the proposed reinforcement learning traffic signal controllers develop the optimal policy in a dynamic, stochastic traffic microsimulation. Results show that the reinforcement learning controllers increases lost time but ultimately achieve superior performance compared to the Webster's controller, reducing mean queues, stopped time and travel time. The reinforcement learning controllers exhibit goal-oriented behaviour, developing a policy which which excludes many phases found in a tradition phase cycle (i.e., protected turning movements) instead choosing phases which maximize reward, as opposed to the Webster's controller, which is constrained by cyclical logic that diminishes performance.

3.4 Introduction

Researchers have long noted reinforcement learning's (RL) suitability as a solution for traffic signal control (TSC), producing an abundance of literature [86, 87, 97]. However, insufficient attention has been given to determining what RL TSC does differently than traditional TSC (e.g, pre-timed, Webster's, loop-detector Actuated). Ample research exists attesting to the success and benefits of RL TSC, however the question of what a TSC trained using RL is doing differently has largely been ignored. The answer to this question is important, as it can aid researchers' in comprehending RL's strengths and weaknesses in the domain of TSC, highlighting future areas of inquiry. The authors' investigate this issue by modelling an authentically challenging traffic simulation scenario and observing the behaviour of the RL TSC. This research models two adaptive TSC using asynchronous RL algorithms, asynchronous Q-learning (AQ) and asynchronous advantage actor-critic (A3C) [46], analyzes their learned behaviours and compares them to a Webster's TSC.

3.5 Literature Review

Traffic signal controllers can be broadly classified into pre-timed and adaptive. Pre-timed methods do not consider the current state of traffic at the intersection, exhibiting fixed phase and cycle durations that repeat the same phase sequence. The *de facto* pre-timed TSC method is Webster's method [72], which relies on recurring traffic demand often estimated from historical data. Adaptive methods come in many varieties, however they all use sensor data to dynamically control the intersection. Adaptive traffic signal controllers can dynamically change the cycle duration, phase duration or phase sequence to improve safety and mobility. Pre-timed methods are predictable, stable and simple to develop, yet may not be optimal, while adaptive methods strive for optimality at the cost of being more complex, potentially unstable and expensive.

Early RL TSC research used simple simulations but offered evidence that RL could be a contending solution for TSC [57, 59, 82, 60]. Improved simulation tools have allowed RL TSC to be applied to large simulations with impressive results [64, 84, 98, 99, 100]. Recently, value-based deep RL methods have been proposed by various authors with varying degrees of success [66, 68, 67, 69]. Extensive reviews of RL TSC have been developed comprehensively outlining previous research [86, 87, 97].

The majority of past RL TSC research have used value-based methods, such as Q-Learning [38]. Policy-based RL has been investigated less for TSC, with the few instances described below.

Some of the earliest policy-based TSC work was completed using the natural actor-critic algorithm [101, 102]. The authors modelled the TSC problem as a partially observable Markov decision process. Their proposed traffic signal controller selected the next phase ever 5 timesteps in an attempt to maximize vehicle throughput. They compared their two policy-based methods against uniform pre-timed and non-RL adaptive traffic signal controllers, in which their proposed policy-based methods reduced travel times in a variety of traffic scenarios. However, as the authors acknowledge, the traffic simulator emphasized simulator speed instead of realism, which they recommended should be addressed in future work.

A radial basis artificial neural network was used to model a fuzzy actor-critic traffic signal controller [103]. The authors' modelled an isolated intersection (2 lanes per direction) and showed their fuzzy actor-critic traffic signal controller performed better than a pre-timed traffic signal controller.

Other research has developed both policy and value-based methods for TSC [75]. The authors' proposed algorithms attempted to minimize a long-run average cost function which is a function of queue lengths and vehicle waiting times. They test their proposed traffic signal controllers on a few network configurations and report that the policy gradient actor-critic algorithm achieves the best performance.

The asynchronous advantage actor-critic algorithm was used to train a traffic signal controller given a simulated image snapshot of the intersection [77]. Their proposed method was able to improve its policy over time, however the network geometry (1 lane each direction) and vehicle generation method (uniform distribution) were simplified compared to reality.

Deep deterministic actor-critic policy gradient methods were used to to develop a signal controller specifically for controlling large-scale traffic networks (i.e., many traffic signal controllers) [76]. This research focused on practical implementation and assumed data available from loop detectors as input for deriving the state observation. The actions were constrained to scaling the phase durations in a fixed cycle, instead of dynamically deriving the phase sequence, to ensure coordination between intersections was maintained. The author also proposed a disaggregated reward instead of a single scalar reward, which they claim is the first of its kind in RL. Their method achieves superior performance on a simulation of a large urban area with dozens of intersections compared against a multi-agent Q-learning traffic signal controller and random timings.

3.6 Reinforcement Learning

Controlling traffic signals at an intersection can be modelled as a sequential decision-making problem. Reinforcement learning is a type of machine learning that can develop solutions to sequential decision-making problems. A RL agent learns how to act in an environment to maximize a numerical reward signal [36], representing success at achieving a desired goal. At time t, the agent observes the environment state $\mathbf{s_t} \in \mathbf{S}$ and then chooses an action $a_t \in A$ conditioned on $\mathbf{s_t}$. After taking action, the agent receives feedback in the form of numerical reward r_t. The agent seeks to develop a policy π mapping from states to actions $a = \pi(\mathbf{s})$. The agent's ultimate goal is to develop an optimal policy π^* which maximizes cumulative discounted future rewards, known as the return G_t, defined in Equation 13.

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{13}$$

Where $\gamma \in (0, 1]$ is the discount factor, which influences how the reward k state-action transitions after t contributes to the return.

Q-learning is a value-based RL method which develops an action-value function defined by parameters θ, defined in Equation 14. An action-value function can be used to estimate the expected return starting from a given state $\mathbf{s}$ and taking action a and following policy π thereafter.

$$Q^{\pi}(\mathbf{s}, a; \theta) = \mathbb{E}[G_t | \mathbf{s_t} = \mathbf{s}, a_t = a; \theta] \tag{14}$$

To implement the optimal policy using Q-learning, the action-value function is used by selecting the action in the current state with the highest value, defined in Equation 15.

$$\pi^*(\mathbf{s}) = \text{argmax}_a Q(\mathbf{s}, a; \theta) \tag{15}$$

Policy-based RL methods attempt to directly approximate the optimal policy $\pi^*(a|s) \approx \pi(a|s;\theta)$ with a set of parameters θ [40]. Directly developing the policy has advantages compared to value-based RL methods, such as the ability to develop stochastic policies, shown in Equation 16.

$$\pi(a|\mathbf{s}; \theta) = \Pr[a_t = a | \mathbf{s_t} = \mathbf{s}; \theta] \tag{16}$$

Actor-critic RL methods represent both a value and policy function, estimating a value function to update the policy function [36, 104]. A parameterized state value function can be used to estimate the expected return starting from a given state s and following a policy π, defined in Equation 17.

$$V^{\pi}(\mathbf{s}; \theta) = \mathbb{E}[G_t | \mathbf{s_t} = \mathbf{s}; \theta] \tag{17}$$

3.7 Model

This section describes the traffic simulation and TSC elements in the framework of RL.

3.7.1 Traffic Simulation

To train the RL TSC agent, a microscopic traffic simulator is used as the environment. The traffic simulation is dichotomized into network geometry and vehicle dynamics.

The network geometry modelled is an isolated intersection with four origin destination zones. Vehicles are generated into the network from an origin zone and exit the simulation from a destination zone. Each zone is denoted with one of the four compass directions, North (N), South (S), East (E) and West (W), forming the set C ={N, E, S, W}. The intersection is composed of eight roads of length 300 m, four incoming roads from each of the origin zones to the intersection, and four outgoing roads from the intersection to each of the destination zones. Each road is composed of 5 lanes with each lane allowing different vehicle movements at the intersection. The rightmost lane allows through and right turning movements, the middle two lanes allow through movements and the two leftmost lane allow left turning movements. Incoming lanes are denoted with an l and subscript consisting of a compass direction and integer from $[0, 4]$. The compass direction

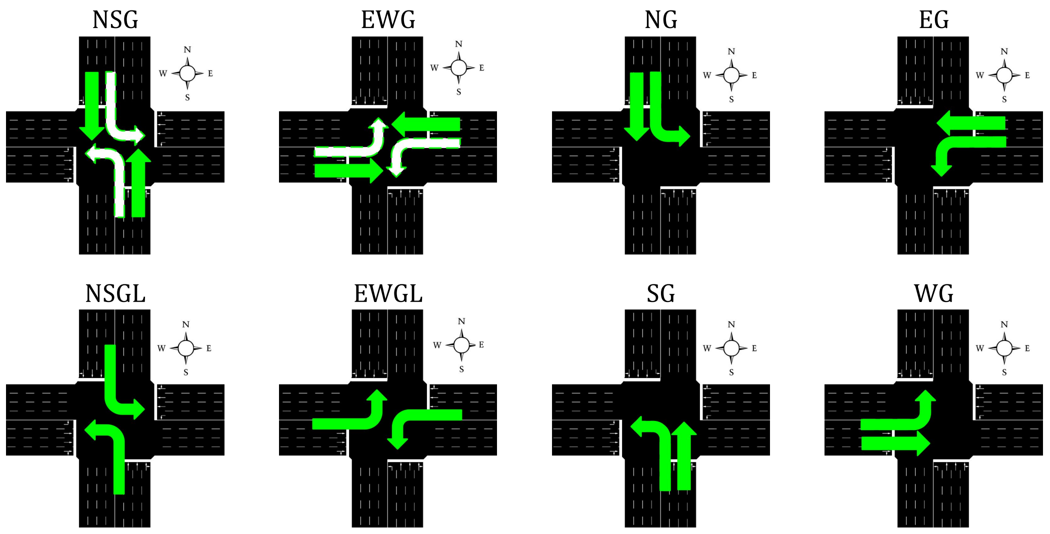


Figure 12: Actions available to RL TSC. The four action set is A_4 = {NSG, EWG, NSGL, EWGL} and the eight action set is A_8 = {NSG, EWG, NSGL, EWGL, NG, SG, EG, WG}. For each action, solid green arrows indicate protected movements and white arrows indicate permissive movements.

indicates the incoming direction of the lane and the integer indicates its position starting from the rightmost lane. For example, $l_{E,4}$ is the incoming East, leftmost turning lane. The set of all incoming intersection lanes is $L = \{l_{c,i} | c \in C, i \in [0, 1, ..., 4]\}$.

The vehicle demand is modelled as a Poisson point process. An exponential distribution models the time between vehicle generation events and is controlled by a dynamic rate parameters $\lambda(t)$. The function defining the rate parameter is different depending on whether training or testing the RL TSC agents. Further details are provided in the Appendix and Figure 14.

3.7.2 Traffic Signal Control

The TSC dictates how vehicles traverse the intersection. A phase is a tuple of traffic signals which indicate movement priority, one for each movement at the intersection. The intersection is controlled by selecting sequences of phases that ensure conflicting movements are not given priority simultaneously. This is accomplished by inserting yellow change and red clearance phases between conflicting phase sequences. In this research, phases are denoted by a pair of compass directions and the movement priority signal. For example, the phase that gives protected movements to NS and SN through movements is North-South Green (NSG). The NSG phase also makes left turning movements permissive (i.e., routes NE and SW). The compass directions not represented in a phase have

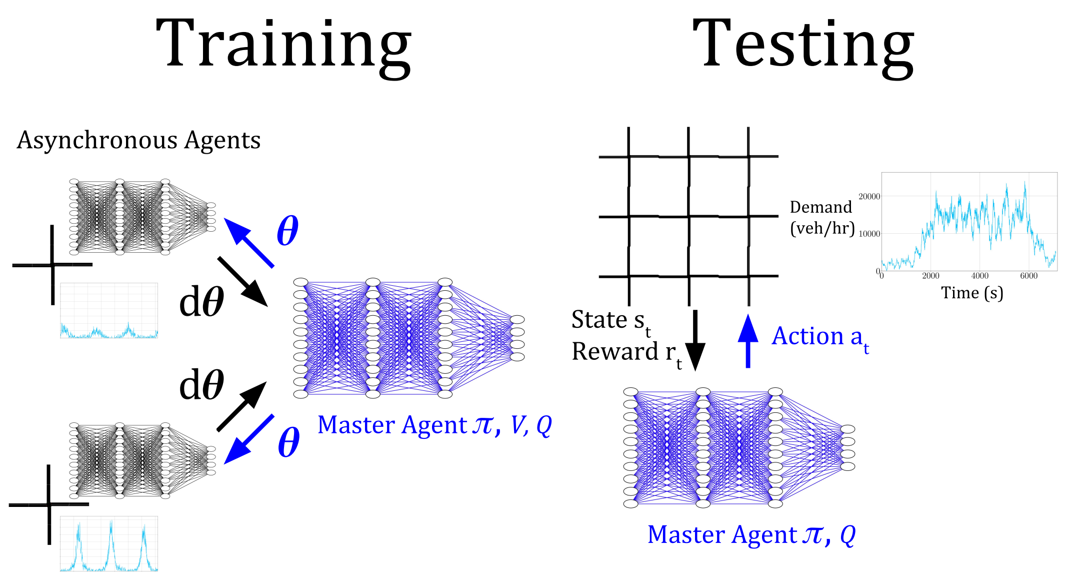


Figure 13: Model training and test architecture for 2 asynchronous agents.

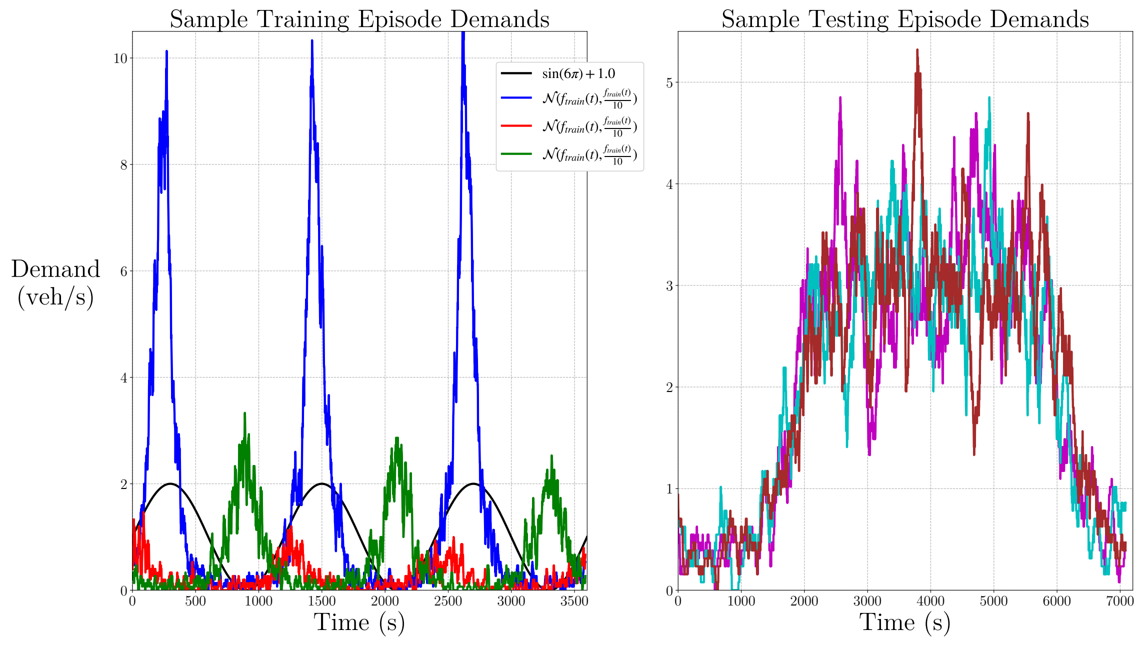


Figure 14: To generate vehicles into the network an exponential distribution is used to create random samples representing times between vehicle generation events. Different functions are used for modelling the rate parameter $\lambda(t)$ of the exponential distribution during testing and training.

implied red signals (i.e., NSG means all East and West movements are red). However, right turning movements are always permissive, regardless of the phase. Phases with an L indicate protected left turning movements and prohibited through movements. The red clearance phase (R) prohibits all through and left turning movements.

3.7.3 State

The TSC agent interacts with the environment in discrete timesteps - one timestep represents one second in simulation. At timestep t, the agent first observes the state of the environment s_t. The state observation in this research is an aggregate statistic of each lane - normalized density and normalized queue. First, an observable distance d is chosen which extends away from the intersection stopline. This distance bounds the agent's state observation and in practice would be the maximum range of the intersection sensors. The jam density c of each lane is then determined given two parameters, the average vehicle length l_v and average stopped vehicle headway h. The jam density of any lane is then $c = \frac{d}{l_v + h}$. Assume V_l represents the set of vehicles on lane l and $V_{l,q}$ represents the set of queued vehicles on lane l, then each lane's state is represented by two real-valued scalars, $\frac{|V_l|}{c}$ and $\frac{|V_{l,q}|}{c}$. The state observation also includes the current traffic phase encoded as a one hot vector. If the intersection has m incoming lanes, then the agent observes a state $\mathbf{s_t} \in \mathbb{R}^{(2m+|A|+1)}$.

The authors' acknowledge that the chosen state representation would require specialized sensors to observe in practice, however, a variety of existing technologies are capable (e.g., video cameras, radar, wirelessly connected vehicles).

3.7.4 Action

After observing the state of the environment s_t, the agent selects an action a_t. The actions available to the agent are the green phases. The authors model two action sets, illustrated in Figure 12, to determine if greater action diversity affects RL TSC performance. The first set A_4 consists of through movements and protected left turning movements and the second, larger action set A_8 contains the first set with additional protected movements for each compass direction. All necessary yellow and red phases are inserted between sequences of conflicting actions/green phases. The action set is constrained exclusively to green phases as these are the only phases the agent has to learn meaningful policies about. Whenever the agent chooses a green phase as the next action, it is enacted for 15 s. After 15 s has elapsed, the agent observes a new state and selects a new action.

A 15 s green phase duration is selected for two reasons. First, it is sufficiently long as recommended by municipalities [105] for stability to prevent quick phase oscillation. Second, it satisfies the constraint, though not modelled in this research, for pedestrian crossing time given the intersection

Table 6: Traffic Signal Phase Information

			Turning Movements		
Action	NEMA Phases	Compass Directions	Left	Through	Right
NSG	2, 6	North, South	Permissive	Protected	Permissive
NSGL	1, 5	North, South	Protected	Prohibited	Permissive
EWG	4, 8	East, West	Permissive	Protected	Permissive
EWGL	3, 7	East, West	Protected	Prohibited	Permissive
NG	2, 5	North	Protected	Protected	Permissive
SG	1, 6	South	Protected	Protected	Permissive
EG	4, 7	East	Protected	Protected	Permissive
WG	3, 8	West	Protected	Protected	Permissive

geometry modelled. Any TSC method, RL or otherwise, must have phase durations sufficient in duration to allow pedestrians to safely traverse the intersection. Assuming an average pedestrian walking speed of 1.5 m/s, and considering the modelled intersection has five parallel lanes in an direction, each 3.2 m wide, pedestrians require a minimum of 11 s to safely cross to the road median. For an increased safety factor, this research selected a green phase action duration of 15 s, above the minimum required 11 s.

There are no additional constraints on the RL TSC's action selection. The RL TSCs do not follow a cycle with structured phase sequences or have maximum green times. The RL TSCs are given as much action autonomy as possible while still prioritizing safety. The RL TSC policy can be considered *ad-hoc*; it observes the current state s_t and then selects an action (i.e., phase $a_t \in A$).

Any policy developed by the agent can be represented using a ring-barrier structure and National Electrical Manufacturers Association (NEMA) phasing. Additional traffic phase information is shown in Table 6.

3.7.5 Reward

After observing state $\mathbf{s_t}$ and taking action a_t, the agent receives feedback in the form of a scalar reward $r_t \in \mathbb{R}$. Many reward functions exist for RL TSC (e.g., functions of queue, flow, delay). In this research, the reward is the negative square of queued vehicles, defined in Equation 18:

$$r_t = -(|V_{t,Q}|^2) \tag{18}$$

Where $V_{t,Q}$ represents the number of queued vehicles on incoming lanes at time t. The maximum achievable reward in single step is 0, because $r_t \leq 0$ (i.e., punishments unless no vehicles queued). The square number of queued vehicles is used to increasingly punish long queues.

3.7.6 Agent

In RL, the agent is the entity which implements and improves the policy through interaction with the environment. In this research, the agent is an artificial neural network. The artificial neural network with parameters θ represents the action-value function $Q(s, a; \theta)$ for AQ and both the policy $\pi(a|s; \theta)$ and state value function $V(s; \theta)$ for A3C. An artificial neural network is chosen because of its flexible function approximation capabilities. According to the Universal Approximation Theorem, feed forward neural networks can be trained to represent a wide variety of functions [106]. Because the state is real-valued, $\mathbf{s_t} \in \mathbb{R}^{\mathbb{N}}$, tabular methods are infeasible. Function approximators, such as artificial neural networks, are useful when tabular methods fail.

The architecture of the artificial neural network is first the state $\mathbf{s_t}$ is input to a fully connected hidden layer of $|\mathbf{s_t}|$ neurons with rectified linear unit (ReLu) activation functions, followed by another fully connected hidden layer of $|\mathbf{s_t}|$ neurons with ReLu activation functions. Both AQ and A3C have the same aforementioned neural network architecture, but differing final layers. The AQ's final output layer is $|A|$ neurons with linear activation functions. The A3C's final output layer is modified depending on whether using the state value function or the policy. The state value function final layer is a single neuron with a linear activation function. The policy function final layer is $|A|$ neurons with softmax activation functions. Constructed in this way, where the final layer is modified depending on whether the state value or policy function is desired, the A3C's parameters are shared, allowing for efficient training.

Each RL TSC method is combined with an action set to yield four RL TSC agents, AQ-4, AQ-8, A3C-4 and A3C-8. The AQ-4 and A3C-4 agents select actions from A_4 and AQ-8 and A3C-8 agents select actions from A_8. Additional information for each RL TSC method is detailed in the Appendix.

3.8 Experiments

The traffic microsimulator SUMO v0.32 [4] was used for training and testing experiments. Libraries used for implementation, including the asynchronous RL algorithms, were Tensorflow v1.2.0 [94], NumPy, SciPy [96] and additional open-source code [95]. The Adam gradient descent algorithm [107] with a learning rate $\alpha = 0.0001$ was used to optimize the parameters of the artificial neural networks.

The system configuration for training and testing was a desktop computer running Ubuntu 16 with an i7 4.2 GHz processor and 16 GB of RAM. As opposed to other modern machine learning

Table 7: Model Hyperparameters

Variable	Hyperparameter	Value
d	Lane State Range	150 m
l_v	Average Vehicle Length	5 m
h	Average Vehicle Headway	2.5 m
c	Lane Jam Density	20 veh
$\lvert\mathbf{s_t}\rvert$	State Cardinality	58
T_{train}	Train Simulation Episode Length	3 600 s
T_{test}	Test Simulation Episode Length	7 200 s
n	Max Experience Return steps	4
γ	Discount Factor	0.99
β	Entropy Regularization Strength	0.01

research, the AQ and A3C algorithms were specifically developed for use without modern graphics processing units. The asynchronous algorithms leverage multi-core processors to run multiple agent-environments in parallel, each on their own, unique thread. This allows research to be conducted efficiently with modest computing resources without expensive graphics processing units. The authors used eight threads each running an agent-environment pair. Each thread's agent generates experiences and computes parameters updates that asynchronously update a master network. An illustration for model training and testing can be seen in Figure 13. Hyperparameters used during training and testing can be found in Table 7. These parameters were selected to reflect practical implementation and determined via empirical testing.

The RL TSC agents only chooses actions when vehicles are present at the intersection, meaning terminal states $\mathbf{s_{terminal}}$ are states where no vehicles are present on incoming lanes L (i.e., $\forall l, V_l = \{\}$). When no vehicles are present, the RL TSC enacts the red clearance phase (R).

The proposed RL TSCs are compared against two TSC methods, a uniform random policy for each action set, Random-4 and Random-8, and Websters-4.

A uniform random policy ($\forall a, \pi_{rand}(a|\mathbf{s_t}) = \frac{1}{|A|}$) is naive with respect to optimality but included for analysis. It is the initial policy of all RL TSCs and therefore a baseline for measuring overall learning progress. The RL TSC's initial action selection is uniform random, as they have no experience to favour any action over another. Ideally, with repeated interaction and experience with the environment, the RL TSC's initially random policy will converge towards the optimal policy π^*.

The Websters-4 TSC develops the green phase durations in a cycle based on the estimated flow capacity ratio. The flow is estimated over some historical interval (one hour in this research), pro-

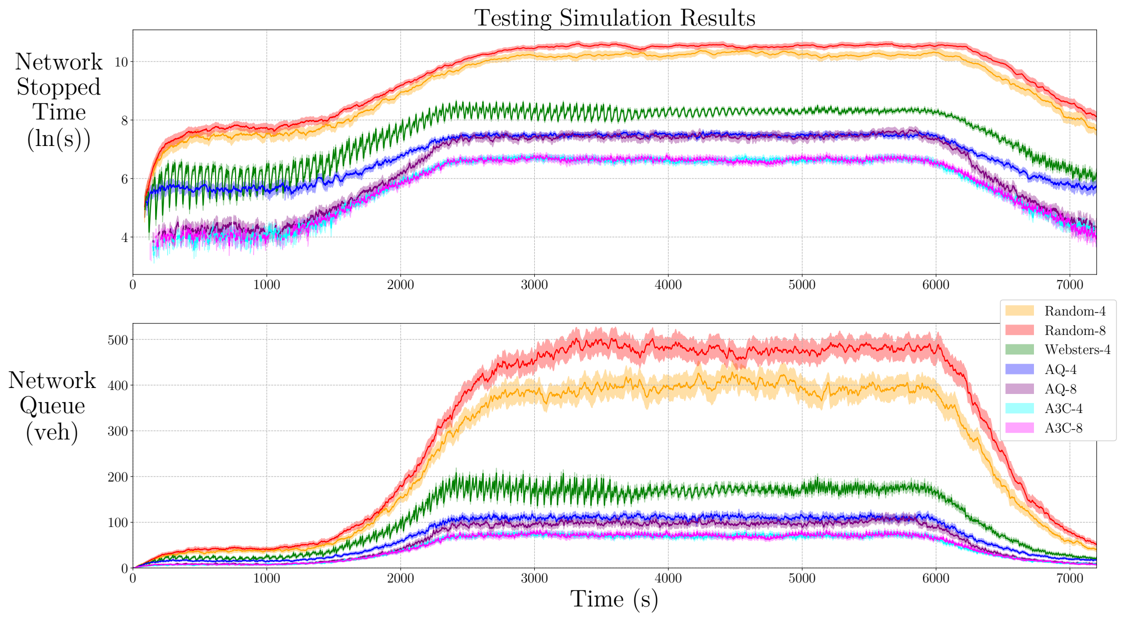


Figure 15: Network stopped time and queue samples over 50 independent testing simulation runs for each TSC method. Note the natural logarithm network stopped time is presented for improved reader clarity because of the variance across TSC methods.

portionally assigning green phases based on a green phases' flow capacity ratio (i.e., green phases with higher flow capacity ratios receive longer green times). Webster's method requires some hyperparameters for stability, a minimum phase duration (15 s), maximum cycle length (110 s) which were selected based on recommendations from the City of Toronto [105].

Each TSC method was executed for 50 independent testing demand simulation episodes, illustrated in Figure 14, with random seeds to collect statistics for comparison. The testing episodes are $T_{test} = 7\,200$ s in duration on a three by three square grid of intersections. Vehicle stopped time, queue, travel time and flow are the measures of effectiveness collected during testing to compare TSC methods. Network data is also collected, representing the cumulative sum of a MoE over the entire testing episode. Additional information on training and testing can be found in the Appendix.

3.9 Results and Discussion

Results comparing all TSC methods are represented in Table 8 and visually in Figure 15 and Figure 16. As expected, the random policy performs poorly, exhibiting the highest vehicle stopped time,

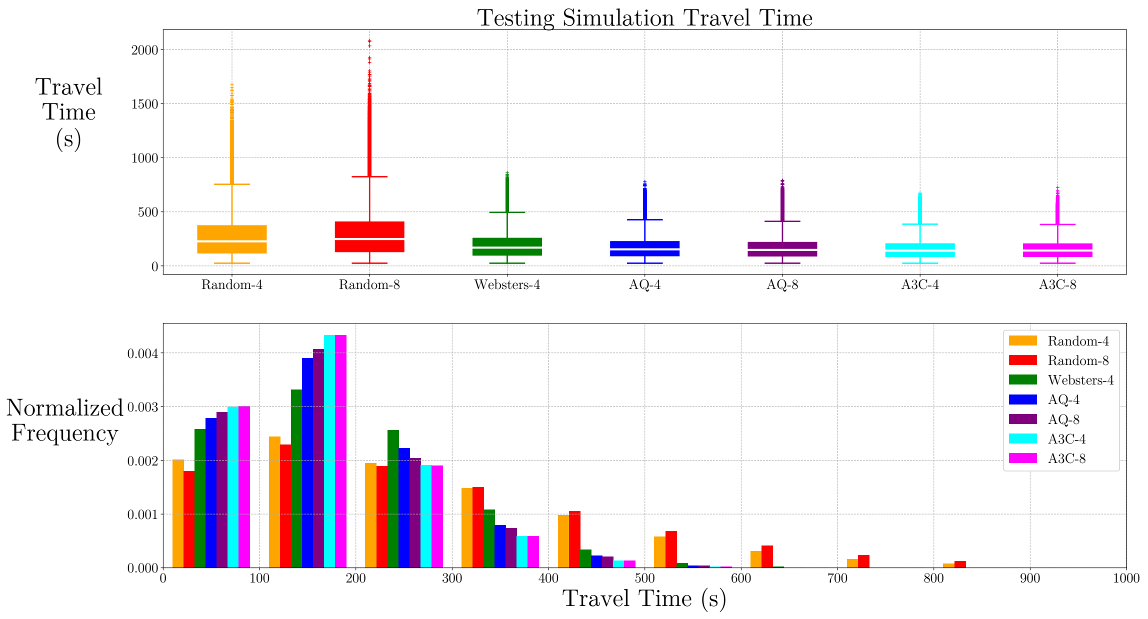


Figure 16: Vehicle travel time samples over 50 independent testing simulation runs for each TSC method.

Table 8: Traffic Signal Control Testing Results

TSC	Network Stopped Time ($s\times10^6$/sim)	Network Queue ($veh\times10^5$/sim)	Network Flow ($veh\times10^3$/sim)	Travel Time (s)	Training Time (hr)
Random-4	134±5.2	18±0.4	14±0.1	267±0.4	0
Random-8	173±5.3	22±0.4	14±0.1	293±0.5	0
Websters-4	19±0.3	8±0.1	14±0.1	187±0.3	0
AQ-4	9±0.1	5±0.1	14±0.1	170±0.2	6
AQ-8	8±0.2	4±0.1	14±0.1	165±0.2	13
A3C-4	**4±0.1**	**3±0.1**	14±0.1	**156±0.2**	1
A3C-8	**4±0.1**	**3±0.1**	14±0.1	**156±0.2**	1

Note: 95% C.I. from 50 independent testing episodes for each TSC method. Network MoE implies cumulative over the entire testing episode.

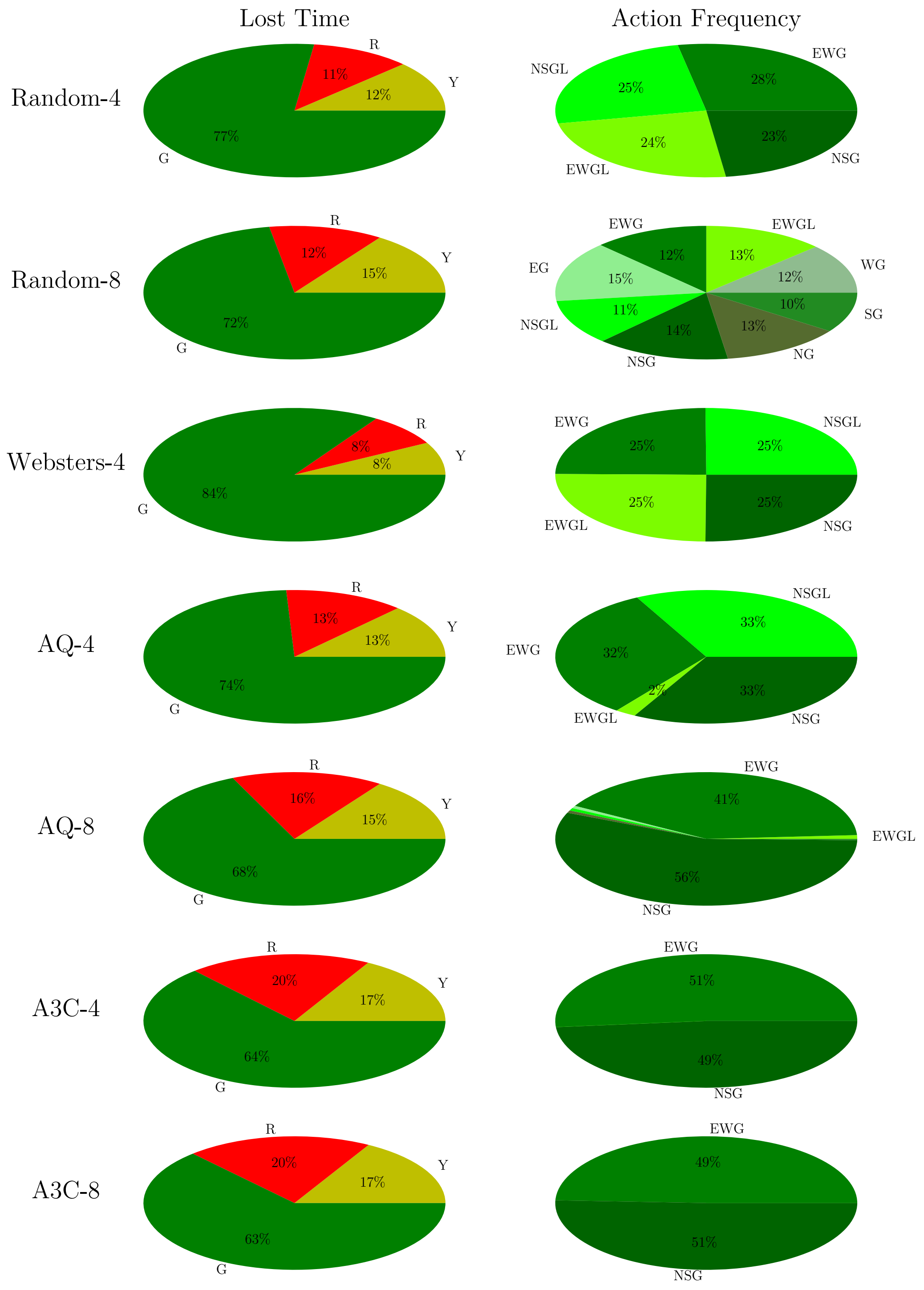


Figure 17: Traffic Phase Lost Time and Action Frequency samples collected over 50 independent testing simulations for each TSC method. Phases selected with a frequency $< 2\%$ are unlabeled.

queues and travel time. Comparing the Websters-4 and RL TSCs, all RL TSCs achieve superior performance, reducing mean vehicle stopped time, queues and travel time. Comparing the AQ and A3C TSCs, the A3C TSC achieves the lowest mean vehicle stopped time, queues and travel time. Interestingly the different action space variants of A3C, A3C-4 and A3C-8, offer no difference in performance. The authors' initial hypothesis was that a larger action space (i.e., $|A_8| > |A_4|$) would allow the TSC agent greater flexibility in developing the optimal policy.

The difference in training times between RL TSC is stark, with both A3C methods requiring only one hour, short by RL standards, while the AQ methods require significantly more time to achieve inferior performance. The training time and performance disparity provide evidence that RL methods which explicitly represent the policy, such as actor-critic methods, instead of indirectly deriving it through an action-value function are more suitable for developing adaptive TSCs.

Policy analysis is conducted using phase/action data for each TSC method, presented in Figure 17. Lost time is defined as phases where vehicles are not traversing the intersection (i.e., yellow and red phases). Lost time occurs when the TSC switches between green phases. Websters-4 spends the most amount of time in green phases and the least in yellow and red phases. The RL TSCs accrue more lost time, indicating more frequent phase switching. Since the RL TSC are acyclic and select the next phase in an *ad-hoc* fashion, this behaviour is likely producing more lost time.

Considering the action frequency for each TSC method, a few observations emerge. First, the Websters-4 does not favour any action. Second, the RL TSCs learn a policy that predominantly selects protected through movements (i.e., NSG and EWG) over others except for AQ-4, which selects almost exclusively from one of NSG, EWG or NSGL. The authors find it curious that the AQ-4's policy isn't more balanced, selecting NSGL frequently but EWGL infrequently. This may be an indication of insufficient training, but without additional experiments this is only conjecture. Third, the AQ-8, A3C-4 and A3C-8 policies have similar action frequencies, with both A3C TSC almost identical. It is surprising to the authors that the learned optimal policy appears to require only two actions, NSG and EWG. The performance similarity between TSCs with different action spaces is now easily understood; despite the potential for different policies to be learned because of different action spaces, different RL TSC all discovered that only a subset, {NSG, EWG}, of the actions is required to achieve optimality in the experiments conducted. Although it may strike the reader as strange that the optimal policy developed by the RL TSCs essentially eliminated all actions/phases except those with protected through movements, this finding was also observed in research using a micro-auction based TSC [108]. Considering that a completely different optimization technique observed the same result can be seen as mutually corroborating of the observed behaviour in this research.

3.10 Contribution

Although significant RL TSC research has been conducted, the following deficiencies are noted and the authors' contributions detailed.

First, this research proposes two RL TSC, AQ and A3C, to determine if there is any difference between RL paradigms (i.e., value or policy-based methods). Additionally, two different action spaces are modelled to determine a larger action space's impact on RL TSC performance. This research also seeks to demonstrate that the proposed asynchronous methods are practical and efficient, requiring modest computational resources and time, highlighting their feasibility for real-world deployment.

Second, some previous work has modelled the TSC too simply by omitting yellow change, red clearance (or both) phases or ignoring pedestrian constraints. This was likely done to reduce uncertainty for the agent, however, it removes an important element of the problem that would be present in practice. If RL methods are to provide solutions to TSC, they must authentically represent the problem in every regard. This research implements an authentically challenging model without simplification, including yellow change and red clearance phases four seconds in duration each [105]. Additionally, previous research has used simple vehicle generation models to generate vehicles into the network. The authors' present a model closer to reality with a stochastic, dynamic traffic demand to test the proposed methods.

Third, although much of the previously conducted RL TSC research has demonstrated it capable of superior performance compared to traditional methods, little attention has been given to analyzing what are the RL TSC methods doing differently. This research analyzes the TSC policies developed by all methods to gain insight into the performance differences. The analysis considers traditional traffic measures of effectiveness (MoE) such as stopped time, queue, travel time [105] along with TSC phases metrics such as lost time and phase frequency. The lost time and phase frequency provide an explicit description of the a TSC's behaviour for analysis. This research is valuable beyond academia; for example, to practitioners, such as transportation engineers, it elucidates and clarifies the capabilities of RL TSC, going beyond previous research which demonstrated its superior performance without inquiring to its specific behaviour.

3.11 Conclusion

Two asynchronous RL TSC were proposed and compared in simulation against random and Websters TSCs. Results show that the RL TSCs outperform all other methods by developing a surprisingly simple policy in a multi-intersection traffic network with dynamic, stochastic vehicle demand. Results also show that RL methods that explicitly represent the policy, such as A3C, achieve superior performance in TSC compared to value-based methods in a fraction of the training time.

Limitations of the conducted research and areas for future work include the following.

In its current form the architecture for action selection is constraining. The agent chooses the next green phase but its duration is constant - 15 s in this model. This limits green phases to interval multiples of 15 (e.g., 15, 30, 45). Instead of having a constant action duration, the action selection architecture could be improved by also choosing the action's duration, as this would allow the agent to better control the environment. Other researchers have termed this 'fine grain action repetition' and have shown it improve RL's performance in other domains [109]. The authors consider this a promising extension that would reap improvements.

If the action space or state space was changed, the model would need to be retrained. Although we have demonstrated that actor-critic based models train quicker than value-based models, they still require time to train. Intersections with different geometries and numbers of green phases would each require their own unique agent to be trained. This limitation could be overcome using transfer learning, which allows RL agents to generalize their behaviour across environments or to new environments [110, 111]. Transfer learning could be used to develop one agent capable of controlling intersections with differing geometries and actions spaces.

Although the testing scenario used in this research consisted of multiple intersections, each intersection was controlled by an isolated RL agent. Therefore, the authors' models can not be considered as a true multi-agent RL system. True multi-agent RL TSC systems have been successfully developed using other architectures [64, 84]. The authors' model could be combined with previous multi-agent RL TSC research for improved performance.

3.12 Appendix

To develop a reinforcement learning agent using neural network functions approximation, the agent learns by interacting with the environment. Environment interaction at time t yields an experience tuple e_t, created by the agent observing the environment state s_t, selecting an action a_t, receiving feedback in the form of a reward r_t and observing a new state s_{t+1}, forming $e_t = (\mathbf{s_t}, a_t, r_t, \mathbf{s_{t+1}})$. Supervised learning trains neural networks given a labelled training dataset containing input data and correctly labelled outputs. Input data from the dataset is fed into the network and the network's output is compared with the labelled output using a loss function to compute an error. The backpropagation algorithm [44] is used to propagate the error and train the network parameters. Instead of a labeled dataset, experiences tuples are used as training data along with a custom loss function to train a neural network as a reinforcement learning agent.

In the original deep Q-network (DQN) [45], only one experience a time was used to estimate the return. It has been demonstrated that using multiple sequential experiences can improve the return estimate, termed n-step returns. This research uses 4-step returns to estimate the return, which has been shown to be effective in accelerating learning by improving reward estimation [46].

This research consists of two phases, training and testing. First, the neural network must be trained using experiences generated by the agent interacting with the environment. Multiple agent-environment pairs execute asynchronously in parallel to generate experiences and parameter updates, using a local parameter set θ', for a master network. The benefits of using multiple environment-agent pairs include training diversity and computational efficiency. Each environment can be different, creating diversity in the agent's experiences, which improves learning. Since multiple environment-agent pairs are executing in parallel, training time is decreased, as the network is being updated from more experiences than would be available in serial.

3.12.1 Training

This research uses training episodes where an agent interacts with a single, isolated intersection with a stochastic demand for $T_{train} = 3\,600$ s. Each episode exhibits a stochastic demand, generated before the start of the episode, illustrated in Figure 14. The vehicle demand generation training function $f_{train}(t) = A\sin(6\pi + \phi) + 1.0$ is created using a periodic function $\sin(6\pi) + 1.0$, randomly scaled by $A = \text{Unif}(0.25, 4)$ and phase shifted by $\phi = \text{Unif}(0, 6\pi)$ as a form of data augmentation for each training episode. Further stochasticity is added by sampling from $\mathcal{N}(f_{train}(t), \frac{f_{train}(t)}{10})$. These efforts are undertaken to ensure training diversity over the entire demand spectrum.

This stochastic training is used to ensure the agent experiences all types of traffic demands and does not overfit to any specific traffic demand. Described in this way, these efforts can be considered as a form of training data augmentation often seen in supervised learning. Training is complete when the neural network parameters have converged and are no longer changing with additional experiences.

3.12.2 Testing

After training is complete, the agent's policy is evaluated. A criticism of many modern reinforcement learning research studies is that the environment used to evaluate the learned behaviour of the agent is exactly the same as used during training. This is problematic because it does not evaluate the agent's capacity to generalize in new environments. This problem is avoided in traditional supervised learning by dichotomizing the dataset into a training set and a testing set. This ensures an honest evaluation of the algorithm's generalization performance on new, unseen data. Influenced by supervised learning, this research attempts to overcome the aforementioned problem by differentiating the test and training data in two ways. First, a different traffic network is used during testing. Instead of a single, isolated intersection, as is used during training, a three by three square grid of nine intersections is used during testing. Each intersection is controlled by a copy of the agent developed during training. This testing network is used to better emulate real-world traffic networks, as square grids are a popular topology in urban centres. Second, the testing traffic demand is generated from a different piecewise, stochastic function than used in training. The testing demand simulates a rush hour scenario of length $T_{test} = 7\,200$ s, illustrated in Figure 14. The testing regimen is designed in this way to highlight the dynamic, stochastic nature of traffic experienced in practice, and how pre-timed, cycle based methods are insufficiently adaptive to the changes in traffic.

3.12.3 Asynchronous Q-learning

The AQ algorithm implements reward normalization by dividing the current reward by the absolute value of the largest reward experienced r_{max}. Given the proposed reward definition, negative square of queued vehicles, this normalization effectively clips the reward to the interval $[-1, 0]$. During training, the ϵ-greedy action selection policy is used, linearly annealed over training until it reaches $\epsilon = 0.05$. In addition to global parameters θ and local parameters θ', the asynchronous Q-learning algorithm maintains a set of target network parameters θ^{-}, as in the original DQN to stabilize learning. The target network parameters are updated to the master network parameters every $I = 10\,000$ experiences for AQ-4 and $I = 20\,000$ experiences for AQ-8. A longer target network update interval is used for AQ-8 because of the larger action space, requiring more experiences for a stable update. The AQ-4 was trained for $M = 500\,000$ experiences and the AQ-8 was trained for $M = 1\,000\,000$ experiences. After sufficient training, the optimal policy is derived by selecting the action with the highest value, $\pi^{*}(\mathbf{s}, a) = \text{argmax}_a Q(\mathbf{s}, a; \theta)$. The AQ training pseudocode can be found in Algorithm 1.

3.12.4 Asynchronous Advantage Actor-Critic

The A3C algorithm utilizes reward normalization and the gradient of the policy entropy to improve learning. After n experiences have been appended to the experience buffer E, the rewards are standardized by subtracting the mean reward μ_r and dividing by the standard deviation of the reward σ_r, calculated from all rewards in the experience buffer. This transformation centres the rewards at 0, controlling their variance and preventing the gradients from becoming too large during backpropagation [112]. Both A3C variants were trained for $M = 100\,000$ experiences. The A3C training pseudocode can be found in Algorithm 2.

Algorithm 1: AQ training pseudocode [46]

```
$m \leftarrow 0$, $r_{max} \leftarrow 1.0$, Initialize $\theta, \theta^{'}$
**While** $m < M$:
 $t \leftarrow 0$, Generate stochastic demand for training episode
 **While** $t < T_{train}$:
  Reset parameter gradients $\mathrm{d}\theta \leftarrow 0$, Set thread parameters to global $\theta' = \theta$
  $E = (\ )$
  Observe state $\mathbf{s_t}$
  **While** $\mathbf{s_t} \neq \mathbf{s_{terminal}}$ **and** $|E| \neq n$:
   Perform $\epsilon$-greedy action $a_t$ from $\mathrm{argmax}_a Q(\mathbf{s_t}, a; \theta')$
   Receive reward $r_t$ and observe new state $s_{t+1}$
   **If** $|r_t| > r_{max}$:
    $r_{max} = |r_t|$
   $r_t \leftarrow \frac{r_t}{r_{max}}$
   Append $e_t = (s_t, a_t, r_t, s_{t+1})$ to buffer E
   $\mathbf{s_t} = \mathbf{s_{t+1}}$, $t \leftarrow t+1$, $m \leftarrow m+1$
  $G = 0$ **If** $\mathbf{s_t} == \mathbf{s_{terminal}}$ **else** $G = \max_a Q(\mathbf{s_t}, a; \theta^-)$
  **For** $e$ **in** reverse($E$):
   $\mathbf{s} = e_{s_t}, a = e_{a_t}, r = e_{r_t}, \mathbf{s_{t+1}} = e_{s_{t+1}}$
   $G \leftarrow r + \gamma G$
   Collect gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \frac{\partial (G - Q(\mathbf{s}, a; \theta'))^2}{\partial \theta'}$
  Asynchronously update $\theta$ with $\mathrm{d}\theta$
  **If** $m \bmod I == 0$:
   $\theta^- \leftarrow \theta$
```

Algorithm 2: A3C training pseudocode [46]

```
$m \leftarrow 0$, Initialize $\theta, \theta^{'}$
While $m < M$:
  $t \leftarrow 0$, Generate stochastic demand for training episode
  While $t < T_{train}$:
    Reset parameter gradients $\mathrm{d}\theta \leftarrow 0$, Set thread parameters to global $\theta' = \theta$
    $E = (\ )$, Observe state $\mathbf{s_t}$
    While $\mathbf{s_t} \neq \mathbf{s_{terminal}}$ and $|E| \neq n$:
      Perform action $a_t$ from $\pi(a_t|\mathbf{s_t};\theta')$
      Receive reward $r_t$ and observe new state $s_{t+1}$
      Store $e_t = (s_t, a_t, r_t, s_{t+1})$ in rollout E
      $\mathbf{s_t} = \mathbf{s_{t+1}}$, $t \leftarrow t + 1$, $m \leftarrow m + 1$
    $G = 0$ If $\mathbf{s_t} == \mathbf{s_{terminal}}$ else $G = V(\mathbf{s_t};\theta')$
    For $e$ in reverse($E$):
      $\mathbf{s} = e_{s_t}, a = e_{a_t}, r = e_{r_t}, \mathbf{s_{t+1}} = e_{s_{t+1}}$
      $r \leftarrow \frac{r-\mu_r}{\sigma_r}$
      $G \leftarrow r + \gamma G$
      $Adv = G - V(\mathbf{s};\theta')$
      Collect policy gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \nabla_{\theta'} \log_e \pi(a|\mathbf{s};\theta')(Adv) + \beta \nabla_{\theta'} H(\pi(\mathbf{s};\theta'))$
      Collect value gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \frac{\partial (Adv)^2}{\partial \theta'}$
    Asynchronously update $\theta$ with $\mathrm{d}\theta$
```

4 City Scale Multi-agent Adaptive Traffic Signal Control

4.1 Introduction

Knowledge generated from the previous chapters is used to design a flexible and scalable architecture for reinforcement learning adaptive traffic signal control model development. Analysis of the learned policies in the previous chapters showed an acyclic controller can learn to ignore some phases which can decrease a minority of road users quality of service. Retaining the improvements demonstrated by directly developing a policy using an actor-critic (i.e., deep deterministic policy gradients) algorithm, a cycle based, dynamic phase duration adaptive traffic signal control architecture is designed. To validate the proposed controller, a simulation of the City of Luxembourg, Luxembourg is used. A distributed architecture is developed to cope with the scale of the validation simulation, as 196 adaptive traffic signal controllers must be developed. The proposed method is parsimonious, as independent reinforcement learning is used to train in a multi-agent setting (i.e., no explicit coordination/communication necessary between intersections).

W. Genders and S. Razavi, "Distributed deep deterministic policy gradients for adaptive traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2018. *Submitted 15 August 2018*, T-ITS-18-08-0805

The co-author's contributions to the above work include:

- Financial and technical supervision of the study presented in this work.
- Manuscript editing.

4.2 Distributed Deep Deterministic Policy Gradients for Adaptive Traffic Signal Control

4.3 Abstract

Congestion in transportation systems negatively impacts mobility, the environment and human health. Intersection traffic signal controllers are an important element of many cities' transportation infrastructure where sub-optimal solutions can contribute to congestion. However, developing optimal transportation control systems at the appropriate scale can be difficult, as cities' transportation systems are large, complex and dynamic. This research develops a scalable architecture for flexible and efficient reinforcement learning, adaptive traffic signal controller model development using recent innovations from machine learning. To validate the proposed methods, a simulation of the City of Luxembourg is used, consisting of 196 intersections. Evaluated over 40 simulated days and compared to random and fixed time traffic signal controllers, the proposed adaptive system reduces delay, queue, travel time and stopped time. Although the cumulative effect of the reinforcement learning adaptive control is positive, 15% of intersections experienced increased delay and queues, leaving room for future improvements.

4.4 Introduction

Cities rely on road infrastructure for transporting individuals, goods and services. Demand beyond infrastructure capacity creates congestion, adversly affecting the environment, human mobility and health. Studies observe vehicles consume a significant amount of fuel accelerating, decelerating or idling at intersections [113]. Land transportation emissions are estimated to be responsible for one third of all mortality from fine particulate matter pollution in North America [17]. Globally, over three million deaths are attributed to air pollution per year [16]. In 2017, residents of three of the United States' biggest cities, Los Angeles, New York and San Francisco, spent between three and four days on average delayed in congestion over the year, respectively costing $19, 33$ and 10 billion USD from fuel and individual time waste [12]. It is paramount to ensure transportation systems are optimal to minimize these costs.

Automated control systems are used in many aspects of transportation systems. Intelligent transportation systems seek to develop optimal solutions in transportation using intelligence. Intersection traffic signal controllers are an important element of many cities' transportation infrastructure where sub-optimal solutions can contribute to congestion. Traditionally, traffic signal controllers have functioned using heuristic logic which can be improved. Using artificial intelligence and machine learning to develop adaptive traffic signal controllers has been significantly researched with promising results

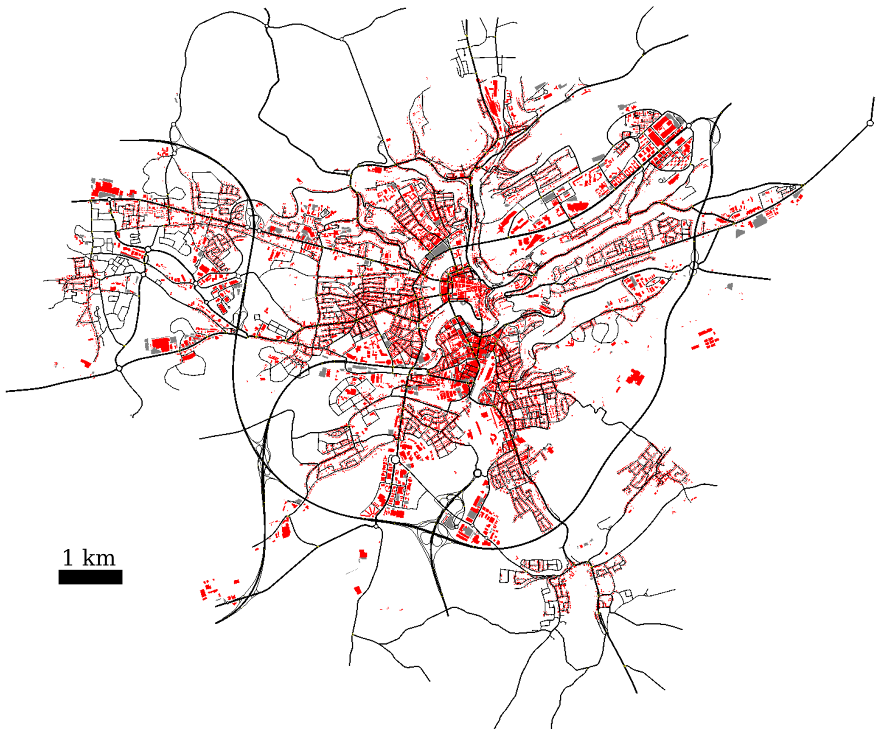


Figure 18: City of Luxembourg SUMO [4] simulation model [5, 6].

[75, 64, 100].

Traffic signal controllers govern how vehicles traverse an intersection and can be modelled as a sequential decision-making problem. Reinforcement learning can solve sequential decision-making problems by developing an optimal policy through experience [36]. The combination of reinforcement learning and deep artificial neural networks has been used to develop systems that can best humans in complicated tasks such as the game of Go [50, 51], Atari video games [45, 114, 115, 48] and image recognition [116, 117]. To reduce congestion and improve mobility at a city scale, deep reinforcement learning techniques are used in simulation to develop an adaptive traffic signal controller. To achieve this goal, considering the scope of the problem, the authors develop a scalable architecture for flexible and efficient model development using recent innovations from machine learning [118]. The proposed reinforcement learning adaptive signal controller is demonstrated to reduce delay, queue, travel time and stopped time compared to random and fixed controllers.

4.5 Related Work

Reinforcement learning has been demonstrated to be an effective method for developing adaptive traffic signal controllers in simulation [70, 57, 71, 60]. Recently, deep reinforcement learning has been used for adaptive traffic signal control with varying degrees of success [66, 68, 67, 69, 77, 119]. However, the majority of this research has used small networks (i.e., single/few intersections) and/or simplified models. Readers interested in additional research can consult extensive review articles [86, 87, 97].

Initial research using multi-agent reinforcement learning for adaptive traffic signal control yielded promising proof-of-concept results [59]. However, it too made simplifying assumptions and used simple models (e.g., vehicle dynamics, signal control dynamics) that necessitated additional work. Examples of research that define the problem at a scale that approaches the real-world are presented in Table 9. Reviewing the research conducted, a few observations are apparent.

First, at best previously simulated networks model sections of cities, and at worst are artificial grids. The number of intersections controlled are in the range of [10,100]. While these simulations are obviously more realistic than research with single, isolated intersections, it is desirable to continue increasing the scale of the simulations to reflect reality.

Second, Q-learning [38] is the dominant reinforcement learning algorithm. Q-learning is powerful, simple to implement and exhibits strong convergence with tabular and linear function approximation. However, Q-learning has some weaknesses, such as being infeasible in continuous action spaces. Also, policy-based reinforcement learning methods have demonstrated higher performance in certain domains [114, 115, 48].

Third, linear functions are the most common form of function approximation. Like Q-learning, linear function approximation is simple to implement with appealing convergence properties. However, it has been demonstrated that function approximators which can model a wider class of functions beyond linear relationships offer improved performance in many domains [45], including traffic signal control [1].

Finally, the majority of previous research used multi-agent algorithms relying on some form of communication between agents or local models of other agents. For example, the max-plus algorithm requires that agents share messages. Communication between agents is simple in simulation. However, if these systems were to be deployed in the real-world, communication infrastructure would be required. Assuming communication infrastructure exists, any system relying on communication becomes vulnerable when communication degrades/fails. A system not requiring communication would be impervious to these issues. It would avoid the problems posed by poor communication and likely be cheaper, with no infrastructure required.

Table 9: Adaptive Traffic Signal Control related work.

Research	Network	Intersections	Multi-agent	RL	Function Approximation
[120]	Grid	15	Max-plus	Model-based	N/A
[75]	Grid, Corridor	<10	None	Q-learning	Linear
[121]	Springfield, USA	20	Max-plus	Q-learning	N/A
[64]	Toronto, Canada	59	Game Theory	Q-learning	Tabular
[84]	N/A	50	Holonic	Q-learning	N/A
[65]	Grid	22	Reward Sharing	Q-learning	Bayesian
[122]	Grid	100	Regional	Q-learning	Linear
[76]	Barcelona, Spain	43	Centralized	DDPG	DNN[1]
[100]	Tehran, Iran	50	None	Actor-Critic	RBF[2], Tile Coding
[123]	Changsha, China	96	Reward sharing	Q-learning	Linear
*	**Luxembourg City**	**196**	**None**	**DDPG**	**DNN**

[1] Deep Neural Network (DNN).

[2] Radial Basis Function (RBF).

* This manuscript's research.

4.6 Contribution

Although several successful examples of adaptive traffic signal control in large-scale simulations have been detailed, the authors' work contributes in the following areas:

- **Scale**: To the best of the authors' knowledge, no previous research has demonstrated adaptive traffic signal control at the scale of the City of Luxembourg, using the Luxembourg Sumo Traffic (LuST) Scenario [5, 6]. Demonstrating the proposed model in a simulated real-world scenario is necessary if it is to ever be deployed in the field.

- **Parsimony**: Multi-agent systems often rely on communication between or models of other agents to function. Depending on the agent space topology, the curse of dimensionality can manifest. The proposed system uses independent reinforcement learning agents requiring only local information at run-time.

4.7 Background

4.7.1 Traffic Signal Control

An intersection is composed of traffic movements, or ways that a vehicle can traverse the intersection beginning from an initial lane. Traffic signal controllers use phases, combinations of coloured lights that indicate when specific movements are allowed, to control vehicles at the intersection. Typically, traffic signal controllers use a repeating sequence of phases known as a cycle. If the phase durations are static within a cycle, this is known as a fixed time traffic signal controller. Actuated traffic signal controllers use sensors and boolean logic to create dynamic phase durations. Adaptive traffic signal controllers are capable of acyclic operation and dynamic phase duration to adapt to traffic conditions at the intersection. Adaptive controllers can achieve better performance at the expense of complexity, cost and reliability.

Fundamentally, traffic signal control requires two decisions at any given time; what should the next phase be and for how long in duration? Using reinforcement learning, the authors strive to learn the answers to these questions.

4.7.2 Reinforcement Learning

Reinforcement learning uses the framework of Markov Decision Processes to solve goal-oriented, sequential decision-making problems by repeatedly acting in an environment. At discrete points in time t, a reinforcement learning agent observes the environment state s_t and then uses a policy π to determine an action a_t. After implementing its selected action, the agent receives feedback from the environment in the form of a reward r_t and observes a new environment state s_{t+1}. The reward quantifies how 'well' the agent is achieving its goal (e.g., score in a game, completed tasks). This process is repeated until a terminal state $s_{terminal}$ is reached, and then begins anew. The return $G_t = \sum_{k=0}^{k=T} \gamma^k r_{t+k}$ is the accumulation of rewards by the agent over some time horizon T, discounted by $\gamma \in [0, 1)$. The agent seeks to maximize the expected return $\mathbb{E}[G_t]$ from each state s_t. The agent develops an optimal policy π^* to maximize the return.

There are many techniques for an agent to learn the optimal policy, however most of them rely on estimating value functions. Value functions are useful to estimate future rewards. State value functions $V^\pi(s) = \mathbb{E}[G_t|s_t = s]$ represent the expected return starting from state s and following policy π. Action value functions $Q^\pi(s, a) = \mathbb{E}[G_t|s_t = s, a_t = a]$ represent the expected return starting from state s, taking action a and following policy π. In practice value functions are unknown and must be estimated using sampling and function approximation techniques. Parametric function approximation, such as neural networks, use a set of parameters θ to estimate an unknown function $f(x; \theta) \approx f(x)$. To develop accurate approximations, the function parameters must be estimated with some optimization technique.

A reinforcement learning agent interacts with its environment in trajectories $e_t, e_{t+1}, e_{t+2}, ...$ or sequences of experiences, $e = (s_t, a_t, r_t, s_{t+1})$. Trajectories begin in an initial state s_{init} and end in a terminal state $s_{terminal}$. To accurately estimate the value function, experiences are used to optimize the parameters. If neural network function approximation is used, the parameters are optimized using experiences to perform gradient-based techniques and backpropagation [43, 44].

4.7.3 Deep Deterministic Policy Gradients

Deep deterministic policy gradients (DDPG) [124] are an extension of deep Q-networks (DQN) [45] (i.e., Q-learning [38]) to continuous action spaces. Similiar to Q-learning, DDPG is a model-free, off-policy reinforcement learning algorithm. This algorithm comes from the temporal-difference [73] learning family of reinforcement learning, which over time develops improved approximations, from approximations, of value functions [36]. Value functions are useful to estimate future rewards based on states or actions. The DDPG algorithm is an example of actor-critic learning, as it develops a policy function $\pi(s)$ (actor) using an action-value function $Q(s, a)$ (critic). The actor interacts with the environment and modifies its behaviour based on feedback from the critic. In practice these functions are not known and must be estimated using samples. As the name of the algorithm implies, deep artificial neural networks are used for function approximation, as they have been demonstrated capable of approximating a wide family of functions.

Training a neural network requires a loss function, which is used to determine how to change the parameters to achieve better approximations of the training data. Reinforcement learning trains neural networks using experiences from the environment. Experiences are tuples $e_t = (s_t, a_t, r_t, s_{t+1})$ that represent an interaction between the agent and the environment. To estimate the policy and value functions (i.e., neural networks), experiences are used as sample data.

The critic's loss function is the gradient of the mean squared error of the target y_t and the prediction, defined in (19).

$$\begin{gathered} y_t = r_t + \gamma \text{max} Q(s_{t+1}, a_t|\theta') \\ \nabla_\theta L_Q(\theta) = (y_t - Q(s_t, a_t|\theta))\nabla_\theta Q(s_t, a_t|\theta) \end{gathered} \tag{19}$$

The actor's loss function is the sampled policy gradient, defined in (20).

$$\nabla_\theta L_\pi(\theta) = \nabla_\theta Q(s_t, a_t|\theta)\nabla_\theta \pi(s_t|\theta) \tag{20}$$

Like DQN, DDPG uses two sets of parameters, online θ and target θ', and experience replay [125] to reduce instability during training. DDPG performs updates on the parameters for both the actor and critic by uniformly sampling batches of experiences from the replay. The target parameters are slowly updated towards the online parameters according to $\theta' = (1 - \tau)\theta' + (\tau)\theta$ with a period of P updates.

4.8 Proposed Method

The authors propose an adaptive traffic signal control model for all intersections using independent reinforcement learning and DDPG.

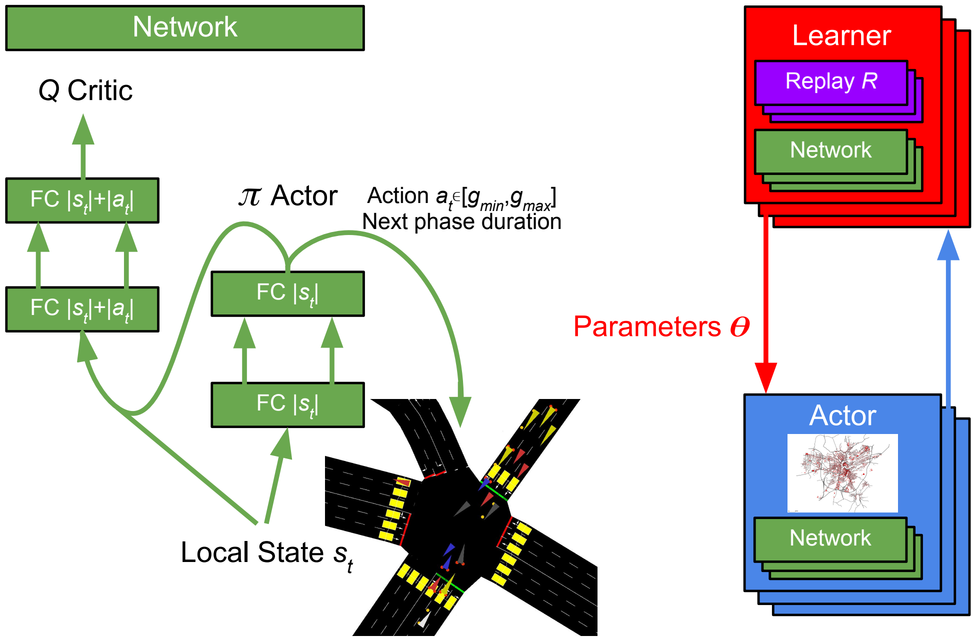


Figure 19: Adaptive traffic signal control DDPG agent (left) and distributed acting, centralized learning architecture (right). Each actor has one LuST environment and neural networks for all intersections. Each learner is assigned a subset of intersections at the beginning of training and is only responsible for computing parameter updates for their assigned intersections, effectively distributing the computation load for learning.

The proposed adaptive traffic signal controller implements a cycle with dynamic phase durations. However, the cycle skips phases when no vehicles are present on incoming lanes. The next phase logic is presented in Algorithm 1. This architecture is motivated by the observation that cycles maintain fairness and ensure a minimum quality of service between all intersection users. Once the next green phase has been determined using the cycle, the policy π is used to select its duration. Explicitly, the reinforcement learning agent is learning how long in duration to make the next green phase to maximize its return.

The set of all intersections $I = \{i_0, i_1, ...\}$ constitute the independent agents that interact in the traffic simulation. Each intersection i is controlled by a DDPG reinforcement learning agent.

Algorithm 1 Adaptive Signal Control Next Phase

```
1: procedure NextPhase($currentPhase, cycle$)
2:     $j = cycle$.index($currentPhase$)
3:     $newCycle = cycle[j + 1 :] + cycle[: j + 1]$
4:     for $phase$ in $newCycle$ do
5:         if VehiclesInPhase($phase$) then
6:             return $phase$                    ▷ Find next phase in cycle with vehicles
7:         end if
8:     end for
9:     return $all\_red$                         ▷ If no vehicles present, next phase is all-red
10: end procedure
```

4.8.1 State

4.8.2 Actor

The proposed state observation for the actor is a combination of the current phase and the queue and density of incoming lanes at the intersection at time t. Assume each intersection has a set L of incoming lanes and a set P of green phases. The state space is then defined as $S \in (\mathbb{R}^{2|L|} \times \mathbb{B}^{|P|+1})$. The queue and density of each lane is normalized to the range $[0, 1]$ by dividing by the lane's jam density k_j. The current phase is encoded as a one-hot vector $\mathbb{B}^{|P|+1}$, where the plus one encodes the all-red clearance phase.

4.8.3 Critic

The proposed state observation for the critic combines the state s_t and the actor's action a_t, depicted in Figure 19.

4.8.4 Action

The proposed action space for the adaptive traffic signal controller is the duration of the next green phase in seconds. The action controls the duration of the next phase from Algorithm 1; there is no agency over what the next phase is, only on how long it will last. The DDPG algorithm produces a continuous output, a real number over some range $a_t \in \mathbb{R}$. Since the DDPG algorithm outputs a real number and the phase duration is defined in intervals of seconds, the output is rounded to the nearest integer. In practice, phase durations are bounded by a minimum time g_{min} and a maximum time g_{max} to ensure a minimum quality of service for all users. Therefore the agent selects an action

$\{a_t \in \mathbb{Z} | g_{min} \leq a_t \leq g_{max}\}$ as the next phase duration.

4.8.5 Reward

The reward used to train the adaptive traffic signal controller is a function of vehicle delay. Delay d is the difference in time between a vehicle's free-flow travel time and actual travel time. Specifically, the reward is the negative sum of all vehicles' delay at the intersection, defined in (21):

$$r_t = -\sum_{v \in V} d_v^t \tag{21}$$

Where V is the set of all vehicles on incoming lanes at the intersection, and d_v^t is the delay of vehicle v at time t. Defined in this way, the reward is a punishment, with the agent's goal to minimize the amount of punishment it receives. Each intersection saves the reward with the largest magnitude experienced to perform minimum reward normalization $\frac{r_t}{|r_{min}|}$ to scale the reward to the range $[-1, 0]$ for stability.

4.8.6 Agent

The agent approximates the policy π and action-value Q function with deep artificial neural networks. The policy function is two hidden layers of $2|s_t|$ fully connected neurons, each with rectified linear unit (ReLU) activation functions, and the output layer is one neuron with a hyperbolic tangent activation function. The action-value function Q is two hidden layers of $2(|s_t| + |a_t|)$ fully connected neurons with ReLU activation functions and the output layer is one neuron with a linear activation function. The policy's input is the intersection's local traffic state s_t and the action-value function's input is the local state concatenated with the local action $s_t + a_t$.

By deep reinforcement learning standards these networks are not that deep, however, their architecture is selected for simplicity and to respect resource constraints, as one agent is needed for each of the 196 intersections. Canonical reinforcement learning problems (e.g., Atari, cartpole, movement simulators) require training a single agent. Since the proposed simulation requires hundreds of agents, a suitable learning architecture is required to make the problem tractable.

4.9 Experiments

4.9.1 Learning Architecture

To train agents for all intersections, a distributed acting, centralized learning architecture is developed [46, 118, 48]. Using parallel computing, multiple actors and learners are created, illustrated in Figure 19. Actors have their own instance of the environment (i.e., LuST scenario) and neural networks for all intersections. Learners are assigned a subset of all intersections, for each they have a neural network and an experience replay buffer R. Actors generate experiences e_t for all intersections, sending them in batches B to the appropriate learner. Learners only receive experiences for their assigned subset of intersections. The learner stores the experiences in an experience replay buffer, which is uniformly sampled for batches to optimize the neural network parameters. After computing parameter updates, learners send new parameters to all actors.

There are many benefits to this architecture, foremost is it makes the problem feasible. Because there are hundreds of agents, distributing computation across many actors and learners is necessary to decrease training time. Another benefit is experience diversity, granted by multiple environments and varied exploration rates.

4.9.2 Traffic Simulation

The Luxembourg Sumo Traffic scenario is used as the environment to train all the agents [5, 6]. The LuST scenario models the City of Luxembourg, displayed in Figure 18, and was developed to be a standardized testbed for evaluating traffic applications in microsimulation. The model topology is an area of 156 km^2 with 930 km of road, 89 km of which is highway. The scenario authors modelled and calibrated the 24 hour scenario to ensure a high degree of realism. Demographic and floating point car data was used to generate the traffic demand, yielding over 250 000 vehicle trips, including transit. Readers interested in additional details can consult the scenario authors' publications [5, 6].

All 196 intersections are controlled with the proposed reinforcement learning adaptive traffic signal control system over the entire 24 hour simulation. The set of intersections I is a heterogeneous set of agents, evidenced by state and action space distributions illustrated in Figure 20. Agents have between 2 and 7 phases and between 3 and 16 incoming lanes. To the best of the author's knowledge, this is the largest application of adaptive traffic signal control in microsimulation.

4.9.3 Training

To train all the agents using the proposed architecture the following training augmentations are implemented to improve learning.

First, actors are distributed temporally in the simulation so as to begin learning at different times. This is similar to the no-op starts in the Atari environment to ensure agents do not become biased to any initial state.

Second, Gaussian noise is added to actions for exploration. Various exploration rates ϵ are used as the standard deviation parameter $a_t + \mathcal{N}(0, \epsilon)$.

Third, actors implement different exploration rates, ranging from highly exploitative to highly

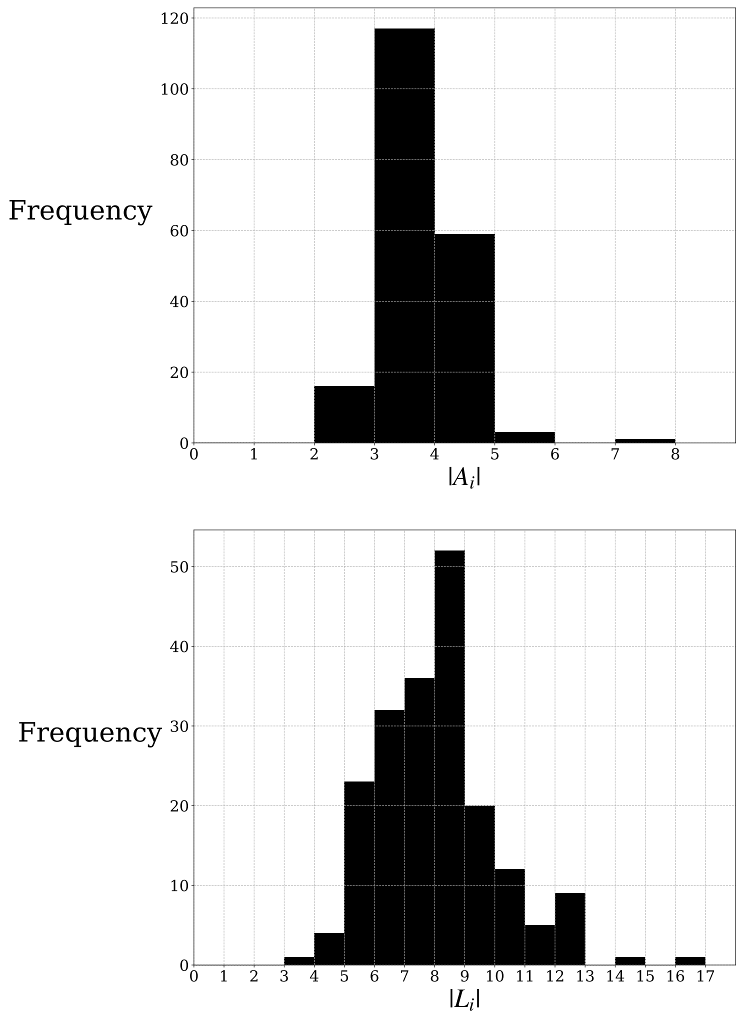


Figure 20: Agent diversity, phase A and lane L set cardinality histogram for all intersections in LuST scenario.

exploratory. This ensures learning is robust; neither exploitation nor exploration is favoured and the state-action space is constantly being searched.

Finally, before learners begin parameter updates, the experience replay buffer is filled to 75% capacity to ensure a diversity of experiences. After the replay is at sufficient capacity, each network performs U parameter updates using batches B sampled from the replay buffer R. All hyperparameters are outlined in Table 10. These hyperparameters were selected via parameter sweeping in an isolated intersection simulation. Supplemental simulation information is detailed in the Appendix.

Table 10: Hyperparameters

Hyperparameter	Variable	Value
Batch Size	$\|B\|$	32
Max Replay Size	$\|R\|$	5000
Parameter Updates	U	5000
Target Update	τ	0.01
Target Network Period	P	8
Learning Rate	α	0.001
Adam Epsilon	ϵ_{Adam}	0.0001
Discount Rate	γ	0.99
Exploration Rates	ϵ	$\{0.025, 0.5, 1.0\}$
Min green time	g_{min}	5
Max green time	g_{max}	30

!

4.9.4 Testing

To evaluate the policies learned, the proposed reinforcement learning adaptive traffic signal controller π_{RL} is compared to random and fixed time traffic signal controllers.

The random traffic signal controller functions exactly as the reinforcement learning adaptive controller except its policy uniformly samples the next duration over the action space $\pi_{Rand}(s_t) = \text{Unif}(g_{min}, g_{max})$. This policy is used as a control to contrast what has been learned via reinforcement learning.

The fixed time traffic signal controller π_{Fixed} represents a conventional system that is currently used in practice. Fixed time controllers implement a cycle of green phases with static green times. The fixed time controller configuration used is included with the LuST scenario.

Traditional traffic engineering measures of effectiveness, vehicle travel time and stopped time, intersection delay and queue, are used to compare and evaluate the different traffic signal controllers. Each controller is subjected to 40 different random seed, 86400 step (24 hour) simulations to estimate the measures of effectiveness as test metrics. During testing, the proposed reinforcement learning adaptive traffic signal controller does not perform parameter updates and uses a constant exploration rate $\epsilon = 0.025$, significantly reduced from training to evaluate the learned policies performance.

4.10 Results and Discussion

Testing results are displayed in Figures 21 and 22 and Table 11.

The performance of each individual intersection compared with different traffic signal controllers is displayed in Figure 21. Across metrics, the random controller performs the worst, but this is to be expected. The fixed time controller performs better than random, but the reinforcement learning adaptive traffic signal controller achieves the lowest delay and queue in the majority of intersections along with reducing travel time and stopped time across the network. The remaining analysis is focused on comparing the fixed and reinforcement learning controllers as the random controller is uncompetitive.

The proposed reinforcement learning adaptive traffic signal controller decreases delay by a statistically significant amount in 94% (184 out of 196) of intersections, similar results are observed with vehicle queues. However, in 4% (8 out of 196) of intersections, the reinforcement learning policy increases delay and queue. Although performance isn't improved at every individual intersection, the cumulative effect on the network is positive, evidenced by the decrease in vehicle travel and stopped time.

Under the reinforcement learning adaptive traffic signal controller, mean and median travel time and stopped time are reduced, illustrated in Figure 22 and Table 11. This is evidence suggesting that the improvements offered by the reinforcement learning adaptive traffic signal controller overcome the detriments caused by the minority of intersections where the reinforcement learning policy performed worse than the fixed time controller. However, considering 94% of intersections have developed superior policies compared to random and fixed controllers, the authors hypothesize improvements to the learning algorithm could learn optimal policies for all intersections. Although these measures of effectiveness are mobility oriented, they also have environmental consequences. Particularly, reducing stopped time is beneficial to reducing fuel consumption, as the acceleration and deceleration involved with stopping consumes unnecessary fuel.

An important observation is that the variance of the intersections performance can be high, as evidenced by the 2% (i.e., 4 out of 196) of intersections where the null hypothesis can not be rejected for difference of means. This can indicate the learned reinforcement learning policy is unstable or insufficiently estimated. Like many control systems, society requires traffic signal controllers be stable, ensuring variance is minimized would be necessary in practice.

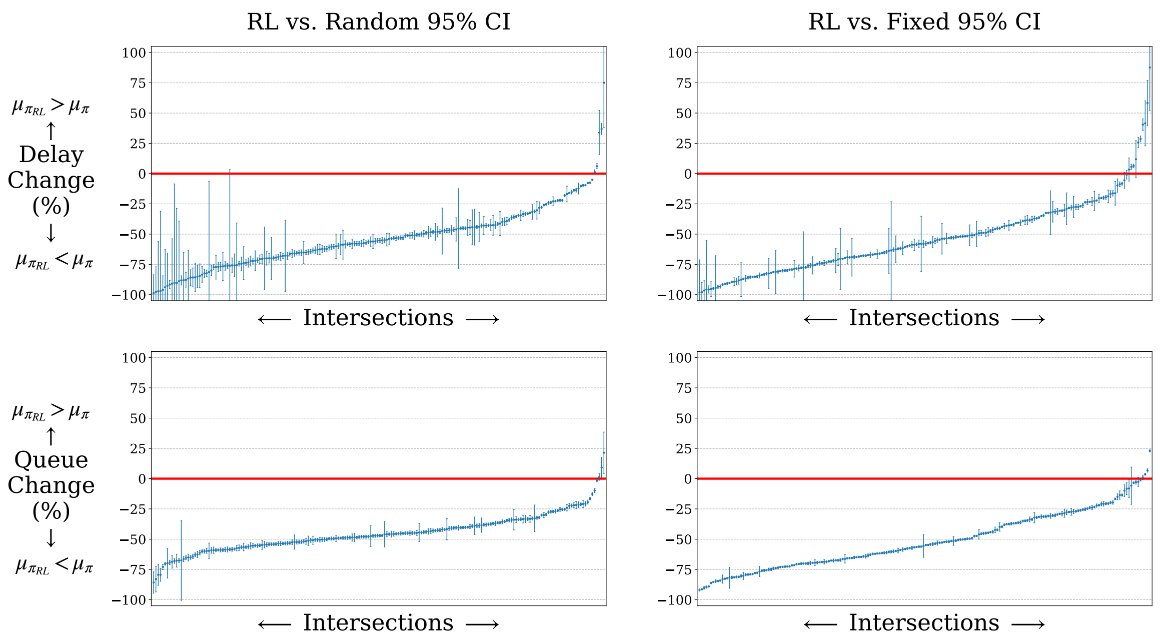


Figure 21: Comparison of mobility measures for different traffic signal controllers at each intersection. Blue vertical lines represent independent, two sample difference of means 95% confidence intervals (CI), used for testing hypotheses $H_0 : \mu_{\pi_{RL}} - \mu_{\pi_{Rand}} = 0$ and $H_0 : \mu_{\pi_{RL}} - \mu_{\pi_{Fixed}} = 0$. Blue lines that do not cross the red line indicate statistically significant different means (i.e., sufficient evidence to reject the null hypothesis H_0).

4.11 Conclusions and Future Work

A reinforcement learning adaptive traffic signal control model was proposed and tested to control 196 intersections in a simulation of the City of Luxembourg. Using a distributed architecture and the DDPG algorithm, the reinforcement learning system reduced travel time, stopped time, queue and delay compared to random and fixed controllers. Despite agent diversity, one set of hyperparameters is sufficient to develop more than 100 deep reinforcement learning agents successfully.

This research contributes to the body of knowledge by developing a scalable architecture capable of training hundreds of reinforcement learning adaptive traffic signal controllers in simulation. This framework can be used on any network as long as a simulation model has been developed.

Many areas for future research exist. Pedestrians were not modelled in the experiments conducted, modelling and observing their behaviour with adaptive signal control is necessary to ensure their quality of service does not diminish.

Many techniques have been developed to improve deep reinforcement learning that were not

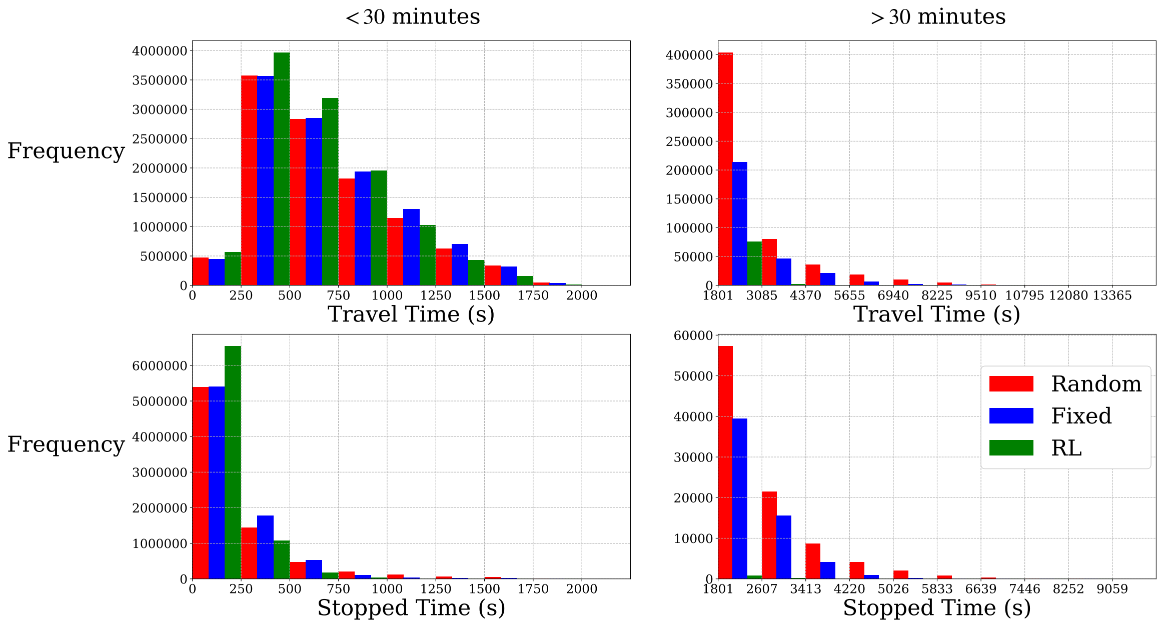


Figure 22: Individual vehicle temporal measures of effectiveness for each traffic signal control method. Times are categorized into short (< 30 minutes) and long (> 30 minutes) durations.

Table 11: Testing results

Traffic Signal Controller	Travel Time (s)	Stopped Time (s)
	$(\mu, \sigma,$ median$, n)$	
Random	(799, 672, 625, 1.1e7)	(262, 394, 144, 7.9e6)
Fixed	(756, 525, 630, 1.1e7)	(236, 295, 160, 8.0e6)
RL	(**651**, 344, **575**, 1.1e7)	(**136**, 143, **92**, 7.9e6)

utilized [47]. Improvements to the learning may develop better policies for the subset of intersections where our proposed model decreased performance.

Recalling that fundamentally traffic signal control requires two decisions at any given time (i.e., selection of the next phase and duration) the proposed model only decides on the next phase duration and uses a cycle for next phase selection. Future models could act on both decisions, choosing the next phase and duration. This would likely require some kind of hierarchical or factorized reinforcement learning architecture, however, researchers have developed and successfully applied these techniques in other domains [126, 109, 127].

4.12 Appendix

Software used include SUMO 0.32.0 [4], Tensorflow 1.8 [128], SciPy [96] and public code [129]. The neural network parameters were intiailized with Xavier [130] and optimized using Adam [107]. During training 40 actors and 8 learners were utilized. The authors found the computational constraint was experience generation, necessitating more actors than learners. However, if more computationally intensive neural networks were used (e.g., convolutional, recurrent), the ratio of actors to learners may need modification. Actors required $\sim$ 4.5 GB and learners $<$ 1 GB of RAM. Training wall clock time was $\sim$ 5 hours with the aforementioned configuration and hyperparameters.

To ensure intersection safety, yellow change and all-red clearance phases of four seconds in duration were inserted between all green phase transitions. By default the fixed time controller included with LuST does not include red clearance phases, which the authors manually include to ensure parity in results. As indicated in Algorithm 1, if no vehicles are present at the intersection, the phase defaults to all-red, which is considered a terminal state $s_{terminal}$. Each intersection's state observation is bounded by the shortest incoming lane or 125 m (i.e., the queue and density are calculated from vehicles up to a maximum of 125 m from the intersection stopline). The dynamic user assignment (dua) configuration of the LuST scenario is used for training and testing with random seeds.

4.13 Acknowledgments

The authors would like to acknowledge this research would not have been possible without freely available software. Specifically, [5, 6] and [4] whose free distribution of their work was an integral part of this research.

This research was enabled in part by support in the form of computing resources provided by SHARCNET (`www.sharcnet.ca`), their McMaster University staff and Compute Canada (`www.computecanada.ca`).

5 CONCLUSION

5.1 Contributions

Transportation systems exhibit significant costs to the environment, human health and mobility when governed by systems which act sub-optimally. Optimal control and decision-making in transportation systems reduces costs and benefits society at large. To improve mobility and efficiency in transportation systems, this thesis systematically developed an adaptive traffic signal control system using reinforcement learning and validates it at scale in a realistic city simulation. This thesis contributes to the academic body of knowledge:

- **Adaptive Traffic Signal Controller:** This thesis develops three different reinforcement learning adaptive traffic controllers, Q-learning, A3C and DDPG, in simulation all with parallel computing methods that decrease development time and allow for future scalability. Using quantitative traffic measures of effectiveness, simulation evidence demonstrated the developed methods can improve upon sub-optimal control from conventional traffic signal controllers. Explicitly, the developed adaptive traffic signal controllers decrease delay, queues, stopped time and travel time; consequently reducing vehicle emissions and improving human health. The developed adaptive traffic signal controller was validated at scale, on a 24 hour simulation of the City of Luxembourg, Luxembourg, controlling 196 intersections independently without communication. To the best of the author's knowledge, this is the largest application of adaptive traffic signal controller in simulation to date, contributing to the state of the art practice.
- **Explicit Policy Representation:** From a machine learning perspective, this thesis contributes evidence that explicitly parameterizing the policy offers improved performance for reinforcement learning adaptive traffic signal control, concurring with an ensemble of research in other domains [46, 131, 48].
- **Flexible Reinforcement Learning:** This thesis demonstrated reinforcement learning is flexible in implementation for adaptive traffic signal control. In answering the question what decisions should an adaptive traffic signal controller make, this research demonstrated both acyclic (i.e., choose the next phase, static duration) and cycle-based (i.e., next phase from cycle, dynamic duration) adaptive traffic signal controllers can be used successfully. However, acyclic controllers may learn to ignore some phases, diminishing some intersection users quality of service, as evidenced in Chapter 2. As stability and fairness is desired in traffic signal

controllers, cycle based adaptive traffic signal controllers developed with reinforcement learning are suggested by the evidence in this research. Additional research included as Appendix B demonstrated a variety of state representations can be used, ranging from low-resolution data provided by loop detectors to high-resolution data from cameras [132]. The reward can be defined as a function of queue, delay or stopped time, all of which achieve favourable results. This flexibility in problem definition when using reinforcement learning for adaptive traffic signal control gives practitioners and academics many options for successful implementation.

These research findings and contributions also have broader impacts that are useful to a variety of parties. Traffic engineers can use the methods proposed in this thesis to develop adaptive traffic signal controllers which can cope with the increasing amounts of traffic data. This thesis details how to develop control logic for adaptive traffic signal controllers at scale, guiding those interested in practical implementation. For researchers and academics, this thesis demonstrates reinforcement learning can be used to provide solutions to real-world, practical problems, beyond the video games and toy problems commonly found in the literature. Policy-makers could learn what technological solutions are available, and capable of, in solving society's transportation problems.

5.2 Limitations & Future Work

This thesis contributes a system for developing adaptive traffic signal control logic and provides a foundation for many areas of future inquiry. Foremost, practical implementation would require expertise from a multitude of disciplines, most importantly the hardware necessary to integrate the methods developed in this research with transportation infrastructure.

Transportation systems are multi-modal. In developing an adaptive traffic signal controller, this research only modelled road based vehicles in simulation. In reality, traffic signal controllers affect users of other transportation modes, such as pedestrians and cyclists. Cyclists share the road with vehicles but exhibit different behaviour. Pedestrians have their own, parallel infrastructure but are still influenced by a traffic signal controller's policy when they want to traverse the intersection. Pedestrians in particular place constraints on traffic signal controllers, requiring minimum amounts of time to safely cross the intersection. While efforts were taken in this thesis to respect these constraints without explicitly modelling these other users/modes, future work should overcome this limitation by consulting previous efforts [133, 100, 134] and include additional modes in their simulation models. Transit and other active modes (i.e., walking, cycling) could be incorporated into reinforcement learning adaptive traffic signal control by modifying the reward function used in training. For example, a reward function defined at the passenger scale instead of the vehicle scale could be used to favour transit over single passenger vehicles. These changes could be used by transportation agencies interested in promoting a more efficient, sustainable transportation system.

A true multi-agent framework was not used in this research. While any adaptive traffic signal control architecture exhibits its own advantages and disadvantages, using multi-agent systems techniques is already a promising area of inquiry. Either through centralized control or decentralization with communication, multi-agent adaptive traffic signal control has already been the focus of much research [64, 98, 135]. Specifically in near saturated and over saturated traffic demand, multi-agent systems would be able to coordinate their behaviour to achieve their goals perhaps better than individual, isolated agents. However, any performance improvements achieved with multi-agent systems would need to be weighed against the additional resource requirements (i.e., centralization, communication).

It is important to know how drivers would respond to adaptive traffic signal control in practice. Drivers often traverse familiar routes (e.g., from home to work and back) and become accustomed to certain regularities, developing expectations about their travel. For example, a daily commuter will become familiar with the functioning of traffic signal controllers on their route. Although it is hypothesized that an adaptive traffic signal controller's decisions would be respected by drivers, it is yet unknown if a driver's familiar expectations will conflict with an adaptive controller. As any traffic signal controller must ensure safe control of the intersection, it is necessary to study driver's response to adaptive control to guarantee safety isn't compromised.

Future research could also focus on technical aspects. Only fully connected neural networks were used for function approximation; other neuron connectivities (e.g., convolutional, recurrent, residual) could offer improved performance. Despite the parsimony offered by model-free reinforcement learning methods compared to model-based methods, traffic demand exhibits recurring patterns from which demand models could be developed and used with model-based reinforcement learning methods. Incorporating some form of continous learning [136] after field deployment is also a promising area of future inquiry. Traditionally, deep reinforcement learning agents train for some fixed duration and then perform no further updates to their model parameters. Developing the capability to learn at all times will likely further improve performance in non-stationary environments like traffic signal control. Bayesian methods are particularly well-suited to develop continuous reinforcement learning models, with some successful research already conducted for adaptive traffic signal control [88, 89, 90]

It is worth noticing that although this thesis successfully developed adaptive traffic signal controllers, this success is actually managing symptoms of a problem and not its cause. The real problem is not one of a lack of infrastructure or even completely inadequate control methods, but a demand-time-capacity problem. For the majority of the time, demand of traffic infrastructure is below capacity. It is only for a fraction of the time (e.g., work day morning/evening commute) that traffic demand exceeds infrastructure capacity and problems arise. Fundamentally, if traffic demand never exceeded infrastructure capacity, this would eliminate many of the problems and inefficiencies currently experienced. Society expects solutions to the problem as it currently exists, but what if solutions were developed which prevented the problem from manifesting in the first place? Some unorthodox, perhaps quixotic, solutions are worth consideration. An obvious solution is public transit, which uses separate infrastructure or uses shared infrastructure more efficiently. Another possibility would be to restructure society such that demand is more uniformly distributed over time. Peak demands occur when the majority of people travel to and from work. If society's working hours were restructured in a way that wasn't so bi-modal, demand wouldn't exceed capacity as often. Both of these solutions would require significant collective change within a population, but they are nonetheless worth considering. Implementing solutions of these kind lie outside the domain of engineering, likely requiring policy and political action, and to some may be considered unrealistic. Nevertheless, an objective account of this problem should consider all solutions, as the increasing costs incurred necessitate considering all possibilities.

Ensuring transportations are efficient transportation systems will continue to be among the foremost concerns for societies. This research successfully developed adaptive traffic signal control solutions, addressing present and future transportation concerns with benefits to mobility, human health and the environment.

REFERENCES

[1] W. Genders and S. Razavi, "Asynchronous n-step q-learning adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, 2018. doi:10.1080/15472450.2018.1491003. *Accepted, In Press*.

[2] W. Genders and S. Razavi, "Policy analysis of reinforcement learning adaptive traffic signal control," *ASCE Journal of Computing in Civil Engineering*, 2018. *4th round of revisions*, CPENG-2667R4.

[3] W. Genders and S. Razavi, "Distributed deep deterministic policy gradients for adaptive traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2018. *Submitted 15 August 2018*, T-ITS-18-08-0805.

[4] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.

[5] L. Codeca, R. Frank, and T. Engel, "Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research," in *Proceedings of the 7th IEEE Vehicular Networking Conference*, pp. 1–8, 2015.

[6] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017.

[7] S. Pandian, S. Gokhale, and A. K. Ghoshal, "Evaluating effects of traffic and vehicle characteristics on vehicular emissions near traffic intersections," *Transportation Research Part D: Transport and Environment*, vol. 14, no. 3, pp. 180–196, 2009.

[8] P. Goodwin, "The economic costs of road traffic congestion," 2004.

[9] M. M. Baker, "White paper european transport policy for 2010: time to decide," 2001.

[10] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 urban mobility scorecard," tech. rep., The Texas A&M Transportation Institute and INRIX, 2015. `https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-scorecard-2015.pdf`.

[11] CPCS, "Grinding to a halt: Evaluating canada's worse bottlenecks," tech. rep., Canadian Automobile Association, 2017.

[12] G. Cookson, "INRIX global traffic scorecard," tech. rep., INRIX, 2018.

[13] X. Liu, F. Lu, H. Zhang, and P. Qiu, "Intersection delay estimation from floating car data via principal curves: a case study on beijing's road network," *Frontiers of earth science*, vol. 7, no. 2, pp. 206–216, 2013.

[14] T. Liao and R. Machemehl, "Optimal traffic signal strategy for fuel consumption and emissions control at signalized intersections," in *Proceedings of 24th European Transport Forum, Brunel University, England*, 1996.

[15] International Energy Agency, "CO2 emissions from fuel combustion 2017 highlights." `https://www.iea.org/publications/freepublications/publication/CO2EmissionsfromFuelCombustionHighlights2017.pdf`, 2017.

[16] World Health Organization *et al.*, "Ambient air pollution: A global assessment of exposure and burden of disease," 2016.

[17] R. A. Silva, Z. Adelman, M. M. Fry, and J. J. West, "The impact of individual anthropogenic emissions sectors on the global burden of human mortality due to ambient air pollution," *Environmental health perspectives*, vol. 124, no. 11, p. 1776, 2016.

[18] W. Q. Gan, H. W. Davies, M. Koehoorn, and M. Brauer, "Association of long-term exposure to community noise and traffic-related air pollution with coronary heart disease mortality," *American journal of epidemiology*, vol. 175, no. 9, pp. 898–906, 2012.

[19] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa, "Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam," *New journal of physics*, vol. 10, no. 3, p. 033001, 2008.

[20] Transportation Research Board and National Academies of Sciences, Engineering, and Medicine, *Signal Timing Manual - Second Edition.* Washington, DC: The National Academies Press, 2015.

[21] T. Li, D. Zhao, and J. Yi, "Adaptive dynamic programming for multi-intersections traffic signal intelligent control," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pp. 286–291, IEEE, 2008.

[22] C. Cai, C. K. Wong, and B. G. Heydecker, "Adaptive traffic signal control using approximate dynamic programming," *Transportation Research Part C: Emerging Technologies*, vol. 17, no. 5, pp. 456–474, 2009.

[23] S. Chiu and S. Chand, "Adaptive traffic signal control using fuzzy logic," in *Fuzzy Systems, 1993., Second IEEE International Conference on*, pp. 1371–1376, IEEE, 1993.

[24] M. C. Choy, D. Srinivasan, and R. L. Cheu, "Cooperative, hybrid agent architecture for real-time traffic signal control," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: systems and humans*, vol. 33, no. 5, pp. 597–607, 2003.

[25] D. J. Montana and S. Czerwinski, "Evolving control laws for a network of traffic signals," in *Proceedings of the 1st annual conference on genetic programming*, pp. 333–338, MIT Press, 1996.

[26] B. Park, C. Messer, and T. Urbanik, "Traffic signal optimization program for oversaturated conditions: genetic algorithm approach," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1683, pp. 133–142, 1999.

[27] H. Ceylan and M. G. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing," *Transportation Research Part B: Methodological*, vol. 38, no. 4, pp. 329–342, 2004.

[28] P. Hunt, D. Robertson, R. Bretherton, and R. Winton, "Scoot-a traffic responsive method of coordinating signals," tech. rep., 1981.

[29] N. Gartner, "A demand-responsive strategy for traffic signal control," *Transportation Research Record*, vol. 906, pp. 75–81, 1983.

[30] P. Lowrie, "Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic," 1990.

[31] P. Mirchandani and L. Head, "A real-time traffic signal control system: architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[32] F. Luyanda, D. Gettman, L. Head, S. Shelby, D. Bullock, and P. Mirchandani, "Acs-lite algorithmic architecture: applying adaptive control system technology to closed-loop traffic signal control systems," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1856, pp. 175–184, 2003.

[33] M. Clarke, "Big data in transport," tech. rep., The Institution of Engineering and Technology, 2018. `https://www.theiet.org/sectors/transport/topics/intelligent-mobility/files/sector-insight.cfm`.

[34] G. E. Moore, “Cramming More Components onto Integrated Circuits,” *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.

[35] R. A. Howard, “Dynamic programming and markov processes,” *Science*, vol. 132, no. 3482, 1964.

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[37] R. Bellman, “The theory of dynamic programming,” tech. rep., RAND Corp Santa Monica CA, 1954.

[38] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[39] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*, vol. 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[40] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[41] S. M. Kakade, “A natural policy gradient,” in *Advances in neural information processing systems*, pp. 1531–1538, 2002.

[42] S. Haykin, *Neural networks and learning machines*, vol. 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[43] S. Linnainmaa, “Taylor expansion of the accumulated rounding error,” *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[44] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, p. 533, 1986.

[45] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[46] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[47] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," *arXiv preprint arXiv:1710.02298*, 2017.

[48] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[49] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, *et al.*, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *arXiv preprint arXiv:1807.01281*, 2018.

[50] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[51] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[53] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3389–3396, IEEE, 2017.

[54] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[55] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *Journal de physique I*, vol. 2, no. 12, pp. 2221–2229, 1992.

[56] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[57] T. L. Thorpe and C. W. Anderson, "Traffic light control using sarsa with three state representations," tech. rep., Citeseer, 1996.

[58] E. Bingham, "Neurofuzzy traffic signal control," 1998.

[59] M. Wiering *et al.*, "Multi-agent reinforcement learning for traffic light control," in *ICML*, pp. 1151–1158, 2000.

[60] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[61] D. de Oliveira, A. L. Bazzan, B. C. da Silva, E. W. Basso, L. Nunes, R. Rossetti, E. de Oliveira, R. da Silva, and L. Lamb, "Reinforcement learning based control of traffic lights in non-stationary environments: A case study in a microscopic simulator.," in *EUMAS*, 2006.

[62] D. Srinivasan, M. C. Choy, and R. L. Cheu, "Neural networks for real-time traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 3, pp. 261–272, 2006.

[63] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[64] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[65] M. A. Khamis and W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 134–151, 2014.

[66] T. Rijken, *DeepLight: Deep reinforcement learning for signalised traffic control.* PhD thesis, Master's Thesis. University College London, 2015.

[67] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[68] E. van der Pol, *Deep reinforcement learning for coordination in traffic light control.* PhD thesis, Master's Thesis. University of Amsterdam, 2016.

[69] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016. `https://arxiv.org/abs/1611.01142`.

[70] S. Mikami and Y. Kakazu, "Genetic reinforcement learning for cooperative traffic signal control," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 223–228, IEEE, 1994.

[71] E. Bingham, "Reinforcement learning in neurofuzzy traffic signal control," *European Journal of Operational Research*, vol. 131, no. 2, pp. 232–241, 2001.

[72] F. Webster, "Traffic signal settings, road research technical paper no. 39," *Road Research Laboratory*, 1958.

[73] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[74] P. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[75] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2011.

[76] N. Casas, "Deep deterministic policy gradient for urban traffic light control," *arXiv preprint arXiv:1703.09035*, 2017.

[77] S. S. Mousavi, M. Schukat, P. Corcoran, and E. Howley, "Traffic light control using deep policy-gradient and value-function based reinforcement learning," *arXiv preprint arXiv:1704.08883*, 2017.

[78] J. Lee, B. Abdulhai, A. Shalaby, and E.-H. Chung, "Real-time optimization for adaptive traffic signal control using genetic algorithms," *Journal of Intelligent Transportation Systems*, vol. 9, no. 3, pp. 111–122, 2005.

[79] T. Wongpiromsarn, T. Uthaicharoenpong, Y. Wang, E. Frazzoli, and D. Wang, "Distributed traffic signal control for maximum network throughput," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 588–595, IEEE, 2012.

[80] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2015.

[81] S. Timotheou, C. G. Panayiotou, and M. M. Polycarpou, “Distributed traffic signal control using the cell transmission model via the alternating direction method of multipliers,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 919–933, 2015.

[82] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, “Optimizing traffic lights in a cellular automaton model for city traffic,” *Physical Review E*, vol. 64, no. 5, p. 056132, 2001.

[83] Y. K. Chin, N. Bolong, A. Kiring, S. S. Yang, and K. T. K. Teo, “Q-learning based traffic optimization in management of signal timing plan,” *International Journal of Simulation, Systems, Science & Technology*, vol. 12, no. 3, pp. 29–35, 2011.

[84] M. Abdoos, N. Mozayani, and A. L. Bazzan, “Holonic multi-agent system for traffic signals control,” *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5, pp. 1575–1587, 2013.

[85] H. A. Aziz, F. Zhu, and S. V. Ukkusuri, “Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility,” *Journal of Intelligent Transportation Systems*, vol. 22, no. 1, pp. 40–52, 2018.

[86] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Design of reinforcement learning parameters for seamless application of adaptive traffic signal control,” *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 227–245, 2014.

[87] P. Mannion, J. Duggan, and E. Howley, “An experimental review of reinforcement learning algorithms for adaptive traffic signal control,” in *Autonomic Road Transport Support Systems*, pp. 47–66, Springer, 2016.

[88] M. A. Khamis, W. Gomaa, A. El-Mahdy, and A. Shoukry, “Adaptive traffic control system based on bayesian probability interpretation,” in *Electronics, Communications and Computers (JEC-ECC), 2012 Japan-Egypt Conference on*, pp. 151–156, IEEE, 2012.

[89] M. A. Khamis, W. Gomaa, and H. El-Shishiny, “Multi-objective traffic light control system based on bayesian probability interpretation,” in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 995–1000, IEEE, 2012.

[90] M. A. Khamis and W. Gomaa, “Enhanced multiagent multi-objective reinforcement learning for urban traffic light control,” in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1, pp. 586–591, IEEE, 2012.

[91] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[92] C. Hill and G. Krueger, "ITS ePrimer Module 13: Connected vehicles." `https://www.pcb.its.dot.gov/eprimer/module13.aspx`.

[93] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[94] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[95] S. Kapturowski, "Tensorflow-rl." `https://github.com/steveKapturowski/tensorflow-rl`, 2017.

[96] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001. `"http://www.scipy.org/"`.

[97] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 34, 2017.

[98] F. Zhu, H. A. Aziz, X. Qian, and S. V. Ukkusuri, "A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 487–501, 2015.

[99] J. Jin and X. Ma, "A group-based traffic signal control with adaptive learning ability," *Engineering applications of artificial intelligence*, vol. 65, pp. 282–293, 2017.

[100] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 732–752, 2017.

[101] J. Peters, S. Vijayakumar, and S. Schaal, "Natural actor-critic," in *European Conference on Machine Learning*, pp. 280–291, Springer, 2005.

[102] S. Richter, D. Aberdeen, and J. Yu, "Natural actor-critic for road traffic optimisation," in *Advances in neural information processing systems*, pp. 1169–1176, 2007.

[103] L. Chun-Gui, W. Meng, S. Zi-Gaung, L. Fei-Ying, and Z. Zeng-Fang, "Urban traffic signal learning control using fuzzy actor-critic methods," in *Natural Computation, 2009. ICNC'09. Fifth International Conference on*, vol. 1, pp. 368–372, IEEE, 2009.

[104] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *American Control Conference (ACC), 2012*, pp. 2177–2182, IEEE, 2012.

[105] Toronto, "Traffic signal operations policies and strategies," April 2018. `https://www.toronto.ca/wp-content/uploads/2017/11/91d6-0\_2015-11-13\_Traffic-Signal-Operations-Policies-and-Strategies\_Final-a.pdf`.

[106] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[107] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[108] S. Goel, S. F. Bush, and C. Gershenson, "Self-organization in traffic lights: Evolution of signal control with advances in sensors and communications," *arXiv preprint arXiv:1708.07188*, 2017.

[109] S. Sharma, A. S. Lakshminarayanan, and B. Ravindran, "Learning to repeat: Fine grained action repetition for deep reinforcement learning," *arXiv preprint arXiv:1702.06054*, 2017.

[110] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.

[111] I. Higgins, A. Pal, A. A. Rusu, L. Matthey, C. P. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner, "Darla: Improving zero-shot transfer in reinforcement learning," *arXiv preprint arXiv:1707.08475*, 2017.

[112] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[113] L. Wu, Y. Ci, J. Chu, and H. Zhang, "The influence of intersections on fuel consumption in urban arterial road traffic: a single vehicle test in harbin, china," *PloS one*, vol. 10, no. 9, 2015.

[114] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[115] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[116] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[117] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, “Residual attention network for image classification,” *arXiv preprint arXiv:1704.06904*, 2017.

[118] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, “Distributed prioritized experience replay,” *arXiv preprint arXiv:1803.00933*, 2018.

[119] X. Liang, X. Du, G. Wang, and Z. Han, “Deep reinforcement learning for traffic light control in vehicular networks,” *arXiv preprint arXiv:1803.11115*, 2018.

[120] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, “Multiagent reinforcement learning for urban traffic control using coordination graphs,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 656–671, Springer, 2008.

[121] J. C. Medina and R. F. Benekohal, “Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy,” in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 596–601, IEEE, 2012.

[122] T. Chu, S. Qu, and J. Wang, “Large-scale traffic grid signal control with regional reinforcement learning,” in *American Control Conference (ACC), 2016*, pp. 815–820, IEEE, 2016.

[123] W. Liu, G. Qin, Y. He, and F. Jiang, “Distributed cooperative reinforcement learning-based traffic signal control that integrates v2x networks’ dynamic clustering,” *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8667–8681, 2017.

[124] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[125] L.-J. Lin, “Self-improving reactive agents based on reinforcement learning, planning and teaching,” *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[126] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation,” in *Advances in neural information processing systems*, pp. 3675–3683, 2016.

[127] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1703.01161*, 2017.

[128] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[129] M. Zhou, "Reinforcement-learning-with-tensorflow," 2018. Accessed on 15 July 2018.

[130] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[131] A. Gruslys, M. G. Azar, M. G. Bellemare, and R. Munos, "The reactor: A sample-efficient actor-critic architecture," *arXiv preprint arXiv:1704.04651*, 2017.

[132] W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," *Procedia computer science*, vol. 130, pp. 26–33, 2018. `https://doi.org/10.1016/j.procs.2018.04.008`.

[133] Q. He, K. L. Head, and J. Ding, "Multi-modal traffic signal control with priority, signal actuation and coordination," *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 65–82, 2014.

[134] Y. Zhang, K. Gao, Y. Zhang, and R. Su, "Traffic light scheduling for pedestrian-vehicle mixed-flow networks," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–16, 2018.

[135] S. Araghi, A. Khosravi, and D. Creighton, "Intelligent cuckoo search optimized traffic signal controllers for multi-intersection network," *Expert Systems with Applications*, vol. 42, no. 9, pp. 4422–4431, 2015.

[136] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *arXiv preprint arXiv:1802.07569*, 2018.

[137] G. Tesauro, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[138] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[139] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[140] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[141] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2018.

[142] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[143] R. Riccardo and G. Massimiliano, "An empirical analysis of vehicle time headways on rural two-lane two-way roads," *Procedia-Social and Behavioral Sciences*, vol. 54, pp. 865–874, 2012.

[144] A. Maurya, S. DEY, and S. DAS, "Speed and time headway distribution under mixed traffic condition," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 11, no. 0, pp. 1774–1792, 2015.

[145] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory.," *Psychological review*, vol. 102, no. 3, p. 419, 1995.

[146] J. O'Neill, B. Pleydell-Bouverie, D. Dupret, and J. Csicsvari, "Play it again: reactivation of waking experience and memory," *Trends in neurosciences*, vol. 33, no. 5, pp. 220–229, 2010.

[147] L.-J. Lin, "Reinforcement learning for robots using neural networks," tech. rep., DTIC Document, 1993.

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

[149] S. Kapturowski, "Tensorflow-rl," 2017. `https://github.com/steveKapturowski/tensorflow-rl`.

APPENDIX A: DQN-TSC

This manuscript was completed in the early stages of thesis research and was published to the preprint website arXiv under the following reference:

W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016. `https://arxiv.org/abs/1611.01142`

The co-author's contributions to the above work include:

- Financial and technical supervision of the study presented in this work.
- Manuscript editing.

Abstract

Ensuring transportation systems are efficient is a priority for modern society. Technological advances have made it possible for transportation systems to collect large volumes of varied data on an unprecedented scale. We propose a traffic signal control system which takes advantage of this new, high quality data, with minimal abstraction compared to other proposed systems. We apply modern deep reinforcement learning methods to build a truly adaptive traffic signal control agent in the traffic microsimulator SUMO. We propose a new state space, the discrete traffic state encoding, which is information dense. The discrete traffic state encoding is used as input to a deep convolutional neural network, trained using Q-learning with experience replay. Our agent was compared against a one hidden layer neural network traffic signal control agent and reduces average cumulative delay by 82%, average queue length by 66% and average travel time by 20%.

Introduction

Modern society relies on its many transportation systems for the movement of individuals, goods and services. Ensuring vehicles can move efficiently from their origin to destination is desirable by all. However, increasing population, and subsequent vehicle ownership, has increased the demand of road infrastructure often beyond its capacity, resulting in congestion, travel delays and unnecessary vehicle emissions. To address this problem, two types of solutions are possible. The first is to increase capacity by expanding road infrastructure, however this can be expensive, protracted and decrease capacity in the short term. The second solution is to increase the efficiency of existing infrastructure and the systems that govern them, such as traffic signal controllers (TSC). We advocate this second solution, by utilizing recent advancements from the domain of artificial intelligence [45] to develop a new traffic signal controller.

We define the traffic signal control problem as follows; given the state of traffic at an intersection, what is the optimal traffic signal phase and sequence that should be enacted? Many systems have been proposed that utilize new sensors, particularly reinforcement learning for traffic signal control, however they do not take full advantage of the available data. We propose a deep artificial neural network as a traffic signal control agent (TSCA), trained using reinforcement learning, that strives to solve the traffic signal control problem by developing an optimal control policy.

Reinforcement learning is a machine learning paradigm where an agent seeks to maximize cumulative reward by developing a state-action policy through repeated interaction with its environment. Reinforcement learning agents achieve optimal control with respect to a defined reward by developing an optimal state-action policy. Function approximators, such as artificial neural networks, have been used in reinforcement learning to approximate value functions when the agent's representation of the environment, or state space, becomes too large [137]. Convolutional neural networks, a specific type of network architecture, are inspired by biological research on the animal visual cortex [138][139] and have displayed impressive performance [140]. They apply the mathematical convolution operation between various filters and the layer input to produce feature maps. Convolutional networks are advantageous because minimal input pre-processing is required and they can develop their own features. We develop a deep Q-network traffic signal control agent (DQTSCA), with the action-value function modelled as a deep convolutional neural network trained using reinforcement learning in a traffic microsimulator, SUMO, on an isolated intersection.

Reinforcement learning is a suitable technique for attempting to solve the traffic signal control problem, as it elegantly represents the elements of the problem - agent (traffic signal controller), environment (state of traffic) and actions (traffic signals). Previous research using reinforcement learning for traffic signal control has yielded impressive results [74][75][64], yet we perceive areas for improvement. We propose a new state space definition, the discrete traffic state encoding (DTSE), as an improved representation of traffic, as it contains more relevant information compared to previous research's state space definitions. The DTSE is proposed as it is information dense; the convolutional neural network is required to take advantage of the information dense state. The DTSE will allow the convolutional neural network to perceive more relevant traffic information than previous research, extract useful features and develop high-level state representations. The agent can then achieve optimal control by choosing the actions with the highest value, or maximum expected cumulative reward.

The succeeding sections are organized as follows: the Literature Review details research conducted in the domain of traffic signal control and reinforcement learning, the Proposed System describes the proposed DQTSCA and defines the state space, action space and reward, the Experimental Setup and Training details the tools used to implement the proposed agent and describes its training, Results and Discussion discusses the results and performance of the agent and the Conclusion summarizes the research conducted and provides ideas for future work.

Literature Review

Significant research has been conducted using reinforcement learning for traffic signal control. Early efforts were limited by simple simulations and a lack of computational power [57][59][82][60]. Beginning in the early 2000's, continuous improvements in both of these areas have created a variety of simulation tools that are increasingly complex and realistic. Traffic microsimulators are the most popular tool used by traffic researchers, as they model individual vehicles as distinct entities and can reproduce real-world traffic behavior such as shockwaves. Research conducted has differed in reinforcement learning type, state space definition, action space definition, reward definition, simulator, traffic network geometry and vehicle generation model. Previous research efforts have defined the state space as some attribute of traffic, the number of queued vehicles [59][60][83][84] and traffic flow [63][74] the most popular. The action space has been defined as all available signal phases [63][64] or restricted to green phases only [74][83][84]. The most common reward definitions are change in delay [63][64] and change in queued vehicles [74][83][84]. For a comprehensive review of reinforcement learning traffic signal control research, the reader is referred to [86] and [87].

Regarding previous research, the following observations can be made. First, the majority of state definitions are abstractions of the traffic state which omit relevant information. A reinforcement learning agent must first observe the state of the environment before it can act, if useful information is missing, it is unlikely to be able to act optimally. For example, if the state space is defined as the number of queued vehicles at the intersection, this ignores all of the moving vehicles, as well as the queued vehicles' lane and queue position. We believe the state space definition should include as much relevant information about the traffic state as possible, including vehicles' location and speed, thus our proposal of the DTSE, formally defined in the Proposed System section. We recognize that in practice it may be difficult for a TSCA to observe the state of all vehicles' location and speed, but we will defend this assumption in succeeding sections. However, some previous research has proposed a similar, less abstracted, yet limited, state definition [57], from which our research acknowledges their contribution and seeks to extend beyond their efforts.

Second, the TSCA should be given as much action autonomy as possible, therefore it must be recognized that defining the action space as choosing between fixed sequences of signal phases is limiting. For example, if we define that an advance left green signal phase must always precede a through green signal phase, this assumes the optimal policy follows such a sequence. However, it is conceivable that the optimal action given a certain traffic state is to have an advance left green signal phase succeed a through green signal phase. Much of the previous research has constrained the agent's action in such a way; our action space definition seeks to endow the agent with a higher degree of autonomy in an attempt to learn the optimal policy.

Finally, all previous research have used computer simulations, as real-world experimentation is infeasible for various reasons. The majority of research assumes vehicle generation can be modelled as a Poisson process, which relies upon the negative exponential distribution, to model the time between vehicle generation events. We propose in subsequent sections that the negative exponential is not the best distribution to model real traffic, as empirical research has shown other distributions to more accurately model different vehicle generation flow rates.

Proposed System

Attempting to solve the traffic signal control problem using reinforcement learning requires a formulation of the problem in the language of reinforcement learning, specifically, defining a state space S, an action space A and a reward R.

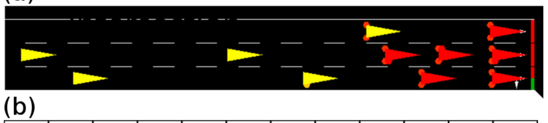
State Space

We propose the DTSE as the appropriate state space S in this research, inspired by a common technique in computing of discretization and quantization of continuous entities. For each lane approaching the intersection, the DTSE discretizes a length l of the lane segment, beginning at the stop line, into cells of length c. The selection of c will change the behavior of system. If c is many times larger than the average vehicle length, the individual dynamics of each vehicle will be lost, however computational cost will be reduced. If c is much smaller than the average vehicle length, the individual vehicle dynamics will be retained, however the computational cost will increase, perhaps unnecessarily. We mention the selection of c is important, however for this research we select c in a simplified manner in an attempt to evaluate the proposed system.

The DTSE is composed of three vectors, the first representing the presence of a vehicle or not in the cell, the second the speed of the vehicle and the third the current traffic signal phase (i.e., the most recent action selected). The addition of second speed vector is an extension beyond [57], as their state definition only consists of a vector representing the presence of a vehicle. Therefore, the state of traffic at an intersection with n lanes is formally defined as the DTSE, where $S \in (\mathbb{B} \times \mathbb{R})^{\frac{l}{c} \times n} \times P$ and P represents the current traffic signal phase. At time t, the agent observes the traffic state (i.e., the DTSE) as $s_t \in S$. A representation of the DTSE can be seen in Fig. 23, with triangles representing vehicles travelling from left to right. In Fig. 23, Fig. 23 (a) shows simulated vehicles approaching the intersection, Fig. 23 (b) is the Boolean-valued vector of the DTSE, encoding the presence or absence of a value and Fig. 23 (c) is the real-valued vector of the DTSE, encoding the normalized speed.

The motivation behind the DTSE is to retain useful information. If the agent is to discover the optimal policy, it must discover the optimal actions for any given state; having knowledge of approaching vehicle's speed and position is conjectured to be superior to only the number of queued vehicles or vehicle flow. The first vector's elements are Boolean-valued, with a one representing the presence of a vehicle and a zero representing the absence of a vehicle. The second vector's elements are real numbers and represent the vehicle's speed, normalized by the speed limit. Each element of P represents a different traffic phase and all elements of P are zero except for the current phase,

(a)



(b)

0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1	0	1	1
0	1	0	0	0	0	1	0	0	1	0	1

(c)

0	0	0	0	0	0	0	0	0.2	0	0	0
0.8	0	0	0	0.5	0	0	0	0	0	0	0
0	0.6	0	0	0	0	0.4	0	0	0	0	0

Figure 23: Example of simulated traffic (a) with corresponding Boolean- (b) and real-valued DTSE vectors (c).

which is one, therefore $P \in \mathbb{B}^{|A|}$.

Technologies over the last decade have made gathering information required for the DTSE possible. Video cameras [91] are becoming more common as sensor devices at intersections and vehicles with wireless communication capabilities (i.e., Connected Vehicles [92]) are expected to be deployed in the near future. Ultimately, the DTSE is sensor agnostic, the means by which the state information is gathered, be it vision, wireless or otherwise, is irrelevant to creating the DTSE. The flexibility in generating the DTSE should be seen as an advantage of the system.

Action Space

After the agent has observed the state of the environment, it must choose one action from the set of all available actions. In this research, the agent's possible actions are the traffic signal phase configurations (i.e., the combination of traffic lights controlling individual lanes for the entire intersection). For simplicity and human comprehension, each action is assigned a compass direction indicating the approaching lanes' traffic signal phases (i.e., the color of the traffic signal lights) and abbreviated for brevity. For explicit clarity, a green traffic signal phase means vehicles can proceed through the intersection, yellow cautions vehicles to slow down and prepare to stop and red means vehicles should stop and not proceed through the intersection. The possible actions are North-South Green (NSG), East-West Green (EWG), North-South Advance Left Green (NSLG), East-West Advance Left Green (EWLG). Note, for any given action, it is implied that the omitted compass direction traffic signals are red (e.g., East-West Green means that all North-South traffic signals are red).

Formally the set of all possible actions A is defined as $A = \{\text{NSG, EWG, NSLG, EWLG}\}$. Therefore, at time t, the agent chooses an action a_t, where $a_t \in A$. However, when an agent chooses an action, it may not be immediately enacted. To ensure safe control of the intersection,

Table 12: Traffic Signal Phase Action Transitions

		Selected Action			
		NSG	**EWG**	**NSLG**	**EWLG**
Current Traffic Signal Phase	**NSG**	-	{NSY, R}	{NSY}	{NSY, R}
	EWG	{EWY, R}	-	{EWY, R}	{EWY}
	NSLG	-	{NSY, R}	-	{NSY, R}
	EWLG	{EWY}	-	{EWY. R}	-

additional traffic signal phase configurations may precede the chosen action. Instead of immediately transitioning from the current traffic signal phase to the selected action, a sequence of intermediate traffic signal phases dependent on the current phase and chosen action may be necessary. All possible action transition sequences to transition from the current traffic signal to the chosen action are shown in Table 12. Note the addition of the North-South Yellow (NSY) and East-West Yellow (EWY) and All Red (R) traffic signal configurations, which cannot be chosen explicitly as actions, but are part of some traffic signal transition sequences. The yellow and red phases are necessary for safety reasons, as they slow down and stop traffic so that succeeding green phases may be enacted.

Reward

The final element of reinforcement learning, after the agent has observed the state of the environment s_t, chosen an action a_t, and performed it, is receiving the reward. The reward is one element that differentiates reinforcement learning from other types of machine learning; developing a state-action policy which maximizes cumulative long-term reward is what the agent seeks. Compared to other types of machine learning, in which correct actions are given by instruction, reinforcement learning has the agent evaluate actions by interacting with the environment. How to select the appropriate reward for a given task is an unanswered problem in traditional reinforcement learning[1]. It would be desirable if the agent could choose its own reward, instead of requiring an expert to define it, and is therefore a goal of many active researchers.

In the context of traffic signal control, various rewards have been proposed, such as change in number of queued vehicles, change in cumulative vehicle delay and change in vehicle throughput. The reward $r_{t+1} \in \mathbb{R}$ is a consequence of enacting a selected action from a specific state. In this research, we define the reward as change in cumulative vehicle delay between actions. This allows for the reward to be positive or negative, meaning the agent can be punished ($r_{t+1} < 0$ for increase in delay) or rewarded ($r_{t+1} > 0$ for decrease in delay). The use of the subscript $t+1$ is intentional, to

[1]See inverse reinforcement or apprenticeship learning.

emphasize the temporal relationship between taking action a_t in state s_t, as the reward succeeds these two events. In addition to receiving a reward from the environment, the agent has the opportunity to observe the new state of the environment s_{t+1}, which was influenced by its most recent action. With this new state observation, a new action can be chosen and subsequently a new reward received. This cycle can be continued indefinitely or stopped according to some criteria, depending on the reinforcement learning task at hand.

Agent

In reinforcement learning, the agent is the entity that learns by interacting with the environment. We model the agent controlling the traffic signals as a deep convolutional Q-network [45]. Artificial neural networks are mathematical functions inspired by biological neural networks (i.e., brains) that are appealing for their function approximation capabilities. Many problems in machine learning can suffer from the curse of dimensionality, which is when the dimensionality of the data increases, the training and computational resources required grow exponentially. Artificial neural networks have the capability to generalize from what they have learned, weakening the problems posed by the curse of dimensionality. Convolutional neural networks are a variant of artificial neural networks inspired by biological research that emulate the architecture of the animal visual cortex [138][139], making them adept at perception tasks.

Most artificial neural networks require data pre-processing, where features of the data are determined by experts. Features are measurable aspects of the data deemed important to the present machine learning task. Expert-crafted features require assumptions to be made about the data that may or may not be true. In the context of traffic signal control, examples of expert-crafted features are queue length or average vehicle flow. These features are abstractions of the individual vehicles behavior that have been extracted and deemed important by experts for solving the traffic signal control problem using reinforcement learning. However, because they are abstractions, we argue important information is lost and the potential for learning is diminished. If only queue length is used, this assumes all vehicles not in a queue are irrelevant to developing an optimal traffic signal control policy - a spurious claim. Similarly, average flow is a historical metric calculated over some time interval, yielding a very coarse approximation of the current traffic state that ignores and abstracts away useful information. Convolutional neural networks are advantageous because they develop their own features from the data. The DTSE is proposed because it is a lesser abstraction of the traffic state than queue length or average flow and the convolutional neural network can take advantage of its information rich nature.

The depth of a deep neural network refers to the fact that there is more than one hidden computational layer of neurons. Additional layers in a network allow it to develop features of features, transforming low-level features of the data to high-level ones, potentially increasing network performance. The combination of the DTSE and the convolutional neural network allow for the creation of a truly adaptive traffic signal controller.

The DQTSCA's architecture is first two identical networks receiving different inputs. Each network receives a different input vector from the DTSE - one the real-valued vector, the other the Boolean-valued vector. The first layer of each network is a convolutional layer with 16 filters of size 4x4 applied with stride 2 using rectifier nonlinear activation functions. The second layer of each network is a convolutional layer with 32 filters of size 2x2 applied using rectifier nonlinear activation functions. The outputs of these two networks and the vector P, representing the current traffic signal phase, are combined and used as input to two fully-connected layers of 128 and then 64 neurons with rectifier nonlinear activation functions. The output layer is $|A|$ (i.e., four) neurons with linear activation functions.

The reinforcement learning algorithm used in this research is Q-Learning [38], which is used to develop an optimal action-selection policy. The optimal policy is achieved by using the convolutional neural network to approximate the action-value function. The action-value function maps states to action utilities (i.e., what is the value of each action from a given state). Values represent long-term reward. If an action has a high value, enacting it means reaping future reward, although potentially not immediate reward. We define the deep convolutional neural network as the action-value function $\eta : X \mapsto Y$, where $X \in S$ and $Y \in \mathbb{R}^{|A|}$, with Y representing the action-values $\forall A$. At time t, the input x_t to the network is $x_t = s_t$. The output y_t is a vector containing all the action-values, with y_{t,a_t} denoting the action-value of a_t. After the action-value function has been sufficiently learned, the optimal policy can be determined by selecting the action with the highest value given s_t. The basis of Q-learning is the value iteration update (Q-update), defined in (22).

$$\begin{aligned} Q(s_t, a_t) = Q(s_t, a_t) + \\ \alpha\Big(r_{t+1} + \gamma \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)\Big) \end{aligned} \tag{22}$$

Where the learning rate α controls the degree to which new action-value estimates are weighted against old estimates and the discount factor γ determines how immediate rewards are weighted against future rewards. Both the learning rate and the discount factor are parameters of Q-learning and are $\gamma, \alpha \in [0, 1]$. To use the Q-update to train the deep convolutional network, a variation of

(22) is used, defined in (23).

$$\begin{aligned} x_t &= s_t \\ x_{t+1} &= s_{t+1} \\ y_t &= \eta(x_t) \\ y_{t,a_t} &= r_{t+1} + \gamma \max\Big(\eta(x_{t+1})\Big) \end{aligned} \tag{23}$$

After (23) has been computed, the deep convolutional network can be trained and the weights θ updated using gradient descent with x_t as the network input and y_t as the output. We use the RMSprop [93] gradient descent algorithm with an α of 0.00025 and a γ of 0.95 to train the network. Once the deep convolutional neural network has sufficiently approximated the action-value function, optimal control is achieved by selecting the action with the highest value given the current state.

A major problem in any reinforcement learning task is the action-selection policy while learning; whether to take exploratory action and potentially learn more, or to take exploitative action and attempt to reap the most reward given what has been learned so far. The explore-exploit tradeoff is an active area of research in reinforcement learning with many proposed solutions. We implement the simple, yet effective, decreasing ϵ-greedy exploration policy, which selects a random action (explore) with a probability ϵ and selects the action with the highest value (exploit) with a probability 1-ϵ. The value of ϵ decreases as training epochs progress according to (24).

$$\epsilon_n = 1.0 - \frac{n}{N} \tag{24}$$

Where n is the current training epoch and N is the total number of training epochs Initially, $\epsilon =$ 1.0, meaning the agent exclusively explores, however, as training progresses, the agent increasingly exploits what it has learned, until it exclusively exploits.

Experimental Setup and Training

All experiments were conducted using the traffic microsimulator SUMO v0.22 [4]. SUMO provides an application programming interface (API) in the Python programming language, by which custom functionality can be implemented in the traffic simulation. We used the SUMO Python API and custom code to implement the DQTSCA. The artificial neural network was implemented using Keras [141] and Theano [142] Python libraries. Additional optimized functionality was provided by NumPy and SciPy [96] libraries. For the DTSE parameters, we define l as 75 m and c as 5 m. We train for 1 600 training epochs, where each epoch is 1.25 hours of simulated traffic. The simulations were executed on a desktop computer with a 3.40 GHz i7-2600 CPU, 8GB of RAM running Ubuntu 14.04. The length of the agent's actions (i.e., NSG, EWG, NSLG, EWLG) are two seconds and the transition phases (i.e., R, NSY, EWY) are five seconds.

Table 13: Vehicle Generation Distributions by Flow Rate

Flow Rate (Vehicles/Hour)	**Distribution**	**Parameters** (α, β)
0-150	Inverse Weibull [143]	(0.65, 5.8)
250-450	Burr [144]	(1.4, 5.9)

The intersection geometry is four lanes approaching the intersection from the compass directions (i.e., North, South, East and West) connected to four outgoing lanes from the intersection. The traffic movements for each approach are as follows: the inner lane is left turn only, the two middle lanes are through lanes and the outer lane is through and right turning. All lanes are 750 meters in length, from the vehicle origin to the intersection stop line.

The method by which vehicles are generated and released into the network greatly influences the quality of any traffic simulation. The most popular vehicle generation method is to randomly sample from a probability distribution numbers that represent vehicle headway times, or the time interval between vehicles. This research does not break from this method entirely, however we strive to implement a nuanced version which better models real-world traffic. Empirical research has shown that different vehicle flow rates are suitably approximated by different probability distributions [143][144]. Instead of using a negative exponential distribution for all flow rates and modifying its rate parameter, we use different distributions for different flow rates, shown in Table 13. The Inverse Weibull distribution is used for generating left and right turning traffic and the Burr distribution is used for generating through traffic.

The agent is trained using a biologically inspired process known as experience replay [145][146][147]. Instead of training after every individual state, action, reward, state sequence, the agent stores the experience, defined $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$, in an experience memory M for periodic, randomized batch training. The training pseudocode is presented in Algorithm 1 and 2. This research takes advantage of the multithreading capabilities of modern computers, running multiple traffic simulations in parallel. Each thread is running Algorithm 2 and generating different experiences for use in the experience replay.

Algorithm 1: Deep reinforcement learning traffic signal control agent experience replay

```
Initialize neural network agent $\eta$ with random weights $\theta$ on main agent
Copy main agent weights $\theta$ to all thread agents
For epoch=1 to $N$ do
  Copy main agent weights $\theta$ to all thread agents
  In parallel run Algorithm 2 on threads
    While all threads not finished do
      If buffer == batch_size do
        Append buffer to $M$, clear buffer
        Randomly sample batch_size experiences,
        from $M$
        Batch train main agent using (23)
  If epoch mod(exp_refill) == 0 do
    Clear $M$
    While len($M$) < min_size do
      In parallel run Algorithm 2 on threads
        Append buffer to $M$
```

Algorithm 2: Thread Traffic Simulation

```
For $t$=1 to sim_len do
  Observe DTSE, $s_t$
  Select random action $a_t$ with probability $\epsilon$,
  else select $a_t = \max\eta(s_t)$
  Implement selected $a_t$, increment simulation,
  observe reward $r_{t+1}$ and $s_{t+1}$,
  If len($M$) == max_size do
    delete $M[0]$
  Append $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$ to buffer
```

The *buffer* in Algorithm 1 and 2 temporarily stores the most recent experiences until it reaches *batch_size*, at which point it is appended to M and cleared. The *max_size* and *min_size* are respective upper and lower limits of M. Training can only begin after M has at least *min_size* experiences. The oldest experience is deleted when M has *max_size* elements. In our research, we use a *batch_size* of 16, a *max_size* of 500 000 and a *min_size* of 50 000. We found learning improved when we periodically cleared M and refilled it with new experiences every *exp_refill* epochs, where *exp_refill* is 200. The *sim_len* is 4 500 timesteps.

We developed a shallow neural network TSCA to compare against our proposed DQTSCA. The shallow traffic signal control agent (STSCA) has one hidden layer with 64 neurons using the sigmoid activation function and four neurons with linear activation functions for its output layer. The state space of the STSCA is two vectors, the first containing elements that represent the number of queued vehicles at each intersection approach (i.e., North, South, East and West) and the second the current traffic signal phase vector P. The action space and reward are the same as the DQTSCA. The STSCA is trained using the same number of epochs, action selection policy and gradient descent algorithm as the DQTSCA. However, the traditional agent does not use experience replay, it trains using (23) after every state, action, reward, state sequence.

Results and Discussion

The performance of the proposed DQTSCA was assessed with respect to common traffic metrics: throughput, queue length, travel time and cumulative delay. The performance of the agent with respect to the traffic metrics while learning can be seen in Figures 24, 25, 26, and 27. The agent's performance with respect to achieving reward while learning can be seen in Fig. 28. The agent's action-reward performance during one epoch is also shown, in Fig. 29 exclusively exploring initially in training and exclusively exploiting after training in Fig. 30. Initially, while learning, the agent is predominantly exploring (i.e., taking random actions), attempting to learn the action-value function. While exploring, the agent's performance with respect to the traffic metrics exhibits high variance and it achieves negative reward (i.e., punishment). Because of the agent's actions, many vehicles are queued, unnecessarily delayed and the overall throughput is low. As the epochs progress, the agent has better learned the action-value function and can begin selecting exploitative actions instead of exploratory ones. The decreasing exploration rate is reflected in improved performance with respect to all four metrics and higher reward - evidence the agent has learned. Not only does the DQTSCA perform better as training progresses, convergent behavior emerges, as the variance in its performance decreases.

The agent's behavioral change before and after training can be seen in Figures 29 and 30. These figures show rewards as a consequence of each action taken within one epoch (i.e., 1.25 hours of simulated traffic). In Fig. 29, the agent is taking random actions with no consideration for reward, reflected as unstable and divergent rewards. The key observation is that the rewards, positive or negative, increase in magnitude as the epoch progresses because the agent is taking random, exploratory actions. Comparing Fig. 30 with Fig. 29, it is apparent the agent is acting differently.

Table 14: STSCA and DQTSCA Traffic Metrics

Traffic Metric (μ, σ, $n = 100$)	**STSCA**	**DQTSCA**
Throughput (Vehicles)	(2 452, 257)	(2 456, 248)
Queue (Vehicles)	(33, 23)	(13, 9)
Travel Time (s)	(197, 107)	(157, 49)
Cumulative Delay (s)	(4 085, 5 289)	(719, 1 048)

In Fig. 30, the rewards are an order of magnitude smaller and stable as the epoch progresses, with no divergence near the end of the epoch as in Fig. 29, because the agent is enacting an exploitative policy. These observations are supported quantitatively, computing the average and standard deviation (μ, σ) of the reward for each epoch, Fig. 29 has $(-347, 2\,220)$ and Fig. 30 has $(-0.485, 59.6)$.

A comparison of the proposed DQTSCA with the STSCA can be seen in Table 14. The data in Table 14 is computed from the last 100 training epochs of each agent, where the agents are taking exploitative action >93% of the time. Although four traffic metrics are considered, cumulative delay is the only metric the agent can tangibly interact with, as change in cumulative delay is its reward function. The DQTSCA achieves an 82% reduction in the average cumulative delay compared to the STSCA. The difference in this key metric provides evidence that the DQTSCA has learned a control policy superior to the STSCA. Comparing the other traffic metrics, there is no difference in the throughput, but the DQTSCA reduces the average queue length by 66% and the average travel time by 20% compared to the STSCA. The DQTSCA outperforms the STSCA in three of the four metrics, due to the use of the DTSE and its deep architecture. Future work should investigate a throughput reward function and compare the two agents performance, as it is the only metric where the agents perform equally.

A limitation of this research is we did not consider how fair the agent's policy is. A fair traffic signal controller would ensure all vehicles are given equal priority to traverse the intersection, however this may be in conflict with optimizing certain traffic metrics, such as minimization of delay or maximization of throughput. A balance between fairness and optimality could be achieved with the appropriate reward function, which should be the subject of future research.

Conclusion

We proposed, developed and tested a DQTSCA in a traffic microsimulator. The results show deep learning can be applied to traffic signal control with improved performance compared to traditional methods.

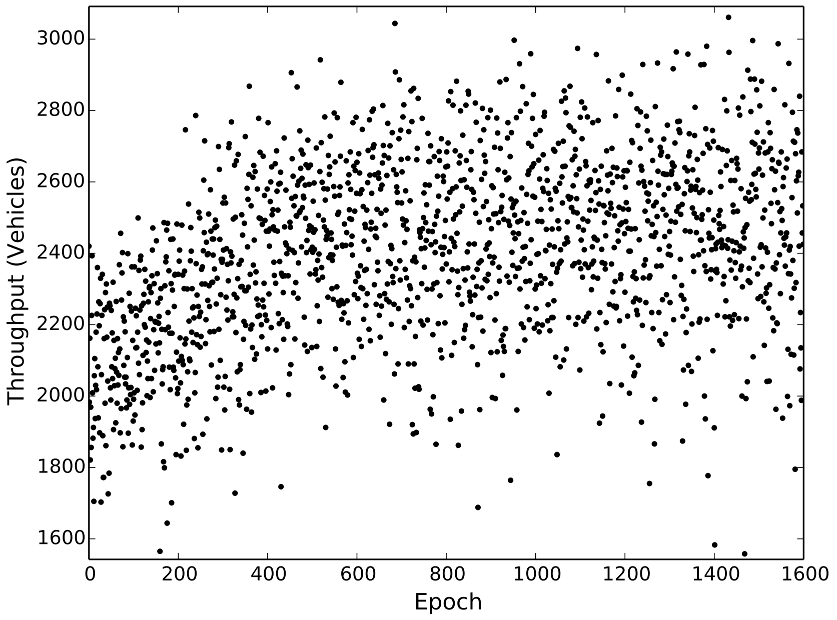


Figure 24: Intersection throughput while training.

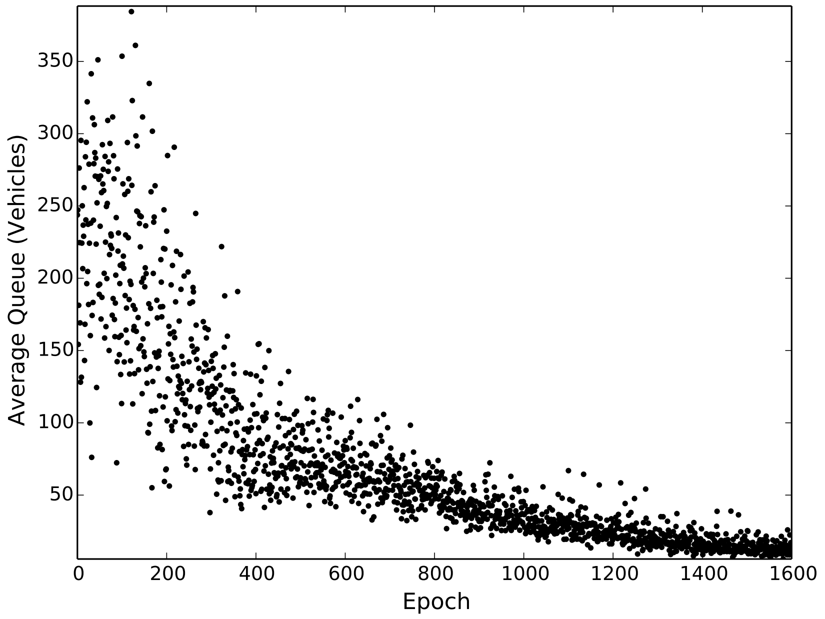


Figure 25: Average intersection queue while training.

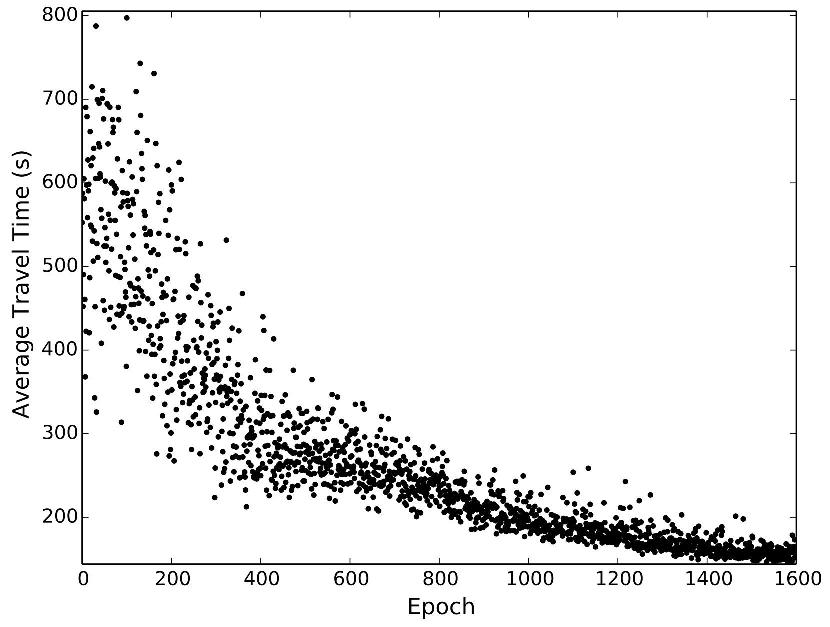


Figure 26: Average travel time of vehicles while training.

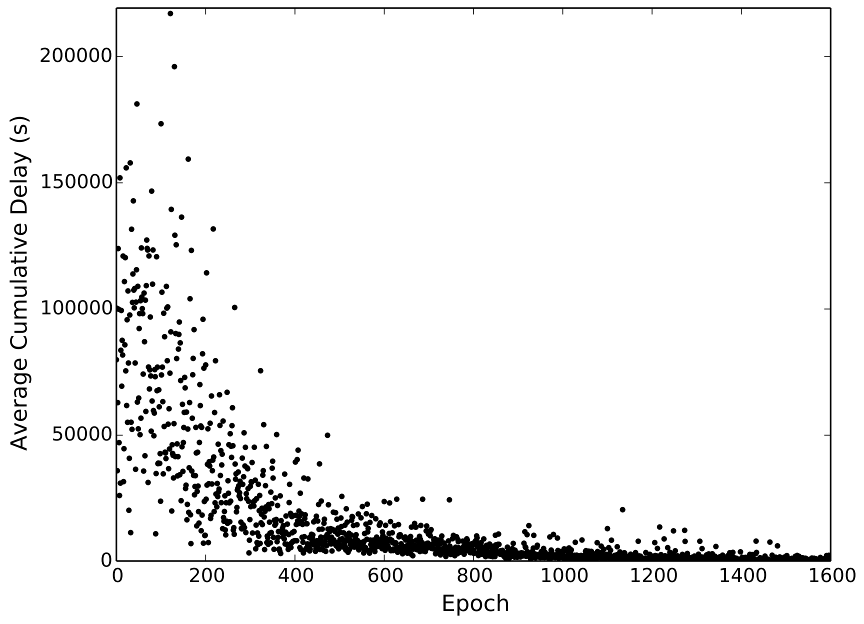


Figure 27: Average cumulative delay of vehicles while training.

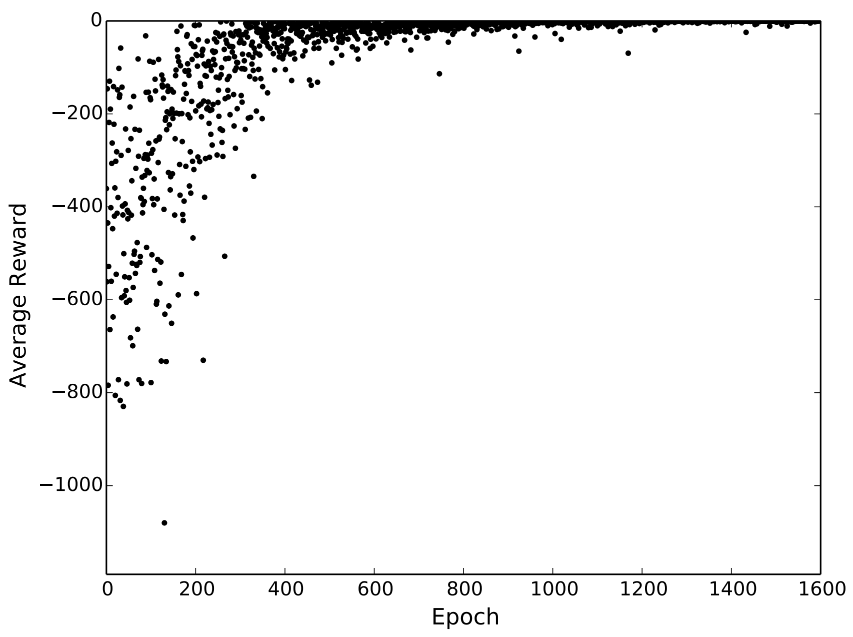


Figure 28: Average reward of DQTSCA while training.

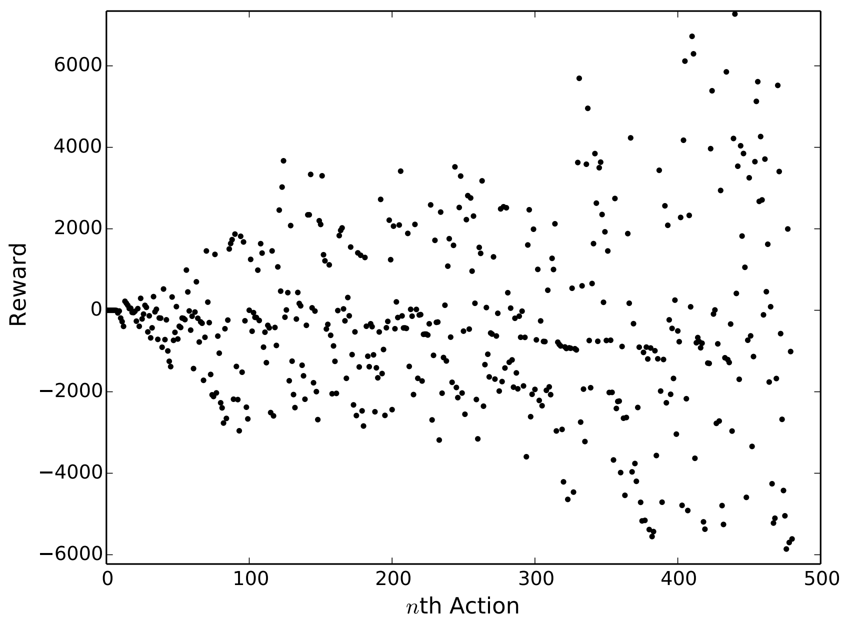


Figure 29: Reward of DQTSCA in an epoch while taking only exploratory action early in training.

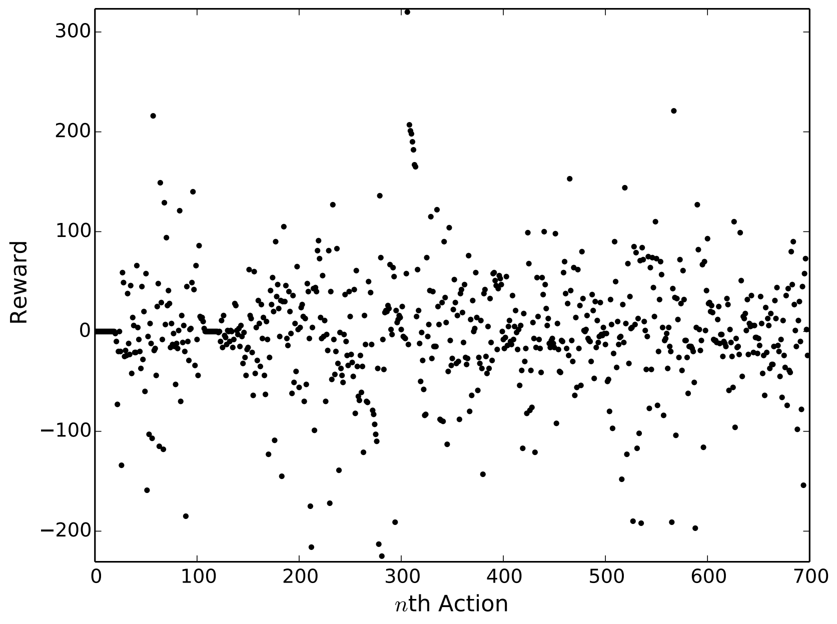


Figure 30: Reward of DQTSCA in an epoch while taking only exploitative action after training completed.

Future work in this area can extend the agent's control to all traffic signals, including the yellow and red phases. Currently, the agent has no capability to control the yellow or red phases, they only exist in the transitional sequences between agent actions. However, it is obvious that situations exist where a dynamic yellow or red phase is desirable. For instance, a vehicle does not decelerate to a yellow phase and accelerates through the intersection; extending the yellow phase until all vehicles have either cleared the intersection or are otherwise decelerating is prudent. We hypothesize the means to accomplish this with a TSCA trained through reinforcement learning would be to change the reward function so that it yielded high reward when vehicles decelerate and are not traversing the intersection.

The use of the DTSE may also allow for training a TSCA to control intersections of various lane configurations without retraining. For example, first train the agent using the DTSE on a four lane intersection. Then it could be used to control a two lane intersection by setting all of the elements of the two 'missing' lanes to zero. The DTSE may allow for a widely applicable TSCA without retraining for different intersection geometries.

Additional research should also increase the complexity of the traffic network and apply the DTSE and deep architecture to multiple TSCA. These ideas will be explored in future endeavors by the authors.

APPENDIX B: RL-TSC State Space Evaluation

This manuscript was submitted, accepted and presented at the 9th International Conference on Ambient Systems, Networks and Technologies (ANT-2018) conference in Porto, Porugal under the following reference:

W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," *Procedia computer science*, vol. 130, pp. 26–33, 2018. `https://doi.org/10.1016/j.procs.2018.04.008`

The co-author's contributions to the above work include:

- Financial and technical supervision of the study presented in this work.
- Manuscript editing.

Abstract

Reinforcement learning has shown potential for developing effective adaptive traffic signal controllers to reduce traffic congestion and improve mobility. Despite many successful research studies, few of these ideas have been implemented in practice. There remains uncertainty about what the requirements are in terms of data and sensors to actualize reinforcement learning traffic signal control. We seek to understand the data requirements and the performance differences in different state representations for reinforcement learning traffic signal control. We model three state representations, from low to high-resolution, and compare their performance using the asynchronous advantage actor-critic algorithm with neural network function approximation in simulation. Results show that low-resolution state representations (e.g., occupancy and average speed) perform almost identically to high-resolution state representations (e.g., individual vehicle position and speed). These results indicate implementing reinforcement learning traffic signal controllers may be possible with conventional sensors, such as loop detectors, and do not require sophisticated sensors, such as cameras or radar.

Introduction

Vehicle congestion is a major problem in cities across the world. Developing additional infrastructure is expensive and a protracted process which can exacerbate the problem until completed. Instead of adding more infrastructure, another solution is to optimize currently available infrastructure. Intersection traffic signal controllers (TSC) are ubiquitous in modern road infrastructure and their functionality greatly impacts all users. Many research studies have proposed improvements to TSC, broadly in an attempt to make them adaptive to current traffic conditions. Reinforcement learning has been shown to be effective in developing adaptive TSC with many research studies detailing promising results. Despite the encouraging research, few reinforcement learning adaptive TSC have been deployed in the field. One inhibiting factor is the resources required; to observe the traffic state, reinforcement learning TSC often require high-resolution data beyond the detection capability of traditional sensors (i.e., loop detectors). This research focuses on the potential state definitions of reinforcement learning TSC and ascertaining the performance differences between them. We seek to answer, can a reinforcement learning TSC function using low-resolution data from traditional sensors such loop detectors? Or is high-resolution data from sophisticated sensors (e.g., cameras, radar) required? Answering this question will help individuals interested in deploying reinforcement learning TSC in the field, as they will be aware of the requirements and potential outcomes. We use the traffic microsimulator SUMO [4] and the asynchronous advantage actor-critic (A3C) algorithm

[46] to train and evaluate multiple adaptive TSC with different resolution state representations.

Literature Review

Many research studies have recognized and displayed reinforcement learning's capability for providing a solution to TSC. Early research provided proof-of-concept for reinforcement learning in TSC [57, 59, 82, 60]. Later research applied reinforcement learning methods to more realistic and complex traffic models [148, 64, 84, 98, 99, 100]. Developments in machine learning have yielded deep reinforcement learning techniques [46, 45, 124] which have subsequently been applied for TSC [66, 68, 67, 69, 76].

Considering the aforementioned research and the extensive reinforcement learning TSC reviews [86, 87, 97], we identify numerous possible state representations: vehicle density, flow, queue, location, speed along with the current traffic phase, cycle length and red time. These state representations form a resolution spectrum of the current traffic state, from coarse (e.g., flow) to fine (e.g., individual vehicle position and speed). We consider state representation across the resolution spectrum, requiring different sensors, and compare their performance. The results can guide individuals interested in practical implementation.

Model

Reinforcement Learning

Reinforcement learning is a type of machine learning for solving sequential decision-making problems [36]. A reinforcement learning agent learns a policy $\pi(s) = a$, mapping from states s to actions a, to achieve a goal in an environment under uncertainty. Through repeated environment interactions, a reinforcement learning agent strives to develop an optimal policy π^*, which maximizes the sum of future discounted ($\gamma \in (0, 1]$) rewards, defined as the return G_t in Equation 25:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{25}$$

The agent interacts with the environment in repeating sequences of, at time t, observing the environment state s_t, taking action a_t, receiving reward r_t and entering a new state s_{t+1}. Over time, the agent learns what actions in what states maximize long-term reward, also known as value. Rewards quantitatively represent how successful the agent's policy is achieving its mandated goal.

The A3C algorithm is used to develop parameterized θ policy $\pi(a|s; \theta)$ (Equation 26) and value $V^\pi(s; \theta)$ functions (Equation 27). The agent develops a value function (critic), which estimates the expected return from a given state, which is used to improve the policy (actor).

$$\pi(a|s;\theta) \quad = \quad \Pr[a_t = a|s_t = s;\theta] \tag{26}$$

$$V^{\pi}(s;\theta) \quad = \quad \mathbb{E}[G_t|s_t = s;\theta] \tag{27}$$

The parameters are used for neural network function approximation, defining the weights between neurons.

Environment

The environment used to train the reinforcement learning adaptive TSC is the traffic microsimulator SUMO[4]. The network geometry is an isolated intersection with four origin-destination zones in each compass direction; North (N), South (S), East (E) and West (W). Each origin-destination zone is connected to the intersection with eight lanes, four incoming and four outgoing. The turning movements for incoming lanes to the intersection are; the right lane allows right turn and through movements, the middle two lanes allow through movements and the left lane allows left turns.

We simulate a peak or rush hour traffic demand scenario for training. The traffic is generated stochastically using a negative exponential distribution with a rate parameter λ. To add further stochasticity, the rate parameter λ is sampled from a normal distribution $\mathcal{N}(\lambda, \frac{\lambda}{10})$, as seen in Figure 31.

State

At time t the reinforcement learning agent observes the state of the environment s_t. The agent's behaviour and ability to learn is greatly influenced by the state and its definition. Three state spaces are defined for reinforcement learning TSC with different resolutions of the environment. All state representations include the most recent traffic phase encoded as a one-hot vector (i.e. a phase from the set of all traffic phases P) and the time spent in that phase. One-hot vectors are used to represent categorical variables. For K categories, a $K \times K$ identity matrix represents all the categories, with each row representing a different category.

Occupancy and Speed

The lowest resolution state space is defined using occupancy and average speed. Loop detectors are the most common sensors at intersections and can be used to collect coarse traffic statistics, such as occupancy and average speed for each lane. We model each incoming lane with two loop detectors, one at the stop line and the other setback 50 m from the stop line. The occupancy and average

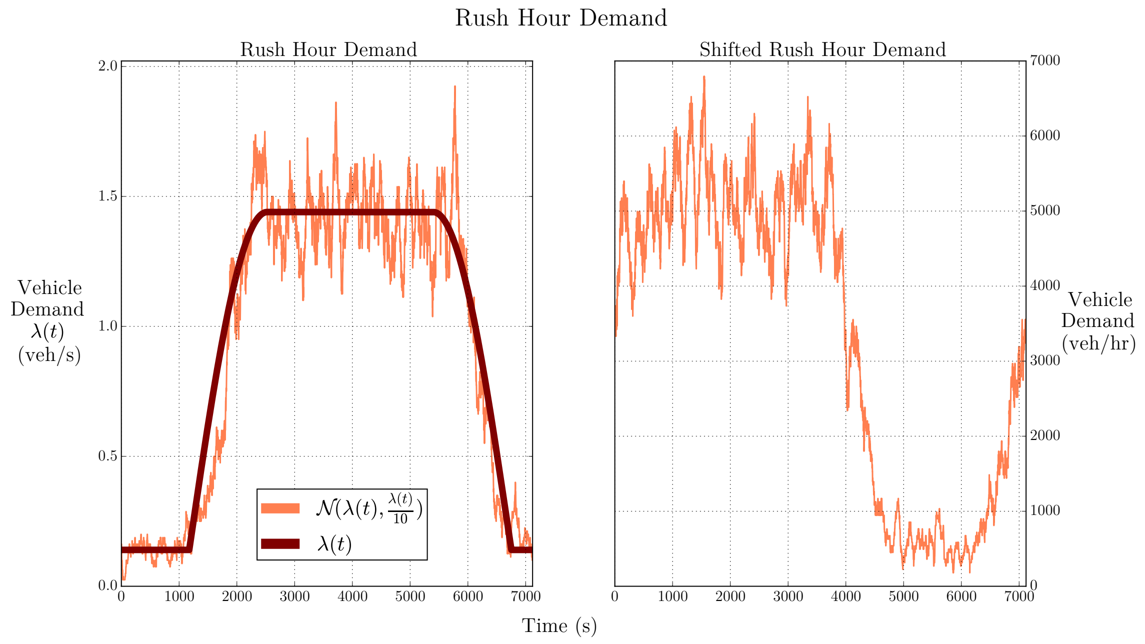


Figure 31: Rush hour demand traffic scenario used for training (right) and testing (left).

speed (normalized by the speed limit) are computed from the the previous 10 s interval. Given m incoming lanes, the loop-TSC state is $s_t \in \mathbb{R}^{(2m+|P|+1)}$.

Queue and Density

A higher resolution state space is defined using vehicle density and queue. Loop detectors are inadequate to collect queue and density data reliably, more sophisticated sensors are required (i.e., video cameras, radar). Assuming a jam density k_j, V_l represents the set of vehicles on lane l and $V_{l,q}$ the set of queued vehicles on lane l, we define vehicle lane density $\frac{V_l}{k_j}$ and vehicle lane queue $\frac{V_{l,q}}{k_j}$. Given m incoming lanes, we denote the queue-TSC state $s_t \in \mathbb{R}^{(2m+|P|+1)}$.

Discrete Cell Encoding

The highest resolution state space discretizes each incoming lane into cells of a fixed length $c = 2.5$ m, termed the discrete traffic state encoding (dtse). Cells are binary encoded, 1 represents the presence of a vehicle and 0 represents the absence of a vehicle. Sophisticated sensors (i.e., video cameras, radar) would also be required to collect this state data. Given an incoming lane length $L = 135$ m, the dtse-TSC state is $s_t \in \mathbb{R}^{(\frac{L}{c}m+|P|+1)t}$, where we use a history of the $t = 2$ most recent states. This state definition was first proposed using SARSA reinforcement learning [57] and has since been utilized in deep reinforcement learning TSC [66, 68, 69].

Actions

After observing state s_t, the agent chooses an action $a_t \in A$. The actions available are the green traffic phases, denoted in this research by a pair of compass directions and set of movement priorities (i.e., G represents protected through movements and permissive left turn movements and LG represents protected left turn movements and prohibited through movements). For example, the action NSG represents North-South protected through movements, permissive left turn movements and prohibits all East-West movements. Actions in a sequence may require yellow change and red clearance phases, with additional action/phase information detailed in Table 15. The set of all possible actions is denoted $A = \{\text{NSG, EWG, NSLG, EWLG}\}$. All actions $a_t \in A$ have a duration of 10 s and yellow and red phases have a duration of 4 s. When no vehicles are present at the intersection (i.e., $\forall l, V_l =)$), all movements are prohibited with the red clearance phase.

The agent's traffic signal control policy is acyclic, unconstrained and *ad-hoc*. We argue imposing a cycle in reinforcement learning TSC is presumptuous. If a cycle is optimal, the agent will develop such a policy. There are no maximum times for each phase and the agent chooses the next action/phase without limitation.

Table 15: Traffic Signal Phase Information.

			Turning Movements		
Action	NEMA Phases	Compass Directions	Left	Through	Right
NSG	2, 6	North, South	Permissive	Protected	Permissive
NSLG	1, 5	North, South	Protected	Prohibited	Permissive
EWG	4, 8	East, West	Permissive	Protected	Permissive
EWLG	3, 7	East, West	Protected	Prohibited	Permissive

Reward

After observing state s_t and taking action a_t, the agent receives a scalar reward $r_t \in \mathbb{R}$ from the environment. The reward is feedback for how 'good' action a_t was in s_t. Many rewards have been proposed for reinforcement learning TSC (e.g., functions of throughput, queue, delay). We define reward in Equation 28 as change in cumulative delay:

$$r_t = D_{a_t} - D_{a_{t+1}} \tag{28}$$

Where $D_{a_t}, D_{a_{t+1}}$ represent the cumulative delay at the intersection when action a_t and a_{t+1} are taken. Cumulative vehicle delay D at time t is defined in Equation 29:

$$D_t = \sum_{v \in V_t} d_t^v \tag{29}$$

Where V_t is the set of vehicles on incoming lanes in the simulation at time t and d_t^v is the delay of vehicle v at time t.

Agent

The agent is the entity, through repeated interaction with the environment, that implements and improves the policy π. In this research, the agent chooses the next green traffic phase. We model the agent as an artificial neural network and train it using the A3C algorithm.

An artificial neural network is chosen for its flexible function approximation capabilities. The architecture of the neural network is an input layer and then a fully connected hidden layer with rectified linear (ReLu) activation functions followed by another fully connected hidden layer with ReLu activation functions. The output layer has $|A| = 4$ neurons with softmax activation functions which output the action probabilities representing the policy. The number of neurons in each hidden layer for each state representation is equal to the cardinality of the input state; loop-TSC and queue-TSC hidden layers have 42 neurons and dtse-TSC hidden layers have 1780 neurons.

The A3C algorithm simulates multiple actor-critic agents in parallel, each with their own environment. Using a local parameter set θ', each agent computes an advantage $Adv = G_t - V(s;\theta')$ from experience sequences of length $t_{max} = 32$. The advantage is a measure of the difference between the actual and expected performance of the policy. The advantage is used to compute parameter gradients $d\theta$ which are asynchronously applied to a global parameter set θ. Each agent periodically copies the global parameters θ as their local parameters θ'. The rewards are standardized before computing the return (i.e., $r_t \leftarrow \frac{r_t - \mu_r}{\sigma_r}$) and the gradient of the policy entropy $\beta \nabla_{\theta'} H(\pi(s_i;\theta'))$ is added to the parameter update for improved learning.

Algorithm 1: Asynchronous Advantage Actor-Critic training pseudocode [46]

```
$n \leftarrow 0$, Initialize $\theta, \theta^{'}$
While $n < N$
  $t \leftarrow 0$, Generate sim with randomly shifted demand
  While $t < T$
    Reset parameter gradients $\mathrm{d}\theta \leftarrow 0$
    Set thread parameters to global $\theta' = \theta$
    $t_{start} = t$
    Observe state $s_t$
    While $s_t \neq s_{terminal}$ and $t - t_{start} \neq t_{max}$
      Perform action $a_t$ from $\pi(a_t|s_t;\theta')$
      Receive reward $r_t$ and observe new state $s_{t+1}$
      $s_t = s_{t+1}$
      $t \leftarrow t + 1$
    If $s_t == s_{terminal}$
      $G_t = 0$
    Else
      $G_t = V(s_t;\theta')$
    For $i \in \{t-1, t-2, ..., t_{start}\}$
      $G_t \leftarrow r_i + \gamma G$
      $Adv = G_t - V(s_i;\theta')$
      $\epsilon = \beta \nabla_{\theta'} H(\pi(s_i;\theta'))$
      Collect policy gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \nabla_{\theta'} \log_e \pi(a_i|s_i;\theta')(Adv) + \epsilon$
      Collect value gradients $\mathrm{d}\theta \leftarrow \mathrm{d}\theta + \frac{\partial (Adv)^2}{\partial \theta'}$
    Asynchronously update $\theta$ with $\mathrm{d}\theta$
  $n \leftarrow n + 1$
```

Experiments

We subject each different state representation reinforcement learning TSC to $N = 1\,000$ rush hour demand scenarios, $T = 7\,200$ simulation steps in duration, for training. As a form of data augmentation and to prevent overfitting, the rush hour demand scenario is randomly shifted in time for each simulation, displayed in Figure 8. Training is conducted using a consumer i7 CPU with 8 parallel, asynchronous actor-critics, each with their own environment. Training for each state representation was 3-6 hours wall clock time.

For comparison after training, each different state reinforcement learning TSC is subjected to 100 rush hour demand scenarios without data augmentation to ensure parity during testing. Throughput, delay and queue statistics are collected during testing simulations.

The traffic microsimulator SUMO[4] is used for all simulations. Tensorflow [94], SciPy [96] and additional Python libraries [149] are used for implementing the neural networks and reinforcement learning.

As a baseline for comparison, we model an Actuated TSC which uses loop detectors to modulate the green phase lengths. The Actuated TSC is cyclic; each phase has a minimum green time of 10 s, after which a gap-out timer begins decrementing from 5 s. If a vehicle is detected in a lane with a protected movement under the current phase the gap-out timer is reset to 5 s, up to a maximum of 40 s.

Results

Testing results are displayed in Table 16 and visually in Figure 32. All reinforcement learning TSC achieve superior performance in reducing delay and queue lengths compared to the Actuated TSC. This result is not surprising, as the reinforcement learning TSC have greater flexibility in action selection compared to the Actuated TSC, which has to implement phases in a cycle.

Observing the traffic metrics collected, there appears to be little to no difference between the different state representations. There is no difference in throughput or queue however the loop-TSC exhibits the highest delay compared to the queue-TSC and dtse-TSC. This result is surprising to the authors, specifically how little difference there is between the different state representations considering the spectrum of data resolution modelled. Results from research in other domains using deep reinforcement learning suggest that high-resolution state data can yield superior performance to low-resolution state data. In this research, we find very little difference in performance despite the disparity in data resolution between the different state representations.

Conclusion

We modelled adaptive TSC using the A3C reinforcement learning algorithm with various state representations to determine any differences in performance. Results show that data collected from traditional and ubiquitous sensors such as loop detectors are sufficient for reinforcement learning

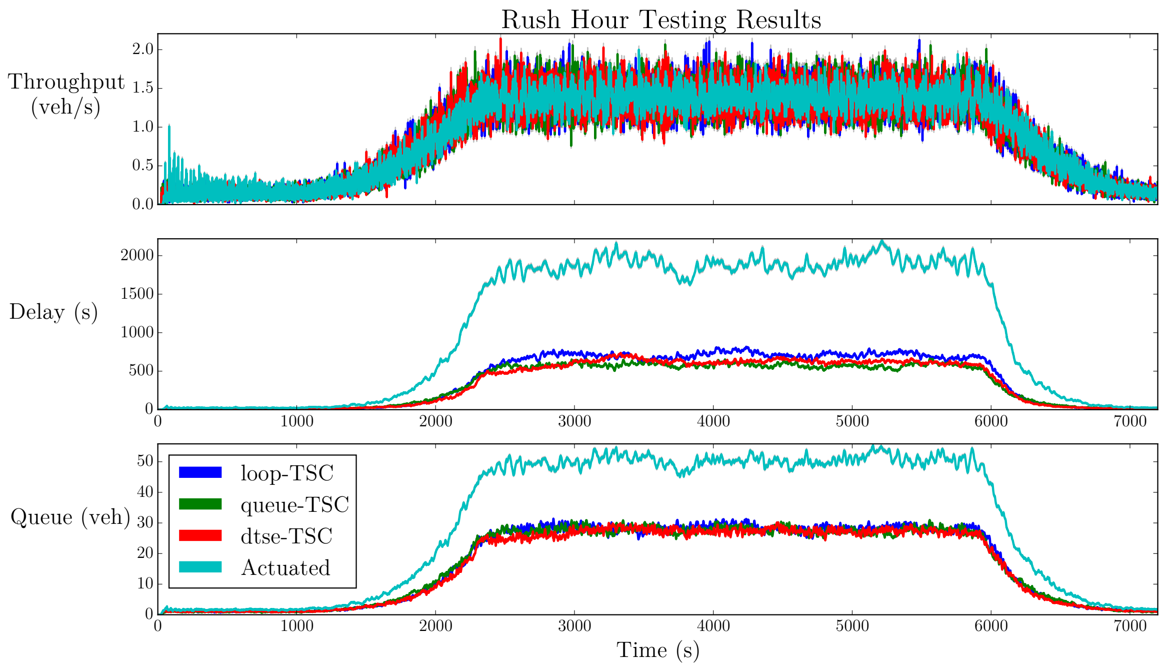


Figure 32: Testing results comparing state definitions.

adaptive TSC. Under the model devised in this research, high-resolution state representations requiring sophisticated sensors offer improvements only by reducing delay by approximately 10%, with no difference in queue or throughput.

A potential explanation for the lack of performance disparity is that high-resolution state data may only yield benefits with sufficiently complex function approximators. The fully connected neural network architecture may be too simple compared to convolutional, long short-term memory layers or residual connections. These architectures may be required to leverage high-resolution state data and develop the rich representations necessary for improved performance. Given the lack of substantial performance between state representations, this may be evidence that sophisticated machine learning methods, such as reinforcement learning, are unnecessary for traffic signal control. Other, less resource intensive machine learning methods such as decision trees or regression methods may be more appropriate. These methods may offer similar performance while also being much more comprehensible, as opposed to the black box nature of a neural network. These ideas should be should be considered in future research.

Table 16: Cumulative statistics testing results comparing state definitions.

	$(\hat{\mu}, \hat{\sigma})$		
TSC	Throughput (veh/sim)	Delay (s/sim)	Queue (veh/sim)
loop-TSC	$(6\,609, 86)$	$(2.8 \times 10^6, 4.6 \times 10^5)$	$(1.2 \times 10^5, 0.8 \times 10^4)$
queue-TSC	$(6\,612, 87)$	$(2.3 \times 10^6, 2.5 \times 10^5)$	$(1.2 \times 10^5, 0.7 \times 10^4)$
dtse-TSC	$(6\,598, 92)$	$(2.4 \times 10^6, 4.9 \times 10^6)$	$(1.1 \times 10^5, 0.9 \times 10^4)$
Actuated	$(6\,529, 98)$	$(7.7 \times 10^6, 7.3 \times 10^5)$	$(2.2 \times 10^5, 1.2 \times 10^4)$