**Dissertation Plan**

**Title:**

**Abstract:**

**Table of Contents**

**Introduction / Motivation**

**Problem Background**

**Related Work**

**Description of Work**

**Technical problem description**

**Methodology**

**Implementation:**

**Evaluation / Results**

**Conclusions and Future Directions**

**Summary and reflections**

**Next Steps:**

* Collect results for advanced CIPHER model
* Set up LDO model to incorporate EMVs & Peds
* Do summary and reflections
* Set up LDO model to use batching
* Write up results for advanced CIPHER model
* Write up Linear Optimization methodology section
* Fill in all missing citations
* Put onto LaTex, add equations
* Proof read first draft

Exploring the Effects of Enhanced Environmental Data for Real-World Traffic Signal Control Using Reinforcement Learning

# Abstract

This paper covers the development of a Reinforcement Learning based agent and a Linear Optimization model that are both capable of routing cars, pedestrians and emergency vehicles across an intersection in a safe and efficient manner.

We propose two agents capable of handling the TSC problem: CIPHER (Control of Intersections for Pedestrians, Highway vehicles and Emergency Responders) and CIPHER+, which incorporates future vehicle arrival data into the state representation. To assess the impact of this more thorough state space, we compare CIPHER and CIPHER+ against six baseline methods from a range of TSC paradigms (static, adaptive and RL-based) using real world traffic data. Additionally, a linear optimization model is described, this model was used for both guiding the agents design and evaluating its effectiveness.

Our results conclude that CIPHER outperforms CIPHER+ across all simulated traffic densities, and that CIPHER is capable of handling cars, pedestrians and emergency vehicles in a safe manner. However, the results also highlight a trade-off when comparing CIPHER against algorithms focused solely on cars and optimizing their travel times. These findings highlight the potential for more RL-based in a real-world deployment situation, as well as the usefulness of linear optimization and classic adaptive TSC fundamentals in RL agent design.

Contents

[Abstract 2](#_Toc193387543)

[Introduction & Motivation 4](#_Toc193387544)

[Description of Work 4](#_Toc193387545)

[Problem Background 5](#_Toc193387546)

[Reinforcement Learning 5](#_Toc193387547)

[Deep Reinforcement Learning 5](#_Toc193387548)

[Linear and Discrete Optimization 6](#_Toc193387549)

[Literature Review: 6](#_Toc193387550)

[Intro 6](#_Toc193387551)

[Reinforcement Learning Literature 6](#_Toc193387552)

[Frameworks Used 6](#_Toc193387553)

[Simulator Choices 7](#_Toc193387554)

[Actors Involved 7](#_Toc193387555)

[Intersection Modelled 8](#_Toc193387556)

[State Representation 8](#_Toc193387557)

[Agent Actions 9](#_Toc193387558)

[Reward Function Choices 9](#_Toc193387559)

[Training and Testing Datasets 10](#_Toc193387560)

[Results 10](#_Toc193387561)

[Conclusion 10](#_Toc193387562)

[Linear Optimization Literature Review 11](#_Toc193387563)

[Intro 11](#_Toc193387564)

[Intersections Modelled 11](#_Toc193387565)

[Queue Transmission Models 12](#_Toc193387566)

[Objective Functions 13](#_Toc193387567)

[Constraints 13](#_Toc193387568)

[Optimizations 13](#_Toc193387569)

[Model Evaluation 14](#_Toc193387570)

[Conclusion 14](#_Toc193387571)

[Methodology 15](#_Toc193387572)

[Agent Design 15](#_Toc193387573)

[State 15](#_Toc193387574)

[Action 15](#_Toc193387575)

[Reward 16](#_Toc193387576)

[Linear and Discrete Optimization 16](#_Toc193387577)

[Model 1 16](#_Toc193387578)

[Model 2 17](#_Toc193387579)

[Implementation 18](#_Toc193387580)

[RL Agent Implementation 18](#_Toc193387581)

[Simulation 18](#_Toc193387582)

[Code 19](#_Toc193387583)

[Traffic Signal Control Manner 19](#_Toc193387584)

[Technical Challenges Encountered 19](#_Toc193387585)

[Linear and Discrete Optimization Implementation 20](#_Toc193387586)

[Results 20](#_Toc193387587)

[Conclusion and Future Directions 24](#_Toc193387588)

[Summary and Reflections 24](#_Toc193387589)

[Bibliography 24](#_Toc193387590)

# Introduction & Motivation

Modern road traffic systems face a series of diverse challenges: rush hour, emergency vehicle prioritization and imperfect driver behaviour are just some of the few driving forces behind why the problem of near-optimal traffic signal control (TSC) is a challenging one to solve. Original fixed-time control algorithms fail to cope with the ever-changing demands of modern road networks, resulting in increased journey times, increased vehicle emissions and reduced road user safety. A well-adapted traffic signal control algorithm should consider all road users, with a focus that extends past maximizing junction throughout to also prioritizing driver safety, emergency vehicle response times and extending fairness to pedestrians and cyclists, in the same manner it does for cars.

The economic cost of the problem alone makes it an enticing one to solve. One study (Inrix, 2018) estimates that traffic congestion costs Americans an average of 97 hours a year. This figure extends up to 164 hours in population-dense cities such as Boston, with an annual cost of $2291 per driver. Meanwhile, in Europe, urban traffic congestion is estimated to cost almost 1% of the EUs annual GDP (Panayotis Christidis, 2012), equating to EUR270 billion. Another study (Shashank Bharadwaj, 2017) found a direct correlation between traffic congestion and CO2 emissions, showcasing the need for proper traffic management in our increasingly carbon conscious world.

Recent advancements in machine learning techniques, namely reinforcement learning (RL), have opened the door to Computer Science and Optimization researchers in the context of the TSC problem. RL-based systems can be trained to effectively handle and process a wide variety of environment data to optimize road traffic signals based on a set of varied success criteria, in real time.

Whilst numerous studies have shown the effectiveness of RL in the context of the TSC problem, a systematic literature review demonstrates two critical gaps in the problem that I will attempt to address:

* There still exists a large gap between real-world and simulation, most pre-existing literature ignores the presence of certain road actors such as cyclists and pedestrians. My research will consider fairness and response times to all road users, not just vehicle drivers.
* Settling a debate from conflicting literature over the effect of an increased number of states and providing enhanced environment data to my model.

# Description of Work

This project aims to design a reinforcement learning agent that is capable of routing cars, emergency vehicles and pedestrians across a junction effectively. The project integrates with SUMO to assist with modelling and visualising traffic flow. This project conducts experiments on what constitutes effective state representation and conducts a series of experiments using real-world data to validate this.

The project includes a variety of classes to assist with training and visualising the agent.

as well as a linear optimization model that will be used for evaluating the agent’s effectiveness.

description of the work explaining what your project is meant to achieve, how it is meant to function, e.g., even a functional specification for a software-oriented project.

# Problem Background

This section introduces the technical preliminaries of reinforcement learning that are required to understand this paper.

### Reinforcement Learning

Reinforcement learning [CITE HERE]? is a subset of artificial intelligence that enables a model to learn through trial and error, rather than utilizing a specific training dataset. Reinforcement learning leverages the Markov decision process (MDP). A framework for decision making in an environment with uncertain outcomes. In RL there are five main components to consider:

* Agent – the learner or decision maker
* Reward – A numerical variable returned to the agent that indicates how favourable a particular state is.
* Actions – the set of actions the agent can undertake; this action is executed on the current environment to produce a new state.
* State – represents the current configuration of the environment in which the agent is deployed.
* Policy - The set of learned actions the agent will take when in a particular state.

RL agents select the most appropriate action based on the long-term reward they can expect to see from it, this value is called a Q-value and there is one assigned to every state x action pair. The Q-values are stored in a Q-table and learned through a process called Q-learning. Q-values are updated using the Bellman equation [insert bellman eq here].

A red arrow pointing to a number of mathematical equations

AI-generated content may be incorrect.

The table is initialized with all Q-values of zero, the model will pick an action, measure the immediate reward returned by the environment, update the Q table and repeat. After multiple iterations a well fit Q-table is ready to be used as a policy which will instruct the model on the most optimal action to take within a specific state.

### Deep Reinforcement Learning

Classical reinforcement learning will struggle to scale for large problems due to the curse of dimensionality – as the state / action space grows so will the respective Q-table. This large Q-table soon becomes computationally infeasible to store and update values. This issue is particularly common in environments with continuous state spaces or a diverse set of potential actions.

To address this limitation, DRL replaces the Q-table with a neural network that is used to approximate Q-values and derive an effective policy. The bellman equation is still used, serving as a comparison value to backpropagate weights and biases in the network. The model receives state information through the input nodes, and outputs Q-values for each possible action, the action with the largest Q-value is enacted upon the environment. Pseudocode for the DRL training process is defined below.

### Linear and Discrete Optimization

Linear and Discrete optimization is a form of mathematical optimization in which problem variables can be either continuous or discrete. It is not discussed here for brevity; an interested reader can find more information from [CITE HERE].

# Literature Review:

## Intro

This literature aims to provide a comprehensive account and comparison of the various approaches to formulating and solving the Traffic Signal Control problem.

TSC methods can be categorized into one of three main types:

Fixed Time Systems: These operate based on a cyclical structure with a fixed time for each traffic phase.

Actuated / Dynamic Systems: These follow rule-based conditions to dictate signal activation, such as extending green phases for a specific road during rush-hour.

Adaptive Systems: These will dynamically adjust the phase length and cycle pattern based on real-time traffic data, to optimize flow in response to current traffic conditions. Notable algorithms include Websters (Webster, 1958), MaxPressure (Varaiya, 2013) and SOTL (Gershenson, 2004).

More recently, research has been focused on leveraging computational methods in the problem context to develop adaptive systems by leveraging heuristics, metaheuristics, artificial intelligence and mathematical optimization. This literature review will focus on the latter two.

## Reinforcement Learning Literature

### Frameworks Used

Reinforcement learning was first applied to the problem through Q-learning with the addition of genetic algorithms to efficiently search and update hyper-parameters across training. Despite being a valid first step in applying RL to the problem, the paper (Kakazu, 1994) rapidly encounters the curse of dimensionality due to the wide and complex state space of the problem. The model is only able to operate within a minimal simulation environment. (Prashanth L. A., 2011) addressed this with function approximation techniques employed to reduce state-space and negate the issues encountered by Kakazu et al. More details on this are discussed in section [INSERT HERE]

Modern research has focused on leveraging neural networks through DRL, to cope with the high-dimensional state space. Studies such as (Huang, 2024) (Tham, 2007) (Romain Ducrocq, 2021) (Mobin Yazdani, 2023) (Haoran Su, 2022) have demonstrated the superior results that can achieved through DRL models. The absence of a large Q-table noted in other studies makes the system more precise and scalable.

### Simulator Choices

Urban simulators are an effective way to model vehicle flow and capture key complexities such as driver behaviour and vehicle acceleration. Using a sophisticated simulator will ensure more accurate adaptation to the model’s real-world deployment environment.

The simulation tools available fall into one of three categories:

* Microscopic simulators consider individual driver behaviour and their interactions with other road users.
* Macroscopic simulators abstract individual details and focus on vehicular flow, without modelling individual vehicle dynamics.
* Mesoscopic simulators are a hybrid of both approaches and will model overall vehicle flow with some consideration to individual behaviour.

With the exception of (Kakazu, 1994) and (Tham, 2007), who developed their own macroscopic environment, the other papers considered here used third-party, microscopic simulators. The most popular simulator noted was SUMO (Lopez, 2018) used by 3 papers (Huang, 2024) (Romain Ducrocq, 2021) (Haoran Su, 2022), other simulation environments used were VISSIM (Martin Fellendorf, 2011) and GLD (M. Wiering, 2004). Another literature review (Syed Shah Sultan Mohiuddin Qadri, 2020) collated 57 papers and found that 33% of papers studied used VISSIM whilst 25% of papers used SUMO, this highlights the two software’s dominant superiority across the field of urban simulation.

One notable shortcoming across the literature is the lack of simulation parameters disclosed, meaning it is hard to replicate and verify results. Factors such as: max car speed, acceleration, road length, vehicle length and driver imperfection rating are often omitted, despite the sensitivity of final results regarding these parameters.

### Actors Involved

Simulation tools allow for the inclusion of various road actors. SUMO allows for the inclusion of Cars, Cyclists, Pedestrians, Emergency Vehicles and Public transport. Despite this, most papers studied focus exclusively on cars. This shortcoming limits the real-world validity of many studies and is a major barrier to deploying systems into the real-world.

One exception to this statement is EMVLight (Haoran Su, 2022). The paper proposes a framework for coupling efficient EMV routing with a RL-based traffic signal control agent. The model aims to reduce travel times for both civilian vehicles and emergency ones. Similarly, IVPL (Mobin Yazdani, 2023) extends the simulation environment to involve pedestrians. The work is based off a pre-existing model [cite here] but extended to penalize pedestrian delays. Real-world validity of this work is extended by simulating jaywalking and studying the effects this realistic factor has on model performance.

Despite the efforts of these two studies, to the authors knowledge, there are no available studies that focus on developing a RL agent that can effectively handle all actors it would encounter in a real-world deployment. Further investigation into this area would not only reduce overall congestion but also potentially result in:

* Optimized emergency response times
* Increased walkability and pedestrian safety by ensuring fairness across signal timings.

### Intersection Modelled

The TSC problem extends to a wide variety of road configurations but is most commonly studied in the context of one or more networked four-way intersection. This could be attributed to the complexity that makes solving such a problem highly desirable: incoming vehicles from all four directions, competing for control of one singular intersection.

A graph of a number of traffic models

AI-generated content may be incorrect.Whilst some studies ( (Huang, 2024), (Romain Ducrocq, 2021), (Mobin Yazdani, 2023)) focus on creating an agent that can effectively control just one of these intersections, others ( (Kakazu, 1994), (Prashanth L. A., 2011), (Tham, 2007), (Haoran Su, 2022)) proved the scalability of their models across multiple intersections networked together. Such a setup allows for experimentation on the results of increased environment context for the models. Shown right are the results from (Tham, 2007) who networked two RL-based agents together to allow for sharing of information. This model (case III) was compared to two solitary models (Case II) and a fixed time model. As shown by the graph, allowing the networked agents to communicate environmental data between one another resulted in significantly reduced average delay within the simulation.

### State Representation

As outlined in section [INSERT HERE], solving problems such as TSC requires them to be formulated as an MDP. Given the problems infinite, continuous state space; efficient representation of the current environment state will enable a model to effectively converge on an optimal policy.

Common state factors observed across the literature are:

* Current phase – a vector representation of lights that are currently green.
* Vehicles in lane – the number of vehicles currently on a lane
* Queue lengths – the number of vehicles queueing on a lane, a vehicle can be defined as queueing if its current speed is 0m/s (stationary).

As noted earlier, Q-learning based algorithms are hard to apply to large problems with complex state spaces, due to the curse of dimensionality. However, (Prashanth L. A., 2011) applied function approximation techniques to categorize the continuous state space within discrete categories. Reducing the size of the state / action pair space from 10101 to just 200. However, this approach proved to be unscalable to larger intersections due to the state space still growing exponentially.

Networked, multi-intersection agents naturally extend their state space with information from other intersections. (Tham, 2007) state information would also include the action and Q-value obtained from any downstream agents, so that upstream agents could anticipate incoming traffic.

State space size also grows with actors present in the simulation, as there is more information to represent. (Mobin Yazdani, 2023) algorithm was focused on the inclusion of pedestrians, and state information would include pedestrian numbers on each crossing. Meanwhile, (Haoran Su, 2022) included comprehensive information on emergency vehicles: the distance of the EMV to the intersection, the ETA of the EMV at the intersection, and the EMVs proposed next step.

By extending state representation to allow for multi-agent co-ordination and varied road users, research is closing the gap between real-world and simulation. However, the computational resources required to develop a comprehensive policy grows exponentially with the state space, and this remains an open challenge in DRL-based TSC.

### Agent Actions

All RL-based algorithms studied have their action focused on picking the next, most-optimal traffic phase. Careful design is needed to design an action space large enough that it can comprehensively respond to all possible states, whilst remaining small enough that training the model is computationally efficient.

The majority of papers noted in this study will choose phases in an acyclic manner. (Hua Wei, 2018) instead opted for a cyclical structure and a binary action space {0,1} where 0 opts to stay in the current phase and 1 will transition to the next. Whilst this policy may be efficient to train to, it gives a much more limited flexibility in comparison to some of the other studies.

Some papers (Haoran Su, 2022) (Romain Ducrocq, 2021) opt to provide the model with a pre-configured set of phases, where the action space size reflects the different phases available to the model, which will output a score for each, indicating the suitability.

Other papers (Tham, 2007) (Huang, 2024) have taken a more flexible approach, allowing the model to select both the next active phase, and its duration, Tham et al’s action space was a continuous one. 0.. n, where n is the number of phases available to the model, which will output a continuous (bounded) value for each indicating the green time it should be allocated. As noted by the author of this paper, it is important to consider the negative effects of such a large action space, even on a simple 2-light intersection, indicating this approach may not viably scale to larger scenarios.

(Mobin Yazdani, 2023) uses a hybrid approach to iteratively clear pedestrians and allocate a constant amount of green time to each car lane. If there are still vehicles remaining in the simulation the RL-agent will allocate additional green time to each lane according to the policy, if there are no vehicles present the cycle repeats. Whilst this approach can ensure fairness across both groups, it does not allow for pedestrian lights and vehicle lights to be green at the same time, even if the two routes do not co-inflict.

### Reward Function Choices

Effective reward functions allow for dictation of the model’s priorities and can differ from problem to problem.

Some reward functions in the literature focus on minimizing cumulative user delay, this correlates well with the real-world objective of minimizing user travel times. Furthermore, this value can also be exponentially weighted to penalize longer wait times. However, it also requires constant vehicle tracking, something much easier done in a simulation environment than real world.

[INSERT EQUATION HERE]

Another common function is to maximize junction throughput, this reward style will encourage smooth traffic flow but may result in unfairness to certain road users, as there is no incentive for the model to regulate wait times.

[INSERT EQUATION HERE]

One intuitive study (Hua Wei, 2019) acknowledges that the primary objective: reducing travel time, is a culmination of micro factors. This model instead implements a reward focused on minimizing intersection pressure: a metric defined by the adaptive algorithm MaxPressure. The introduction of RL-based methods to the MaxPressure algorithm proved more effective than the standalone MaxPressure algorithm.

[INSERT EQUATION HERE]

### Training and Testing Datasets

Due to there being no known optimal solution for the TSC problem, evaluation is usually done through comparison of different models. A significant challenge regarding this is the low availability of open source, consistent datasets, meaning most studies use synthetic data for the training and evaluation. Arrivals are often simulated with a Poisson or Gaussian distribution to mimic arrival patterns.

Certain papers (Mobin Yazdani, 2023) (Huang, 2024) have used real-world datasets to evaluate their models. However, it is observed in an alternative literature review (Syed Shah Sultan Mohiuddin Qadri, 2020) that 42.2% of studies would use generated data for evaluation. This once again, raises questions about the real-world validity of the models.

From the literature review, it is clear that the lack of real-world data is a serious barrier to standardization and real-world deployment of models. The lack of datasets could be attributed to privacy concerns, or practical difficulties in collecting such a comprehensive dataset. Nevertheless, work focused on capturing and processing detailed traffic data would be a valuable contribution to the field.

### Results

From the literature, it is appropriate to say that RL-based TSC algorithms are capable of outperforming non-RL adaptive ones and fixed time algorithms in a variety of metrics including junction throughput, vehicle delay and overall travel time. Studies that included pedestrians (Mobin Yazdani, 2023) and EMVs (Haoran Su, 2022) were also able to extend fairness to these actors.

Interestingly, there are conflicting results across the literature about including information on the wider environment for the model. (Tham, 2007) observed significantly improved results from networking two intersections together and (Haoran Su, 2022) successfully routed EMVs by predicting their arrival times. In contrast, other papers [cite here from interim report] were able to outperform various benchmark models through minimalist state design, leading to reduced model training times and costs.

### Conclusion

[insert interim report table, here? Or elsewhere]

The literature shows extreme potential for the application of RL paradigms in the context of TSC and DRL specifically as a promising approach for handling such a complex problem. However, it also highlights the significant gap between real-world and simulation. The majority of papers cited only consider cars, meaning it is impossible to evaluate how the models will perform when deployed and encounter EMVs or pedestrians. Although certain papers have set out to include one or the other, to the author's knowledge, there is no model capable of effectively handling all 3.

Furthermore, there is a lack of standardization across the field, models are often designed and trained across varying simulation environments and road networks. Detailed simulation parameters are often not included, making it hard to reproduce results.

From this review, it is clear that future work should be focused on:

* Producing real-world datasets for standardized evaluation across models,
* Developing systems capable of optimizing vehicle, pedestrian and EMV travel times.
* Conducting research into the effects of supplying increased environment data to the model.

## Linear Optimization Literature Review

Recent work has also focused on using linear optimization methods to solve the problem. These models leverage mathematical programming techniques to determine (near) optimal traffic settings. A systematic review of the remaining literature highlights two flaws also commonly seen within RL based studies: a lack of real-world validity and a high computational cost associated with running these models.

Despite these flaws, LO-based methods are able to provide a near globally optimal solution, meaning they can provide a strong baseline for comparison against RL-based approaches and serve as a highly valuable area of future research.

### Intro

Out of the 4 papers (Kentaro Wada, 2017) (M.A.S. Kamal, 2013) (S.M.A. Bin Al Islam, 2017) (Guilliard, 2020) noted in this study, all formulate the key decision variable as an array arr[S][T] ex {0,1}. Where:

* S represents a specific traffic signal present in the network
* T represents a discrete time step
* Arr[S][T] ex {0,1} indicates whether signal S is green at time T.

This binary formulation offers a flexible and effective representation of the problem as a Mixed integer Linear Programme (MILP). However, the large number of binary variables does not optimize well, particularly when using the branch-and-bound method used by LO solvers today.

### Intersections Modelled

Unlike the microscopic approach noted amongst the RL based studies, all LO models took a macroscopic approach to modelling the intersections, this can be attributed to the challenges in representing microscopic factors such as individual driver behaviour mathematically and the inability to leverage simulators like SUMO when solving the problem in this manner.

Similar to the RL-based papers studied, the literature in this field covers both singular and networked intersections. However, all intersections modelled represented a simplistic layout, with one lane per incoming direction. This reduces the problem size and simplifies constraints in that only one light can be active at a time. There remains an important direction for future research into models of more complex intersections, which would theoretically allow for multiple lights to be safely green at the same time.

Furthermore, the vast majority of LO studies focus exclusively on vehicle traffic, mirroring a similar limitation seen in RL studies. Amongst the four papers reviewed, three exclusively considered cars, while only one paper (Guilliard, 2020) had the additional actor of light-rail systems. This study was able to successfully model constraints surrounding this and optimize the phases in regard to both traffic flow and rail systems simultaneously. The bias towards cars in the literature is evident. The lack of extraneous road actors considered, and the omittance of microscopic factors make the results from these model’s mere approximations, rather than fully developed solutions that would be expected to work in a real-world environment.

### Queue Transmission Models

A fundamental of modelling the TSC is the equation used to model vehicle flow across a road network and the buildup of traffic queues. This area of the problem will be subsequently referred to as the queue evolution strategy. One of the first such strategies was initially proposed by (Miller, 1963). This is called the Queue Transmission Model (QTM). Miller states that the queue for one approach at a single intersection, at a given time step t can be modelled as [INSERT EQUATION HERE]

Across the modern literature there were three primary approaches observed that aim to utilize and build off of Miller’s work:

* CTM (Cell transmission model) – Initially proposed by (Daganzo, 1995) and building directly upon the work of a QTM proposed by (Miller, 1963) . This was incorporated into models by (M.A.S. Kamal, 2013) (S.M.A. Bin Al Islam, 2017). Daganzo broke each road segment into discretized distances of a fixed length, a section is denoted by its origin (o) and destination (d) as (od). The continuous variable vod denotes the number of vehicles on the section (od) at time t. Therefore, the queue update equation can be modelled as:



Where pout and pin are the inflow and outflow traffic of a section respectively. Whilst the extension of the CTM upon the QTM gives the model a finer representation of vehicle positions within the network and partially satisfies the lack of microscopic representations noted previously, it also extends the problem space by representing each road as a series of segments rather one as one large cell. With that said, employing a CTM is a necessity for modelling multiple intersections networked together, as these two papers did.

* Extended CTM – Building upon the work of (Daganzo, 1995) was (Guilliard, 2020). Guilliard correctly stated that a CTM is only advantageous (and necessary) over a QTM when a roadway does not diverge or merge into another. Guilliard then employed a hybrid approach for the complex road networks modelled, only using a CTM when strictly necessary. Further optimisations employed by Guilliard are discussed in section [INSERT HERE]. Although because of this extended CTM he was able to scale the model to effectively handle a 3x3 grid of intersections and still achieve optimal results, more than any other paper considered in this study.
* Kinetic Wave Model– proposed by (Daganzo, 2005) and utilized by (Kentaro Wada, 2017) states that traffic flow can be characterized by: ur (forwardwavespeed),wr (backwardwavespeed), qr max (saturationflowrate),andκr j (jamdensity). Unlike the CTM, this model captures the stochasticity of traffic flow, a valuable microscopic factor that most models do not. However, implementation of this equation with the model also results in more binary models than the previous two queue evolution strategies.

### Objective Functions

A variety of objective functions were noted across the literature, perhaps the most sophisticated being presented by (Guilliard, 2020) who leveraged the extended CTM to calculate exact travel times for vehicles in the simulation. Other objective functions would focus on minimizing total traffic in the network at any given time (M.A.S. Kamal, 2013) or similarly, maximizing junction throughput (S.M.A. Bin Al Islam, 2017).

### Constraints

There were also multiple similarities noted across the literature in terms of the problem constraints noted. Multiple papers (S.M.A. Bin Al Islam, 2017) (Guilliard, 2020) (Kentaro Wada, 2017) imposed constraints on both the minimum and maximum green time. The respective authors cited various reasons such as: mimicking pedestrian clearance, safety and realism.

Furthermore, all papers modelled cars travelling through the network by using a constant saturation rate parameter. Formulated as: [insert equation here: dcell,time < lambda. Where lambda is the maximum number of vehicles that can travel across a cell per time step.

One notable shortcoming of all papers cited above is the lack of explicit mention of yellow signals within the problem. Yellow signals will cause lower throughput than a green signal due to the necessary acceleration/deceleration of cars from the preceding or succeeding red phase. This omission could be attributed to the fact that the inclusion of yellow signals would introduce another set of binary decision variables. Instead, researchers modelled this lost start-up time in various ways: (S.M.A. Bin Al Islam, 2017) introduced a constraint to reduce the previously mentioned saturation rate parameter at the initiation of a green signal, to account for the lost start-up time. Meanwhile, (Guilliard, 2020) (Kentaro Wada, 2017) both introduced a penalty for phase switching in the objective function.

### Optimizations

A consistent theme throughout the literature is the high number of binary variables involved in representing the problem, despite optimization attempts such as the extended CTM. Other successful optimization attempts include splitting time up into discrete intervals. This was an optimization proposed by (Kentaro Wada, 2017) where instead of a decision variable being represented as an array of time \* lane size it is instead represented by an array of size (time / lambda) \* lane, where lambda is the discretized time step value. Although being an efficient reduction of problem size it is important to note the loss of fine-grained control with such an approach. This is due to all arrival departure information from time steps n to n + lambda being collated within one array cell.

Alongside the extended QTM model, (Guilliard, 2020) laid out further optimizations. By considering just a small problem window at the same time to drastically reduce the problem space. Furthermore, when running the model on networked intersections, the model was run on subsets of the network in parallel. Although successful in reducing the computational cost of solving the issue, it was acknowledged by the researcher that this approach would obviously lead to producing only an optimal solution to the problem.

### Model Evaluation

Across the literature there are notable flaws in the data used to evaluate model results. No papers studied made use of real-world data and instead opted to modelling traffic arrivals assuming either a linear (Guilliard, 2020) (S.M.A. Bin Al Islam, 2017) (M.A.S. Kamal, 2013) or Poisson (Kentaro Wada, 2017) distribution. Theses distribution methods do not effectively capture the stochastic and variable demand that modern RTNs encounter. Future studies should look to include real world data when evaluating their models.

All studies reported levels of success in developing a model that can find (near) optimal solutions to the TSC problem; however, all studies also cited the computational times as being the key barrier to deploying these solutions to the real world. One interesting set of results comes from (Kentaro Wada, 2017), who experimented with the level of control the model had over signal green times. Three different scenarios were tested: model has full control over a phases green time, green time is constrained to an exact integer value, green time is calculated using Webster’s formula (Webster, 1958). It was recorded that giving the model full control over green time gave the most optimal solution.

### Conclusion

The literature review has shown that studies often take a similar approach to problem representation. Common themes include modelling the environment macroscopically and utilizing binary decision variables to represent signal status. Whilst having been proved through successful results as a correct representation of the problem, the large problem space and high occurrence of binary values leads to a high computational cost being associated with the problem. Although certain optimization attempts have been made and were successful in reducing the computational complexity, reductions were not enough to allow for modelling of additional microscopic constraints.

The omittance of modelling microscopic factors and the lack of real-world data used in previous studies means that models are not an accurate reflection of real-world studies seen today. Future work should focus on modelling the problem as one with less binary decision variables, consider the inclusion of more realistic constraints where possible, and focus on the introduction of real-world datasets in model evaluation.

# Methodology

A crossroad with a crosswalk

Description automatically generatedThe TSC problem has been applied to a variety of different road networks and environments. This paper will focus on the most common one: a classic 4-way intersection with 3 incoming lanes and one outgoing lane in each way (shown right). Across each incoming way is also a pedestrian crossing. With this configuration there are 16 traffic lights that must be configured as one of three potential states to form the phases.

## Agent Design

### State

The state representation of the agent combines both continuous and discrete factors to effectively represent the wide range of road actors present in a simulation at time t. The state of an agent at time t can be defined as:

Queue lengths – The number of cars currently waiting in queue j at time t. A car is only defined as ‘waiting’ if its current speed is zero.

Phase status – A binary array representing whether a light is green or not at time t, since the state is captured before a phase transition occurs, information on yellow phases is not relevant here.

EMV distance – The distance of the nearest EMV approaching from each direction.

Pedestrian flag – A binary value indicating whether the number of pedestrians at a crossing is exceeds zero.

Enhanced environmental data – to satisfy one of the experimental goals of this paper, investigations were carried out upon the effect of providing the model with the expected number of vehicles arriving per lane before the next action, decision, results of this experiment are discussed in section [INSERT HERE]. A lookahead value of 30 time steps were chosen for this parameter. This was intentionally equivalent to the amount of time steps in-between model inference times, as information past this point would be covered by remaining state variables.

As highlighted by (Jiahui Yu, 2022), data normalization helps enhance numerical stability of neural network-based models. Therefore, all continuous state information was normalized using min-max normalization [insert equation here] to normalize values between a range of 0-1. This is with the exception of EMV distances, which were normalized to be between 0 and 1 according to a known upper and lower bound.

### Action

The action space for the model consists of 36 different traffic light phases, phases are not constrained to a cyclical structure, but each phase is forced to be active for 30 simulation time steps.

As observed by [CITE HERE], providing models with a large action space will result be more expensive in both time and computational resources, when training the model to converge on an optimal policy. For the environment defined above, there are 16 traffic lights each with 3 possible states (red, yellow, green). This gives 3^16 = 4096 configurations for the traffic lights. However, the majority of these are either unsafe or inefficient, the following assumptions and constraints have been applied to reduce the action space down to 36.

• Every green phase of a traffic light must be preceded and succeeded by a yellow phase, in the interests of driver safety. This is handled externally by the environment; the model does not need to predict yellow values. This reduces the action space size to 256.

• Routes that co-inflict with one another (e.g. vehicle paths from two different routes would cross) should not be active simultaneously as this would result in crashes or gridlock. This removes 45 phases, leaving an action space of size 211.

• The most efficient implementation will have the maximum number of routes (4) active at the same time. This assumption further narrows down the action space to just 36 potential phases the model can select.

### Reward

As concluded by (Liu, 2024), waiting-time based reward functions result in increased mean-vehicle speed as well as a reduction in CO2 emissions. Waiting-time centred reward functions were also noted to outperform vehicle-speed based rewards and emission-based reward functions across a span of metrics. Thus, the reward for the model is defined as follows: min([INSERT REWARD HERE])

The reward function is defined so as not to penalize the model that entered the simulation during the execution of the traffic phase and thus the model was previously unaware of. It gives the model incentive to minimize pedestrian and vehicle wait times. In addition to this, a collision penalty is applied to reward safe routing of emergency vehicles. Some pre-existing work has implemented a squared reward function, to penalize longer waiting times more; however, it was observed that this would cause numerical instability in the CIPHER model and was therefore not included.

## Linear and Discrete Optimization

The literature review concluded that a valuable contribution to the field would be to formulate the TSC problem as one with fewer binary variables, as this would allow for the modelling of more precise and realistic constraints.

### Model 1

#### Decision Variables

An initial attempt was made to propose a new model that utilized two continuous decision variables to indicate whether a signal was active or not.

[insert these two decision variables here]

Additional helper variables for tracking queue length and departures were also used:

[insert here]

#### Objective Function

An objective function focused on minimizing cumulative queue length:

[insert here]

#### Constraints

An additional constraint not seen in the pre-existing literature was also needed, to ensure a light cannot reactivate until after it has been switched off.

[insert light activation constraint here]

Other constraints were also used, which were seen commonly across the literature.

[insert conflicting signals constraints]

Using enhancements proposed by (Guilliard, 2020), it can be concluded that for our network, Daganzo’s (Daganzo, 1995) CTM will suffice, as there are no diverging roadways in our network. Therefore, our queue evolution strategy is as follows:

[insert queue evolution]

Along with helper equation, to ensure departures do not exceed the queue length:

[insert here].

However, issues were encountered with the model when trying to calculate da,t.

[insert problem constraint here]

With linear optimization techniques it is not possible to implement comparatives between decision variables, as these are continuous and not pre-defined. Therefore, it was concluded that it was not possible to implement the TSC in a continuous manner and this approach was unfortunately abandoned.

### Model 2

Following the failed experimentation of the previous model, a new formulation was created with the aim of applying certain noted optimizations to allow for modelling of select important constraints, whilst still representing the decision variables as a binary one. This model has two binary decision variables:

[insert green equation here]

[insert yellow equation here]

The same helper equations [cite here] seen from the previous model remain the same.

#### Constraints

Naturally, with the introduction of the possibility of yellow phases, it follows that a light cannot be yellow and green at the same time. Therefore, a constraint was introduced:

[insert here]

A similar queue evolution approach was used to the CTM, however extended to account for the reduced saturation rate when a light is yellow, in response to vehicle acceleration and deceleration before / after a red signal.

[insert here]

#### Objective Function

The same objective function [cite here] was carried over from the previous model.

Owing to the problem now having twice the number of binary decision variables, optimizations needed to be introduced. Alongside the simplified CTM being employed, it was decided that time steps would be discretized to intervals of 5, as initially proposed by (Kentaro Wada, 2017). For an instance of the problem with 16 lanes and 1000 time steps, this reduces the number of binary decision variables from 16,000 to 3,200.

The literature review also suggested improvements in the types of actors represented in the model. Therefore, this formulation was extended to include the same actors as the reinforcement learning model described earlier: pedestrians and EMVs.

This required the implementation of the following decision variables:

[insert ped departure here]

[insert emv departure here]

And the following constraints were defined to mimic EMV priority:

[insert constraints here]

And, to mimic pedestrian flow

[insert constraint here]

Finally, the objective function was extended to reflect this:

[insert here]

Various iterations of this model will be used to feed into training decisions for the RL-based model described previously, as well as for evaluating the effectiveness of the final model.

# Implementation

## RL Agent Implementation

### Simulation

As concluded by numerous literatures [CITE HERE], SUMO is widely accepted as one of the most effective and popular road traffic simulation tools available for use. Due to its flexibility, GUI and strong integration capabilities, it was decided that SUMO would be used to model the environment within which the agent operates.

The lack of extensive real-world data sources meant that data augmentation strategies had to be used for training the agent. Vehicles were generated according to a Poisson distribution [INSERT EQUATION HERE] with a mean inter-arrival rate of 2 simulation time steps.

It was noted across the literature that there was a lack of simulation parameters provided to the reader which limits readability. To improve this, all relevant simulation parameters are included below in table [INSERT HERE].

### Code

Communication through SUMO has originally handled via the Traci API, a web-socket based library provided by the SUMO developers for retrieving and updating various simulation parameters. However, due to the high communication overhead caused by the socket-based protocol used the project was migrated to LibSumo (a more efficient C++ API). This change significantly reduced training times for the model.

The core implementation was written in Python, leveraging LibSumo to interface with the simulation environment. The system architecture consists of:

* Environment Wrapper: Provides a standardized interface for the RL agent to interact with SUMO.
* Simulation Runner: Loads a specific traffic signal control (TSC) policy and steps through the simulation, with an optional GUI for real-time visualization.
* Controller Modules: Two controllers were implemented for debugging purposes and experimentation:
  + Fixed-Time Controller: Operates on a predefined cyclical schedule.
  + RL-Based Controller: Selects actions based on a trained reinforcement learning model.

The framework for my DRL model was taken from the PyTorch Docs [CITE HERE] and adapted to function in my chosen environment. Instances of the model were originally trained locally, but soon computational demand grew and training was offloaded to the universities GPU cluster for more efficient training. The model comprises of 3 hidden layers, each with 128 neurons per layer and a ReLu activation function.

The agent was trained on a different set of randomly generated arrival times each iteration, to avoid overfitting to one specific arrival pattern. It is important to note that mean-inter-arrival times between cars were kept consistent across the entire training process, to enhance the numerical stability of the reward function.

[INSERT UML DIAGRAM HERE]

### Traffic Signal Control Manner

As described in section [INSERT HERE], numerous steps were taken to provide the model with an action space small enough (36) to allow for rapid convergence on a policy whilst the mandatory 36 other yellow phases would be enforced by the environment wrapper itself. However often two successive phases will have green lights in common, switching these lights to yellow and back to green again would unnecessarily slow down cars and reduce junction throughput. A specific light control module was developed to negate this, a system was built off of base-3 arithmetic to efficiently represent over 443 unique phases and allow the environment wrapper to select the most efficient one according to the agents action, whilst still keeping the RL agents action space as small as possible.

### Technical Challenges Encountered

The main technical challenge encountered over the course of development was the training times for each instance of the model, as state space grows the model has more weights to tune with each iteration, and backpropagating this correctly takes time. Furthermore, due to the complex environment within which the agent is operating, convergence on an optimal policy would take time. Training one instance of the model can take up to 24 hours in some cases.

This effect was negated as much as possible by optimizing code where possible. As noted earlier, upgrading to the LibSumo API reduces the communication overhead in interacting with the simulator. Other optimizations were also implemented such as limiting batch size initially during training and only scaling this number up when an acceptable model configuration was discovered. These changes helped streamline training times as much as possible, but model training times was still a big technical blocker throughout the course of the project.

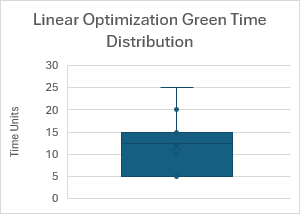
Another technical challenge was the frequent experimentation needed between minorly different versions of the mode. Successive model versions would often only have small tweaks of independent variables such as the state information, reward function or hyper-parameters. However changing variables in isolation was a necessity due to the highly sensitive nature of the environment in which the agent is deployed. Naturally, each model incurs the high training times previously noted. To minimizing the time spent comparing two models, a comprehensive metric logging system was created. The system automatically collects and graphs metrics surrounding cars, EMVs, Pedestrians and the environment as a whole. This would allow me to compare results visually and quickly decide on which was the better of the two models.

## Linear and Discrete Optimization Implementation

* Talk about batching the time steps
* What else? – talk about how car departures and ped departures were aggregated into one array for simplicity

# Results

* Talk about training of CIPHER, times, resources, compare to instance of running LDO model.

When training the models, a hybrid approach was taken to avoid expensive tuning of the green time hyper-parameter: results from the linear optimization model were analysed to influence decision-making regarding particular hyper-parameters for the RL model. Green time dictates the fixed amount of time steps that a light is green for, whilst a shorter green time will give the model finer control over the simulation and its decisions. A shorter green time will also result in more inference calls on the model, which will in turn increase training times; therefore, careful balancing of this parameter is essential.

The linear optimization model described in section [insert here] was run using a synthetic arrivals dataset and the results analysed. Figure [insert here] shows the distribution of green time as chosen by the linear optimization model, from this we can conclude that the most optimal green time duration lies in-between 10-15 time units. Therefore, a green time of 15 time units was selected to balance model control and computational efficiency.

Another notable set of results from this running of the LO model is shown in figure [INSERT HERE]. Postprocessing of the linear optimization data was conducted to analyse the number of lights active at each time step. The graph shows the frequency that each number of lights was active for. This chart supports the assumption made in section [INSERT HERE], that most of the optimal phase configurations will have 4 lights active at a time.

As noted in the literature, there are no pre-existing models which account for emergency vehicles and pedestrians in the road network, meaning direct comparison between CIPHER and similar models was not possible. However, a simplified version of CIPHER was taken for comparison, one which was trained with cars training data and only having information that represents cars in the state information. This could at least server as a basic measure of effectiveness of the model’s design.

To assess the effectiveness of CIPHER and CIPHER+, 6 TSC models were chosen for comparison, covering a range of traditional, adaptive and RL based techniques:

* Fixed – The static fixed time algorithm used in over [insert here] % of road networks today.
* Adaptive – Adaptive, mathematical based models such as: Websters, SOTL, MaxPressure.
* Adaptive RL – Two RL based algorithms were also included in the comparison. Both algorithms used a deep reinforcement learning framework. Synthetic arrival data was generated using a non-homogenous Poisson distribution.
  + DDPG is a RL based algorithm that allows for selection of both the next phase, and its cycle length.
  + DQN works similarly to CIPHER, simply selecting the next best phase and keeping it active for a hardcoded cycle length.

Both RL agents’ state representation consists of lane density and the last green phase, the reward functions are both focused on minimizing total vehicle delay. Synthetic arrival data was generated using a non-homogenous Poisson distribution. Further implementation details of all models mentioned, source code can be found at (Razavi, 2019).

All models were evaluated using a real-world dataset, obtained from a traffic intersection in Hangzhou China. The original dataset comprises of vehicle arrivals and routes over a one-hour period. Arrival times from this dataset were then linearly scaled down to half an hour and 15 minutes respectively, this allowed for testing of models across 3 different traffic demands: low, medium and high. All models were evaluated based on vehicle travel times.

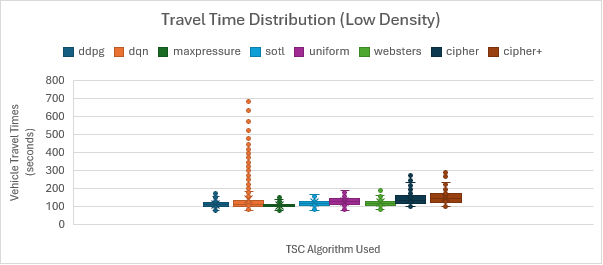


Figure [INSERT HERE] presents the distribution of travel times from each model in a low-density (1.2 second mean inter-arrival rate) environment. CIPHER and CIPHER+ exhibit the longest mean travel times among the eight models, indicating a sub-optimal performance. However, both models also result in fewer extreme outliers in comparison to DQN, suggesting that whilst they may increase average travel times, they also result in fairer wait times than some RL-based methods.

Another relevant observation is the narrower inter-quartile range of CIPHER compared to CIPHER+, implying that the inclusion of future arrival data does not enhance the model’s ability to optimize traffic flow.

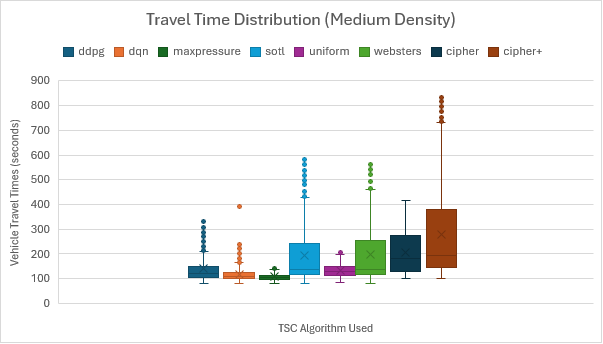


Figure [insert here] showcases the performance of algorithms during a medium density (0.6 second mean inter-arrival rate) simulation. From this we can see DDPG, DQN and CIPHER all achieving superior results at higher congestion levels, in comparison to Websters and SOTL, whilst MaxPressure still serves as the most optimal control method.

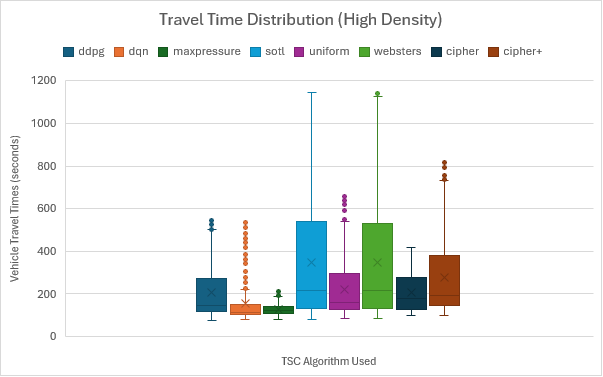


Figure [insert here] showcases the performance of algorithms during a high density (0.3 second mean inter-arrival rate) simulation. The results show that most RL-based methods (DDPG, DQN and CIPHER) are able to outperform Websters and SOTL at higher congestion levels. Interestingly this does not hold true for CIPHER+, which exhibits significantly worse performance than CIPHER, reinforcing the hypothesis that future arrival data is not relevant state information.

The charts also show the Uniform (Fixed Time) controller often being on par with or outperforming the more advanced adaptive methods. It is important to note that distribution of cars in the real-world dataset was moderately symmetrical, and the Fixed time algorithm would not be expected to perform so well when arrivals aren’t evenly distributed across all lanes.

As one final metric of model effectiveness, the environmental impact of each algorithm was assessed by calculating the estimated CO2 emissions. Under the assumption that an idling vehicle consumes 0.5 gallons of fuel per hour [CITE], and each gallon produces 8887 grams of C02, the total emissions per scenario were computed and shown in Table [insert here].

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm Used | Low | Medium | High | Total |  |
| ddpg | 135.602 | 169.929 | 248.1427 | 553.67 | -17% |
| dqn | 166.584 | 141.463 | 187.887 | 495.93 | -26% |
| maxpressure | 127.262 | 129.958 | 155.2297 | 412.45 | -38% |
| sotl | 143.342 | 236.417 | 390.6487 | 770.41 | 16% |
| uniform | 151.781 | 161.169 | 267.0762 | 580.03 | -13% |
| websters | 143.848 | 239.476 | 389.5698 | 772.89 | 16% |
| cipher | 169.99 | 248.51 | 248.5099 | 667.01 | 0% |
| cipher+ | 179.143 | 338.254 | 338.2538 | 855.65 | 28% |
|  |  |  |  |  |  |

Naturally, these results correlate with the box plots observed previously. We can see CIPHER resulting in fewer emissions than Websters and SOTL, whilst being outperformed by MaxPressure, Uniform time control, DDPG and DQN. Cipher+ was the worst performing algorithm in comparison, resulting in 28% more emissions. Combined with the overwhelming data from the boxplots, we can conclude that CIPHER outperforms CIPHER+ across all 3 testing instances, showcasing that future arrival data is not an effective state representation factor for RL based TSC algorithms.

One thing evident across all 3 charts is the superiority of the adaptive MaxPressure algorithm in comparison to all other algorithms shown here. Furthermore, it would be trivial to extend the MaxPressure implementation to account for pedestrians. However, unlike CIPHER, it is still unable to dynamically respond to EMVs and adapt in a way that minimizes their travel times over others.

**TODO**

# Conclusion and Future Directions

From the results presented in this paper, it can be concluded that future arrival data is not a useful vector to include in an RL agents state space. Furthermore, it is evident that whilst RL agents can out-perform some adaptive algorithms, MaxPressure is still superior when using vehicle travel times as a metric.

Notably, an adaptive algorithm’s shortcomings lie in the fact it cannot dynamically respond to EMVs to give priority. This paper shows it is possible to design RL agents capable of handling such events, whilst acknowledging the compromise in-terms of overall performance.

Additionally, we show that LO models are capable of handling the same circumstances, and with the implementation of certain optimizations such as batching, a simplified CTM and discretising time steps, it is possible to explicitly model yellow transitions. This in turn results in more realistic results. Despite these optimisations, the large number of binary variables leads to increased model inference, making real-time deployment challenging.

This paper has set the foundations for several potential future directions. The results show the potential of RL agents against the limitations of the classical adaptive algorithms, whilst acknowledging that they currently fall short. Incorporation of traditional adaptive elements into an RL agent, such as a reward function based on minimizing pressure, could have extreme potential.

The scarcity of real-world training data for RL agents remains a challenge. CIPHER was mostly outperformed by the other RL algorithms studied; this could potentially be due to the more sophisticated approach taken in generating vehicle arrival data. Future work should systematically evaluate various traffic distribution models to determine their effect on RL training and how it impacts the model's overall generalization capability.

Finally, the development of a more comprehensive linear optimization model will help guide and evaluate future RL solutions to the problem. Whilst our LO model provided valuable insights, reducing the reliance on binary variables even further is critical to achieving scalability. Future work could explore other mathematical optimization methods such as dynamic programming, or a heuristic approach.

# Summary and Reflections

Summary and Reflections including a discussion of results in a wider context (considering other work).

o Project management covering the tasks as a part of your work plan and progress as well as how time and resources are managed.

o Contributions and reflections providing the details of your achievements and contributions including innovation, creativity and novelty (if there is any) as well as a personal reflection on the plan and your experience of the project (a critical appraisal of how the project went). This section should also explain how you have considered the “bigger picture” within which your work is situated in terms of LSEPI (refer to the Laws, Social, Ethical and Professional Issues in Projects section above). You should present your reflections on each issue, including why they are (or are not) significant for your project.

Project management:

* Finished as expected, allows me to use my 2 week buffer for improvements.

Contributions and reflections:

* Hybrid workflow allowing a LO model to feed into RL design
* If I were to repeat would start with the LO model

LSPEI:

# Bibliography

Daganzo, C., 1995. The cell transmission model, part ii: Network Traffic. *Transportation Research Part B: Methodological,* Volume 29, pp. 79-93.

Daganzo, C. F., 2005. A variational formulation of kinematic waves: basic theory and complex boundary conditions. *Transportation Research Part B,* 39(2), pp. 187-196.

Gershenson, C., 2004. *Self-Organizing Traffic Lights.* s.l.:arxiv.

Guilliard, I., 2020. *Mixed Integer Linear Programming for Traffic Signal Control.* s.l.:s.n.

Haoran Su, Y. D. Z. J. Y. C. B. D. L. J., 2022. EMVLight: a Multi-agent Reinforcement Learning. *Transportation Research,* Volume 3.

Hua Wei, C. C. G. Z. K. W. V. G. K. X. Z. L., 2019. *PressLight: Learning Max Pressure Control to Coordinate Traffic.* s.l.:Proceedings of the 25th ACM SIGKDD Internatinoal Conference on Knowledge Discovery & Data Mining.

Hua Wei, G. Z. H. Y. Z. L., 2018. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. *ACM SIGKDD International Conference,* Volume 24.

Huang, Z., 2024. Reinforcement learning based adaptive control method for traffic lights in intelligent transportation. *Alexandria Engineering Journal,* Volume 106, pp. 381-391.

Inrix, 2018. *INRIX: Congestion Costs Each American 97 hours, $1,348 A Year,* s.l.: s.n.

Jiahui Yu, K. S., 2022. *Normalization effects on deep neural networks.* s.l.:arXiv.

Kakazu, S. M. a. Y., 1994. Genetic reinforcement learning for cooperative traffic signal control. *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence,* 1(10), pp. 223-228.

Kentaro Wada, K. U. T. T. M. K., 2017. An optimization modeling of coordinated traffic signal control based on the variational theory and its stochastic extension. *Transportation Research Procedia,* 23(2352-1465), pp. 624-644.

Liu, A. J. a. T., 2024. Impact of Reward Function Selection on DQN-Based Traffic Signal Control. *IEEE International Conference on Green Energy and Smart Systems,* Volume 10, pp. 1-6.

Lopez, P. A., 2018. Microscopic Traffic Simulation using SUMO. *International Conference on Intelligent Transportation Systems,* pp. 2572-2582.

M. Wiering, J. V. J. v. V. a. A. K., 2004. Simulation and Optimization of Traffic in a City. *IEEE Intelligent Vehicles Symposium,* pp. 453-458.

M.A.S. Kamal, J. I. T. H. A. O. K. A., 2013. Traffic Signal Control in an MPC Framework Using Mixed Integer Programming. *IFAC Proceedings Volumes,* 46(21), pp. 645-659.

Martin Fellendorf, P. V., 2011. Microscopic traffic flow simulator VISSIM. *Fundamentals of Traffic Simulation.*

Miller, A. J., 1963. Settings for Fixed-Cycle Traffic Signals. *Journal of the Operational Research Society,* Volume 14, pp. 373-386.

Mobin Yazdani, M. S. S. A. B. N. N. J. P. H. P., 2023. Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control. *Transportation Research Part C: Emerging Technologies,* Volume 149.

Panayotis Christidis, J. N. I. R., 2012. *Measuring Road Congestion,* s.l.: European Commision.

Prashanth L. A., S. B., 2011. Reinforcement Learning with Function Approximation for Traffic Signal Control. *IEEE Transactions on Intelligent Transporation Systems,* 12(2), pp. 412-421.

Romain Ducrocq, N. F., 2021. *Deep Reinforcement Q-Learning for Intelligent Traffic Signal Control with Partial Detection.* s.l.:arxiv.

S.M.A. Bin Al Islam, A. H., 2017. Distributed coordinated signal timing optimization in connected transportation networks. *Transportation Research Part C: Emerging Technologies,* Volume 80, pp. 272-285.

Shashank Bharadwaj, S. B. R. M. K. C., 2017. *Impact of congestion on greenhouse gas emissions for road transport in Mumbai metropolitan region,* s.l.: s.n.

Syed Shah Sultan Mohiuddin Qadri, M. A. G. &. E. Ö., 2020. State-of-art review of traffic signal control methods: challenges and opportunities. *European Transport Research Review,* Volume 12.

Tham, S. S. a. C. -K., 2007. SensorGrid for Real-Time Traffic Management. *International Conference on Intelligent Sensors, Sensor Networks and Information,* Volume 3, pp. 443-448.

Varaiya, P., 2013. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies,* Volume 36, pp. 177-195.

Webster, 1958. Traffic Signal Settings.