The Power of Adaptability Applied to Vehicular Traffic Management

The power of adaptability applied to vehicular traffic management

Ruben Y. Terrazas Ruiz

A Thesis in the Field of Information Technology
for the Degree of Master of Liberal Arts in Extension Studies

Harvard University
November 2018

Abstract

Nobody likes sitting in traffic, at least the author of this work does not. In this thesis, we take the question of *how can we improve traffic?* We break it down and iterate on it, finally arriving at the problem of traffic optimization, via the optimization of traffic lights. This is the main driver of this research work. Traffic light control strategies have been previously classified as *static* and *dynamic*. We typically see static controllers in the fixed-cycle or actuated controllers that are widely deployed in the traffic lights of our cities; however, truly smart traffic lights are still not a widespread technology. This work tries to ground some of the research in dynamic traffic light controllers, so that smarter traffic lights become practical and ready for roll-out. We describe a Reinforcement Learning algorithm which we designed, and which only uses information that is local to the traffic light. A key element of this algorithm is the fact that we believe the inputs to it can be obtained by existing traffic intersection technologies, making its deployment feasible. We compare this algorithm against an optimal static traffic control policy, in the pursuit of understanding the strengths of both controller types. Finally, we analyze and present the results of this comparison, and illustrate a set of opportunities that may improve traffic management through different approaches, such as optimal traffic management, following the footsteps of this work or by having drivers collaborate with load balancing.

Dedication

To my wife, Lily. Without your support and understanding, this would not have been possible. This is as much my achievement as it is yours. I love you.

Acknowledgments

First and foremost, I would like to thank my thesis director, Dr. Victor Farutin, for his mentorship in the process of this research endeavor. His mentorship helped me build an appreciation of how the research process can guide your work. This is a lesson I take with me, and I am sure will help me throughout the rest of my professional career.

I would also like to thank my research advisor, Dr. Jeff Parker, who was a great mentor in the discovery process of this thesis.

Table of Contents

List of Figures

Chapter I.

Introduction

This thesis studies the problem of traffic management with a practical perspective in mind, and a preference for solutions that could be applicable today.

We developed a survey of the state of the art in the area of traffic management strategies and traffic simulation systems, given that these two go hand in hand. In order to test their hypotheses, prior researchers in this field have used computer simulations to validate their assumptions, this thesis is no exception.

From our understanding of the state of the art, we developed a hypothesis about the differences between the two main traffic control mechanisms that exist today. We designed a new dynamic traffic control algorithm using Reinforcement Learning, a machine learning technique.

We compare our dynamic traffic controller against a static controller, evaluate the results, and test our hypothesis. Finally, we conclude with a set of ideas and recommendations that we believe can pave the road for future research and further real-world applicability of existing research.

Motivation

Waiting in traffic is one of the most frustrating things in modern life, yet it is something that millions of people have to put up with.

If we want to think about the problem of improving traffic, we need a way to model the problem, and we need to understand the levers are at our disposal. We can think of this as a problem whose goal is the maximization of a network's throughput and in order to do so, we either increase the capacity of the network, or we learn to manage its flow more efficiently.

Increasing the capacity of the network would mean infrastructure changes on cities, which is not always feasible, carries a great cost, and disrupts traffic and people's lives while construction takes place.

Managing traffic flows more efficiently is not a new idea, most of the previous work has looked at traffic lights as the lever that could most easily be used to this end. Given that this is the lever that is most easily available, this work will also be centered around this approach.

In essence, this work began with the goal of understanding the overarching question of *how can we improve traffic?* We recognized the need to evolve this into a more actionable question, and through the initial review of the state of the art, this question evolved to *how can we improve traffic management techniques?* this was still a very high-level question, but it guided us to a clearer path.

As we will see in the survey of the state of the art, the more promising approaches to traffic management have to do with the optimization of traffic light policies, and the two main approaches to managing traffic lights could be classified as static or dynamic policies. If we are to take a step towards improving traffic, through better traffic management, we believe that a good place to start, is by answering the question *can we use computer simulations to understand the strengths of static and dynamic policies?*

The next section has the goal of helping the reader build a mental model of this thesis and will provide an outline of what is presented herein.

Thesis outline

In order to ease the reader into the work that is presented herein, we consider it beneficial to describe at a very high-level, the material presented in each chapter of this thesis.

In Chapter II of this thesis, we present a survey of the state of the art in traffic simulation systems. In this section, we go over the types of traffic simulation systems that exist and the software systems that implement these concepts. We conclude this chapter by deciding the simulation system that will be used herein.

Chapter III presents a survey of the state of the art in traffic management strategies, focusing on traffic control by means of traffic light policies. In this chapter we introduce some of the earlier work in this area, which studied the optimization of fixed-cycle controllers, also called static policies herein. We then move on to more current techniques, which use several mechanisms to try and optimize traffic light policies dynamically. We conclude this chapter by stating the hypothesis evaluated in this thesis.

In Chapter IV we introduce the simulation configurations that will be used to test the hypothesis. We use one traffic grid and two different traffic configurations. One of the traffic configurations represents *normal traffic patterns* for the intersection, while the other one represents a drastic change in these patterns, we usually call this configuration the *unexpected traffic patterns*.

In Chapter V we analyze various metrics that are available through SUMO, the simulation system we use to test our hypothesis. We conclude this chapter by identifying the most useful metric for the analysis performed on this thesis, and which will be used to test our hypothesis.

Chapter VI presents a dynamic control mechanism designed as part of this thesis. This is an autonomous control agent that uses Reinforcement Learning to adaptively manage traffic light policies. We begin the chapter by presenting the Reinforcement Learning algorithm, as well as the process we took to train this algorithm. It is important to clarify that this algorithm is trained on the normal traffic patterns alone.

In Chapter VII we compare a static control policy and our dynamic control policy. The static control policy has been crafted for optimality when exposed to the *normal traffic patterns*. As mentioned earlier, the Reinforcement Learning was trained with this traffic configuration. Up to this point, none of these control policies has ever been exposed to the traffic configuration we refer to as the *unexpected traffic patterns*. This is how we measure the adaptability of these techniques.

Chapter VIII of this thesis evaluates the results presented in the previous chapter and draws a conclusion based on this data, and our hypothesis.

Finally, in Chapter IX we use our conclusions, along with our understanding of the state of the art to propose new ideas that could take this area of research forward.

A mental model for the problem

It is also beneficial to ease the reader into a more solid understanding of the general structure that this problem takes when approached from the perspective of managing flows. In this section, we introduce some of the concepts that will be used throughout this thesis.

Provided that there are no road closures or accidents blocking lanes, the main point of contention is usually an intersection. Identifying this main point of contention can help us reduce the search space we operate on when exploring prior work.

We believe it is useful to have an understanding of the types of intersection that exist, given that this will help us build a better understanding of the tools at hand. MASSDOT (2006) defines four types of intersections:

- Simple: These are four-way intersections with no additional lanes for turns.
- Flared: These intersections have additional sections near an intersection's approach, to help alleviate the congestion that may result from vehicles preparing to turn.
- Channelized: These intersections use markings to designate intended vehicle paths. The most common use for these is right turns, usually accompanied by an auxiliary right turn lane.
- Roundabouts: A channelized intersection, with a one-way traffic flow, that revolves around a circular island at the center.

When discussing guidelines regarding when to choose a signalized intersection or a roundabout, MASSDOT (2006) recommends the use of roundabouts as a traffic calming mechanism and signalized traffic intersections as a control mechanism for high traffic volumes.

This recommendation fits really well with the work that previous researchers have done in the field, as they have usually explored the optimization of traffic through the use of *traffic lights*. Traffic lights, intuitively, are the most configurable mechanism that is used to manage the flow of traffic, and researchers have studied them in order to understand how to better control them to increase throughput on a traffic network with limited resources.

Liu, H., Oh, J.-S., & Recker, W. (2002) classify the traffic controllers that are used in traffic lights into three categories: *Pre-timed, actuated and traffic responsive*.

Pre-timed controllers are the oldest and simplest kind of traffic light control strategy. In pre-timed controllers, the occurrence and duration of all timing intervals are pre-determined and fixed, pre-timed controllers have no way of reacting to its environment, the cycles in the traffic light behave the same with high demand and no demand.

Actuated controllers improve on pre-timed controllers by reacting to the input of sensors, typically induction loops. These systems are configured with a fixed-cycle policy; however, the controller can optimize according to the local traffic conditions by skipping a phase if there is no demand in that direction. The problem with these controllers is adaptability, as they do not perform any systematic optimization. Most modern-day intersections are actuated controllers.

Traffic responsive controllers are the most complex kind, and they improve on actuated controllers by creating a signal timing plan that responds to current traffic conditions measured by a detection system and attempts to optimize the control policies dynamically.

Usually, a fixed-cycle controller is configured through a cycle in which all of its traffic gets a green light at some point. The time in which a direction has a green light is defined as the split time. Directions that carry most of the traffic flow can be given preference by adjusting the split time of this direction. The complete duration of a cycle is called the cycle time. The offset of a cycle defines the starting time of an intersection's cycle relative to other traffic lights. Cooperation between different traffic intersections is achieved by adjusting the offset, with the goal of creating green waves, a period of time in which a series of traffic lights allow for continuous traffic flow over several intersections.

Traffic responsive controllers, use inputs from the environment to optimize the flow of traffic while ideally maintaining green light allocation fairness, that is, no direction should be blocked because of low demand. Most of the modern research in this area aims to improve adaptive control strategies to manage traffic more efficiently.

This thesis will study different adaptive traffic control algorithms that leverage traffic lights as the mechanism to control traffic flow. Most of the prior work uses some type of artificial intelligence techniques in order to optimize the policies of traffic responsive controllers.

Previous researchers have encountered the need to evaluate the performance of different traffic control strategies, and this has typically been done through the use of *traffic simulations*. We can think of both *traffic lights* and *traffic simulations* as the main building blocks for most of the research on this field. The use of computer simulations to understand the behavior of traffic management strategies is not a new idea, Webster, F. V. (1958). pioneered the use of simulations and developed the formulas that are used to determine the length of a fixed-controller cycle and cycle splits.

In the next section, we will review some traffic simulation systems that exist today and use this analysis to understand which one will be the most suited for this work, after all, we must also obtain data to test our hypothesis.

Chapter II.
Survey of simulation systems

Most of the prior research uses computer simulations to obtain data about the behavior of different traffic control policies and evaluate their behavior against a hypothesis.

Our work will also leverage simulations, as they have proven to be an efficient mechanism in prior research. We will use the knowledge that we summarize below to identify the system that is the closest fit to our requirements.

In order to consider that a simulation system is a good fit, it must support all of the following: *the flexibility to configure traffic grids and the vehicles that will traverse them, their general routes, additionally, we must also be able to configure traffic lights, and their policies (static and dynamic), and we will need visibility into the metrics for all of the vehicles that traversed the grid during a simulation*.

Two publications stand out as the most comprehensive reviews of traffic simulation systems to date. The first one, published by Kotusevski and Hawick (2009) did a very thorough evaluation of simulation systems, was a well-focused survey, with the clear goal of understanding the state of the art in the area of traffic simulation. The second one, a survey done by Mahmud and Town (2016, June 15) analyzed traffic simulation systems with the goal of understanding which tool worked best to evaluate the energy requirements of electric vehicles.

We will combine the contributions done in both publications; however, we will add and clarify, whenever we feel that we have obtained more complete information about the system at hand.

Mahmud, Town (2016, June 15), give us the most comprehensive categorization of traffic simulation models, and they classify traffic models in four main categories: *microscopic*, *macroscopic, mesoscopic and metascopic*.

By combining the classification and contributions of Mahmud, Town (2016, June 15) with TrafficSimulation-Wikipedia (2018), we can better describe these categories by saying that *microscopic* models focus on the dynamics of individual vehicles, while *macroscopic* models ignore the dynamics of individual vehicles, and attempt to model traffic as a compressible fluid with the main properties of flow, density, and speed.

Mesoscopic models analyze the transportation dynamics of small homogeneous groups, such as vehicle platoons. Unfortunately, Mahmud, Town (2016, June 15) do not define what they mean by *metascopic* models and provide no examples of this type of model.

The *microscopic* and *macroscopic* models are the most relevant models to this study; hence they will be the main focus of this survey.

It is worth noting that most of the existing research in this area uses microscopic simulators to test the effectiveness of traffic control algorithms. The reason for this is that by modeling the dynamics of individual vehicles, the microscopic model lends itself very well to the approach of controlling traffic through traffic light policies.

SUMO

The simulation system called SUMO stands for Simulation of Urban Mobility and is an open source, microscopic traffic simulator. SUMO is portable across different operating systems and development is still very active, with the last recorded change to the code-base happening on July 2018, according to SUMO-GitRepo (2018).

When Kotusevski, Hawick (2009) evaluated SUMO, it did not provide a graphical editor, out of the box, and this was mitigated by the existence of an open-source editor; however, nowadays SUMO does provide a graphical editor, called Netedit. Netedit is a powerful and easy to use application that allows users of SUMO to build fairly complex traffic grids.

The native way that SUMO users define traffic patterns, is by specifying their routes using an XML schema. This mechanism enables them to define individual vehicles, repeated vehicles also called flows. Vehicles traverse the network through pre-specified routes, or through incomplete routes, specified by an origin and a destination.

Kotusevski and Hawick (2009) mention that traffic lights can be configured manually through XML in SUMO but do not mention SUMO's support for dynamic traffic controllers, which SUMO does support through the use of its API, called TraCI which stands for Traffic Control Interface.

Results from a simulation that ran in SUMO are presented in an easy to use XML file. These results include vehicle level results such as trip length, trip duration, emissions, times the vehicle had to stop, among many others as further specified in SUMO-ProjectPage (2018)

Finally, SUMO includes other complex features such as the specification of vehicle length, vehicle rerouting, street closures, and road access restrictions as detailed in Kotusevski and Hawick (2009)

Quadstone Paramics

Paramics is a commercial microsimulation software that is capable of emulating complex real-world traffic and transportation problems, Mahmud and Town (2016, June

15) observe that Paramics is mostly used for traffic engineering, for example, in the design of highways, freeways or complex intersections. Unfortunately, PARAMICS, like all commercial systems evaluated, is not a cross-platform system and only works on Windows.

Traffic networks are defined through an automatic network generation wizard; however, Kotusevski and Hawick (2009) were unable to confirm this, given that the demo version only generated a simple network. Traffic patterns are defined using origin-destination matrices.

Kotusevski and Hawick (2009) found PARAMICS graphical representation to be the best, as it allows their users to configure buildings, vehicles, pedestrians creating a more realistic-looking simulation when compared to other systems.

While Kotusevski and Hawick (2009) do not mention the ability to control traffic through external controllers, prior work has used PARAMICS for their simulations, therefore this must be possible.

Transport Simulation Systems - Aimsun

According to Kotusevski and Hawick (2009), Aimsun is a microscopic commercial simulation system that integrates three types of transport models: static traffic assignment tools, a mesoscopic simulator, and a microscopic simulator.

Additionally, Mahmud and Town (2016, June 15) identify that some common applications of this system include the analysis of bus rapid transit, transit signal priority assessment, highway infrastructure impact analysis, and feasibility analysis of high occupancy vehicle lane tolling.

Traffic grids are configured by manually drawing them using the provided graphical editor, and traffic patterns are defined through origin-destination matrices. Aimsun has an API that can be used to integrate dynamic traffic control algorithms in a simulation.

Trafficware SimTraffic

This is a commercial microscopic simulation package that supports the configuration of traffic grids through a graphical editor in the full edition. Kotusevski and Hawick (2009) were not able to specify traffic patterns in the demo version but were confident that this must be supported in the full edition. It is unclear if SimTraffic has an API which would allow for custom traffic control strategies.

Additionally, Mahmud and Town (2016, June 15) describe its ability to model emergency vehicles, and their interaction with the surrounding traffic, and the fact that it supports a central management module that monitors and optimizes the traffic control signals. Given these capabilities, it appears that this system is in direct competition with Aimsun, and has a similar target audience, such as city planners.

TSIS-CORSIM

Kotusevski and Hawick (2009) refer to this system as CORSIM-TRAFVU; however, it appears that the name of the full application suite is CORSIM, as stated in TSIS-CORSIM (2018).

CORSIM is a commercial microscopic traffic simulation system. Kotusevski and Hawick (2009) could only test this simulator under a pre-defined traffic grid as the demo version did not allow them to configure more complex networks.

None of the previous publications describe CORSIM's traffic pattern definition functionality; however, official documentation TSIS-CORSIM (2018), shows that it supports origin-destination matrices. It is unclear whether there is API support.

Mahmud and Town (2016, June 15) describe CORSIM's main use-cases as traffic management through the configuration and coordination of street signal timing, freeway and bus lane design, incident detection and management, truck weight station and toll plaza design. Given this information, this tool seems to be directed towards the traffic engineering departments of cities, just like Aimsun and SimTraffic.

Green Light District

This is a home-brewed discrete step microsimulator build by Wiering et al. (2004) to compare their reinforcement learning algorithms against fixed-cycle controllers. It is unclear how much theory of traffic simulation was used in the development of this simulator as it seems the priority was the experimentation of new reinforcement learning techniques and not the advancement of simulators in this area.

Green Light District allows for the configuration of arbitrary traffic networks and traffic patterns and monitors traffic statistics such as average waiting times. Arbitrary traffic control strategies can be built and compared on Green Light District as it is an open-source system, built on Java. Unfortunately, this is not a well-maintained project, having its last code commit in 2005 GreenLightDistrict-Sourceforge (2018).

Ad-hoc discrete step simulation frameworks

Finally, there exists a number of discrete simulation frameworks that could lend themselves to the development of a traffic microsimulator. Some of these frameworks are general purpose simulators such as SimPy (2018) and ArenaSimulation (2018) that could be used to build ad-hoc microscopic simulators; however, this approach will not be pursued given the complexity in building a high-quality traffic simulator.

The simulation system decision

As stated previously, the goal of this survey was not only to understand what is available but to make a decision regarding the simulation system that will be used in this work.

We believe that SUMO is the most sensible choice of a simulation system for this work, as it meets all of the criteria we need in order to measure various traffic control strategies; moreover, it is an open source system, which means that others will be able to build on top of this work and use SUMO as their simulation system if they so choose.

We will use the next chapter to provide a survey of the prior research in the area of traffic light optimization. Additionally, we will use this information to formulate the hypothesis that will later be evaluated in this thesis.

Chapter III.

Prior research in the area of traffic management

In this section, we survey the existing research in the area of traffic management, from optimization of fixed-cycle controllers to dynamic control policies. From our high-level questions, that originated this work and a new understanding of the state of the art in this domain, we will also derive the hypothesis of this thesis.

Fixed-cycle controllers

This is probably the first alternative towards improving traffic signals. Webster (1958) pioneered the use of computer simulations to derive a set of formulas that can be used to configure an intersection's cycle with the goal of minimizing overall delay. These formulas are still used to calculate the optimal cycle and cycle-split length for fixed-cycle controllers.

Ohno, Mine (1973) expand on the work done by Webster and come up with an alternative technique based on linear programming. Additionally, they show that this approach works on more complex configurations, given that Webster's work is challenging to apply in the context of phases that enable various directions at once.

We believe that the biggest limitation of fixed-cycle controllers is their lack of adaptiveness, not their lack of optimality for known situations. Many events like accidents, construction, and road repair can change traffic patterns and submit fixed-cycle controllers to circumstances for which its policy is not optimal.

Expert systems

Expert systems attempt to emulate the decision-making process of a human expert. Findler, Stapp (1996) explored the use of expert systems to address the problem of increasing traffic throughput.

Their expert system proposal consists of a distributed system where each intersection is controlled by an identical expert system that can communicate with the four adjacent intersection controllers. Each intersection controller may have a different knowledge base, given the setup of the intersection and the traffic patterns around it.

Their prototype is a simplified version of the expert system they propose, the most important simplification is the lack of conflict resolution, as this prototype would always pick the first rule that matched the current conditions.

They used computer simulations to test the effectiveness of their approach; however, they do not specify the type of model they use. One can assume that they used a microscopic model, given that they describe some properties of the vehicles in this simulation, for example, the measure they use to compare different rule bases is mean travel time.

The performance of two versions of this approach were compared: the original expert system, and a version where they optimized the system's parameters through hill-climbing apriori. They obtained a 36% improvement after hill-climbing optimization, however, it is unclear how this would compare against other types of controllers, such as fixed-cycle controllers.

Wen (2008) also uses a microsimulation, built on Arena, a commercial simulation software, to test the effectiveness of an expert system-based traffic control system. In this research, Wen (2008) uses interarrival times, a value that is used in queuing theory to analyze systems that involve waiting time. Interarrival time in this context is the time between the arrival of one vehicle to an intersection and the next. Simulations are used to compare different configurations of this system, and unfortunately, there are no comparisons against alternative approaches, such as fixed-cycle controllers.

Neither of these controllers based on expert systems supports the ability to learn new rules from prior experience. Findler, Stapp (1996) mention the intention of adding this capability; however, it is unclear how, given that they did not publish additional material that would immediately generalize to the problem of traffic management.

Expert systems, as evaluated, suffer from the same problem as fixed-cycle controllers, the lack of adaptability.

Prediction-based optimization

Liu, Oh, and Recker (2002), propose a responsive system that uses predictions to estimate the delays vehicles will encounter and adjust the control policies with the goal of minimizing total delay while keeping the green light allocation fair. It is unclear how they generate these predictions, as they do not mention if they use regression or some other technique for this.

The system described in this work relies on data collected from advanced inductive loops used for delay estimation. Delay estimation is done through vehicle identification, they propose using advanced inductive loop detectors in order to record a vehicle's waveform and use that as it's identifying signature. The system calculates a vehicle's delay through vehicle identification at intersections, a vehicle's delay is the time difference between the time the vehicle left an intersection to the time it arrived at the next intersection. They do not elaborate on whether this data is shared among peers or if it is stored at a central location.

Vehicle identification through inductive loops is the key component of their system. When elaborating on the feasibility of this feature, they explain how to distinguish similar vehicles, despite the fact that they may generate similar waveform signatures. They claim that unique attributes such as the number of passengers, their weight, and their luggage modify a car's waveform, making it unique; however, it is unclear if the probability of two distinct vehicles being identified as one has been determined.

This work does not address edge cases such as vehicles with variable passengers, which would be the norm in public transportation. Additionally, this work does not mention any privacy concerns, as this system is tracking individual vehicles as they move along a city.

Their system has two objective functions that it attempts to minimize. The first one is the total delay, described as the sum of the delay of all vehicles across all movements, for the duration of the simulation. The second objective function is system fairness, defined as the standard deviation of the delays for all movements, for the duration of the simulation. Their algorithm can optimize an intersection's cycle length and phase split.

They tested their approach on a four-way intersection using Paramics, a commercial microscopic traffic simulator, and they compare six traffic control strategies, a typical pre-timed controller, a typical actuated controller, which serve as baselines and both pre-timed and actuated controller optimized based on average and total delay.

The controller that achieved the most throughput was their online pre-timed control based on average delay (OPA), allowing 11,284 vehicles through the intersection in the span of two hours, this was also the fairest of the controllers tested, achieving a standard deviation of delays of 30.8 seconds.

The classic non-optimized controllers, the pre-timed controller (PTC) and the actuated controller (AC) had a throughput of 11,072 vehicles and 10,772 vehicles, respectively. They had a standard deviation of delays of 35 and 48 seconds respectively. Not all online controllers beat the classical approaches, the online pre-timed control based on total delay (OPT) had a throughput of 11,057 vehicles during the simulation, 15 vehicles less than the pre-timed controller, and a standard deviation of 40.4 seconds, 5.4 more than the classical pre-timed controller. Liu, Oh, and Recker (2002) do not mention of any theories or insights based on these results, but we suspect that the controllers that used average values in their objective functions achieved better performance because they were less sensitive to outliers.

Evolutionary Algorithms

Sanchez-Medina, Galan-Moreno, and Rubio-Royo (2010) use genetic algorithms to evolve the traffic control policies of their controllers. Chromosomes in their traffic control algorithms are represented as a sequence of states that a traffic controller has at a given intersection. They have four variants of this algorithm, with the key difference being the fitness functions, them being:

- Number of vehicles that exited the traffic grid during the simulation (NoV). The objective is to maximize this function.

- Mean Travel Time (MTT). The objective is to minimize this function.
- Time of Occupancy/State of Occupancy (TOC/SOC). Through the simulations they obtain the average TOC, which has the same value as the average SOC, hence they named this function TOC/SOC.
- Global Mean Speed (GMS). The objective is to maximize this function.

Their algorithm has an elitist selection strategy, that is for every generation, they select the best two individuals, the next generation is created by crossing individuals from the best fitness subset. Their algorithm uses a standard two-point crossover operator, that selects two random points at which to cut a parent chromosome into three pieces, they then interchange the central chunk. Finally, when an individual is chosen to be mutated, the value stored at a random position of the chromosome is changed.

The simulation on which they run these algorithms is a fairly complex one, they model a real-world section of a city, the "Almozara" a district in Zaragoza, Spain. They do not compare their approach against fixed-cycle controllers but make reference that prior literature has compared adaptive controllers against fixed cycle controllers and found that adaptive controllers were more efficient. We believe that they assumed their controllers would outperform fixed-cycle controllers, and thus made no attempt to test this.

When evaluating their results, Sanchez-Medina, Galan-Moreno, and Rubio-Royo (2010) do not focus too much on throughput evaluation or fairness of each approach, they focus on fitness function behavior, as a function of the number of cars, and the relationship between each fitness function. The fitness functions that showed consistent improvements as the number of vehicles increased were: The number of vehicles (NoV) and mean travel time (MTT).

Through their simulations, Sanchez-Medina, Galan-Moreno, and Rubio-Royo (2010) found that an uncongested traffic network is unlikely to be optimized, given that the output of trained and untrained models was similar. We believe that this statement is a simplification, it is not that uncongested networks cannot be optimized, but the benefits of optimization will not be seen as easily, and they will hold less value.

This publication was both a systems design, and a genetic algorithms research endeavor, and we believe that the fact that two separate threads were being discussed limited the ability of Sanchez-Medina, Galan-Moreno, and Rubio-Royo (2010) to pursue some of these questions in depth. For example, we believe that this would be a promising technique to achieve coordination between intersections, and achieve green waves; however, we have no way of knowing the position of the authors, given that this topic was not explored.

Reinforcement Learning

Reinforcement learning is probably the most popular technique in current literature related to the topic of traffic optimization. A highly referenced paper in this area by Wiering et al. (2004) studies a co-learning multi-agent Reinforcement Learning algorithm with uses road-user value functions to determine optimal decisions of a traffic light.

The decisions of a traffic light are based on the cumulative vote of all vehicles that stand at a junction, each vehicle votes with its estimated advantage of setting the light to green in a direction.

We say that their algorithm is a co-learning reinforcement learning agent, given that the quality estimates of a decision are tracked at the road-user level, and these estimates inform a traffic light by having vehicles vote through the sum of expected benefits of a traffic light enabling a direction.

They do not clarify how road-user value functions could be implemented in real life; however, the work from Liu, Oh, and Recker (2002) regarding advanced inductive loops sheds some light into how a real-world implementation of this approach would be feasible.

This work is tested on a home-brewed, discrete-step microsimulator called Green Light District. They compare several strategies against fixed-cycle controllers, however, it is unclear what policies they used for fixed-cycle controllers and whether the policies in these controllers were optimized somehow. Their results favor dynamic reinforcement learning based controllers, this can be seen when comparing the average waiting times, the best adaptive controller has an average waiting time of 2.67 time steps, whereas the best fixed-cycle controller has an average waiting time of 3.57 time steps.

They acknowledge the fact that their work still needs improvement in the setup of green waves, a series of traffic light states that allows the uninterrupted flow of traffic in a given direction across several intersections. We believe that if their algorithm made decisions that spanned multiple time steps instead of a single one, this could be achieved naturally. Upstream intersections sending continuous traffic to downstream intersections would allow these to optimize and allocate green light intervals accordingly.

Deep Reinforcement Learning

This is the latest trend and a very promising hybrid approach. Li, Lv, and Wang (2016) used deep neural networks and Q-learning, an approach known as Deep-Q networks, pioneered by DeepMind (2014) to build a traffic control algorithm.

The purpose of the deep neural network, in this case, is to compute the Q-value that will be used by the reinforcement learning algorithm known as Q-learning, a model-free reinforcement learning algorithm.

They use Paramics, a commercial microsimulator on a traffic grid that is made up of just one four-way traffic light and compare their algorithm against a standard reinforcement learning algorithm, using average delay and number of fully stopped vehicles as metrics for this comparison.

Through their simulations, they found that the deep reinforcement learning algorithm performs better than the standard reinforcement learning algorithm, achieving an average delay that is 14% better, while also generating shorter queues. When compared to the standard reinforcement learning algorithm, the longest queue observed for the deep reinforcement learning controller was made up of 60 vehicles, compared to 70 for the standard reinforcement learning controller.

Unfortunately, it is unclear how the Deep Reinforcement Learning algorithm would perform when compared to a fixed-cycle controller. It is also unclear how this algorithm would perform in more complex settings as they do not explore other types of traffic grids.

Summary and formulation of the hypothesis

In this section we surveyed the methods others have explored in the area of traffic control through traffic lights, from its origins in the study of fixed cycle controllers, and the evolution of traffic control strategies, in the search for more adaptive traffic lights.

As we observed, researchers have tried several strategies to improve the state of the art, expert systems, as explored through the work of Findler, Stapp (1996) and Wen (2008), while expert systems improve on fixed-cycle controllers, their lack of adaptiveness limits their applicability.

Prediction-based systems, as explored in the work of Liu, Oh, and Recker (2002) is a promising area of research; however, we believe that the privacy concerns that would arise from these methods would need to be addressed in order to achieve widespread acceptance. Their work is not without merits, and we can see that some of the ideas in this work could enrich the work done by Wiering et al. (2004), given that both publications converge on the idea of vehicle-level estimation of delays.

Evolutionary algorithms, as explored in the work of Sanchez-Medina, Galan-Moreno, and Rubio-Royo (2010) is probably one of the most promising technique in the pursuit for well-coordinated traffic lights. Their algorithm's potential in this area stems from the fact that it operates over the full traffic grid, as their chromosome encodes the policies off all intersections.

Wiering et al. (2004) explored the use of Reinforcement Learning by using a model in which vehicles provided an estimation of their waiting times and learned to derive a quantitative estimated benefit from a traffic light's state. This information influenced the decision of a traffic light, through a voting system In this system, votes represent the estimated benefit a vehicle would get if a given direction were enabled, the direction with most expected benefit is then enabled. While we may not believe that this is a feasible approach at the moment, this work proved that Reinforcement Learning is suited for this control problem.

We believe that the Reinforcement Learning family of algorithms stands as the most well suited to this type of problem, primarily because it was designed for dynamic control problems, with no need of human supervision. This was better stated by the founding fathers of Reinforcement Learning, Richard Sutton, and Andrew Barto.

> Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. Sutton, Barto (1998).

Li, Lv, Wang (2016) explored the use of Deep Reinforcement Learning, a technique that combines Deep Neural Networks and Reinforcement Learning which was pioneered by Deep Mind, now a Google company. In this work they found that a Deep Reinforcement Learning outperformed typical Reinforcement Learning based traffic controllers; however, one of the questions that remain open from the work by Li, Lv, and Wang (2016) is how the Reinforcement Learning algorithm encoded state and rewards, given that this is critical to the performance of a Reinforcement Learning algorithm, as described in one of Deep Mind's first publications, *Playing Atari with Reinforcement Learning (2013)*.

> Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning (RL). Most successful RL applications that operate on these domains have relied on hand-crafted features combined with linear value functions or policy representations. Clearly, the performance of such systems heavily relies on the quality of the feature representation. (Mnih et al., 2013, p. 1)

We believe that Reinforcement Learning is well suited for this problem if we assume that inputs are obtained by inductive loops, and we think that Deep Reinforcement Learning would be very well suited whenever we want to derive the state from an image, as was explored in Mnih et al. (2013) in the context of building unsupervised agents that played Atari.

After reviewing the prior work, we are better equipped to answer the question posed at the beginning and state our hypothesis.

When faced with the question *Can we use computer simulations to understand the strengths of static and dynamic policies?* We believe that the answer is yes; moreover, we believe that dynamic policies will outperform static control policies due to their adaptability. This is the hypothesis of this thesis, dynamic control policies are a better traffic control mechanism than static control policies when presented with changing traffic patterns, due to their adaptability.

This does not mean that static policies are without merit, assuming complete knowledge of a traffic configuration, an optimal static policy could be configured for optimality, and in fact may outshine dynamic policies; however, we believe that this assumption is unrealistic.

In Chapter VI, we explain our implementation of a dynamic traffic control agent that uses Reinforcement Learning. In Chapter VII we compare this against an optimal static control policy. Finally, in Chapter VIII the results are analyzed, and our hypothesis is evaluated.

In the next chapter, we will explain the configuration for the simulations that will be used for these comparisons. We provide the reader with an understanding of the traffic grid, the traffic patterns and how we use the SUMO traffic simulator.

Chapter IV.

Simulation configuration

We use simulations to measure how well a traffic control policy manages traffic. In order to run these simulations, we use an open-source traffic simulation suite called SUMO (Simulation of Urban Mobility). SUMO's many capabilities allow us to model complex traffic networks, the vehicles that flow through it, and the conditions that these flows have. In this section, we will give an overview of how SUMO works, and how it will be used in this work.

An overview of SUMO's traffic control mechanisms

SUMO's default mechanism for controlling the behavior of traffic lights is pre-defined policies. Policies are pre-configured in an XML file and go unchanged throughout a simulation.

A traffic light's cycle has timed phases that control the behavior of the intersection. For example, on a four-way intersection, the phases may enable northbound traffic from the south, then signal that this direction will no longer have priority through a yellow light, finally disabling this direction. After that, all traffic will be stopped for some time until the intersection clears, next we would have enabled eastbound traffic from the west and repeat a similar set of phases.

SUMO also allows developers to interface with its simulations through their API, called TraCI, which stands for Traffic Control Interface. Through this API, developers can obtain information from the simulation environment, for example, if the traffic light is equipped with detectors, TraCI can be used to obtain the number of vehicles that are waiting and the mean speed of vehicles that are within range of a detector.

Developers can also modify the state of the traffic lights by enabling whatever phase they desire, for as long as they want. This is a powerful but dangerous capability, given that software defects can starve important directions of green lights, and preclude the simulation from concluding, which would require the user to manually terminate the process.

The traffic grid

All of the simulations in this work take place on the same type of intersection, a four-way intersection, with vehicles arriving on three-lane roads, from the North, South, East, and West.

All valid directions in a four-way intersection are possible. The illustration below represents such an intersection.

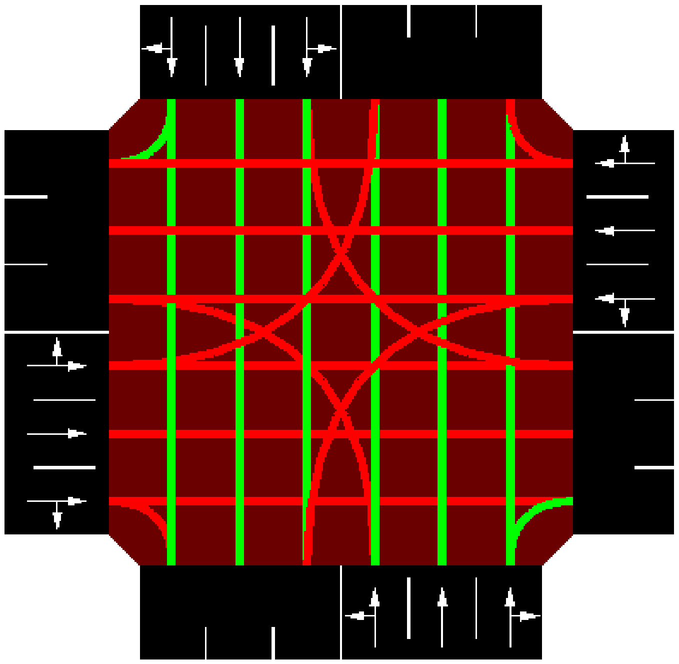


Figure 1: Traffic grid used in this work.

A traffic grid made up of four sources, four destinations, one four-way intersection, with traffic flowing on three lanes in each direction.

The traffic grid above not only shows the directions that are possible, but it also highlights a sample configuration in which the *North to South* and *South to North* directions are enabled at the same time. Paths for these directions are highlighted in green. All other paths that are valid, but not enabled in this specific setting are highlighted in red.

The traffic flows

There are two types of traffic patterns that will be used throughout this work, the first of these traffic patterns represent expected traffic for the intersection, we can think of this as the rush hour traffic that would happen normally on a downtown intersection, the second traffic pattern represents an unexpected traffic pattern, which would stand as a change in the vehicular patterns caused by an accident, construction, or any other type of event that can temporarily affect how many vehicles approach an intersection.

In the first scenario, the normal scenario, all vehicles are generated evenly throughout the first 200 time steps of the simulation, this is to simulate the contention that occurs during rush hour traffic. Traffic flows have the following volumes: *North to South with 800 vehicles, East to West with 600 vehicles, South to North with 400 vehicles, West to East with 200 vehicles*.

In the second scenario, representing unexpected traffic patterns, all vehicles are also generated during the first 200 time steps, however, new directions are explored, and some directions have less traffic volume than they used to have.

The *North to South* direction, which used to be the direction with the most traffic now has only 100 vehicles flowing through it. This change makes this the direction with least demand.

The *East to West* direction, which was the second most popular direction, now has half the traffic it used to have, with only 300 vehicles flowing through it.

The *South to North* direction's volume is also reduced by half, with 200 vehicles.

Conversely, the *West to East* direction's volume is doubled, with 400 vehicles now flowing in this setup.

Additionally, left turns are introduced, with the *North to East* direction having 500 vehicles, and the *South to West* direction having 700 vehicles. Notice that overall, the number of vehicles is almost the same in both scenarios, with only a 10% increase in the second scenario, going from 2,000 to 2,200; however, in the second scenario we introduce left turns and make them the directions of the heaviest traffic flow.

Having provided the reader with an understanding of how we have configured our simulations in SUMO, it is time to present our analysis of the metrics SUMO provides. We will use this to select best metric to will test our hypothesis with.

Chapter V.

Understanding SUMO metrics

SUMO emits multiple metrics per trip in a simulation, some are qualitative and describe interesting properties such as whether or not a vehicle was re-routed. Some other metrics are quantitative and represent measurable properties from each vehicle that traversed the traffic grid during the simulation, such as emissions, route length, trip duration, time loss, departure delay.

The goal of this section is to understand how a policy's optimality is reflected in the metrics we use. To this goal, we will run simulations on three different static traffic control policies designed with varying degrees of optimality. Two control policies stand at the opposite ends of the optimality spectrum, one of them being clearly optimal and the other clearly suboptimal for the given traffic pattern. We will compare metrics obtained during the simulations and use these comparisons to decide which shall be used for further comparisons presented herein.

These three types of traffic controller will be tailored to have a specific behavior when exposed to the first traffic pattern described in the previous section, which represents normal rush hour traffic on a downtown intersection. The reader may remember that we had defined three types of traffic controllers in the introduction, *pre-timed, actuated and traffic responsive*. The types of controllers that we refer as static controllers in SUMO simulations correspond to the *actuated controller* variety. They have a fixed cycle but can skip phases if the traffic enabled by that phase has no demand.

The first controller, is explicitly designed to have very poor performance when exposed to this traffic pattern and allocates preference to directions in the inverse order of their traffic flows, for example, the direction with the least priority is the North to South direction, which under this traffic patterns has the highest number of vehicles traversing the grid.

The second controller is designed to perform an average job at managing traffic through this intersection, for example, this controller takes advantage of parallelization of non-competing directions, such as North to South and South to North, but it does not do it enough. This controller will also allocate some time to some phases that are not heavily used under this setting, such as North to South and East.

Finally, the third controller is designed to be the most optimal controller for the traffic patterns. Given that we have full knowledge of the traffic patterns, this controller allocates time according to the flow ratios and takes full advantage of parallelizable directions. The direction with highest demand will be enabled for the longest time by this controller. Later in this work, we will use this controller as a measure of optimality, when evaluating dynamic traffic control methods.

We expect that trips that occur while the optimal policy is managing traffic will have a better performance overall; therefore, this exercise will help us validate this assumption and allow us to select metrics with which we intend to measure optimality in traffic control policies.

Understanding metric sensitivity

When we talk about metric sensitivity, what we mean is that a metric must react to the optimality of a traffic control policy. For example, trip duration is a metric that intuitively is expected to increase for non-optimal traffic control policies and will decrease with more optimal traffic control policies. A metric that does not behave differently for two control policies is not a useful metric and will not be used in our analysis.

The following five sub-sections describe the SUMO metrics *trip duration, departure delay, time loss, carbon dioxide emissions and wait steps*. These metrics were chosen because they are easily interpretable and applicable to the physical reality. For each of these, we provide a brief description and illustrate their behavior for the three static policies that were evaluated across a total of 20 simulations per policy.

Trip duration – Metric sensitivity

This is the total time that a vehicle takes to traverse its full route. In the context of SUMO, this is measured as the sum of time steps from the time the vehicle was inserted into the simulation to the time the vehicle arrived at its destination.

In the diagram below, we can observe that trip duration is sensitive to policy changes, and the results behave as one would expect, the best traffic control policies will result in a shorter trip duration.

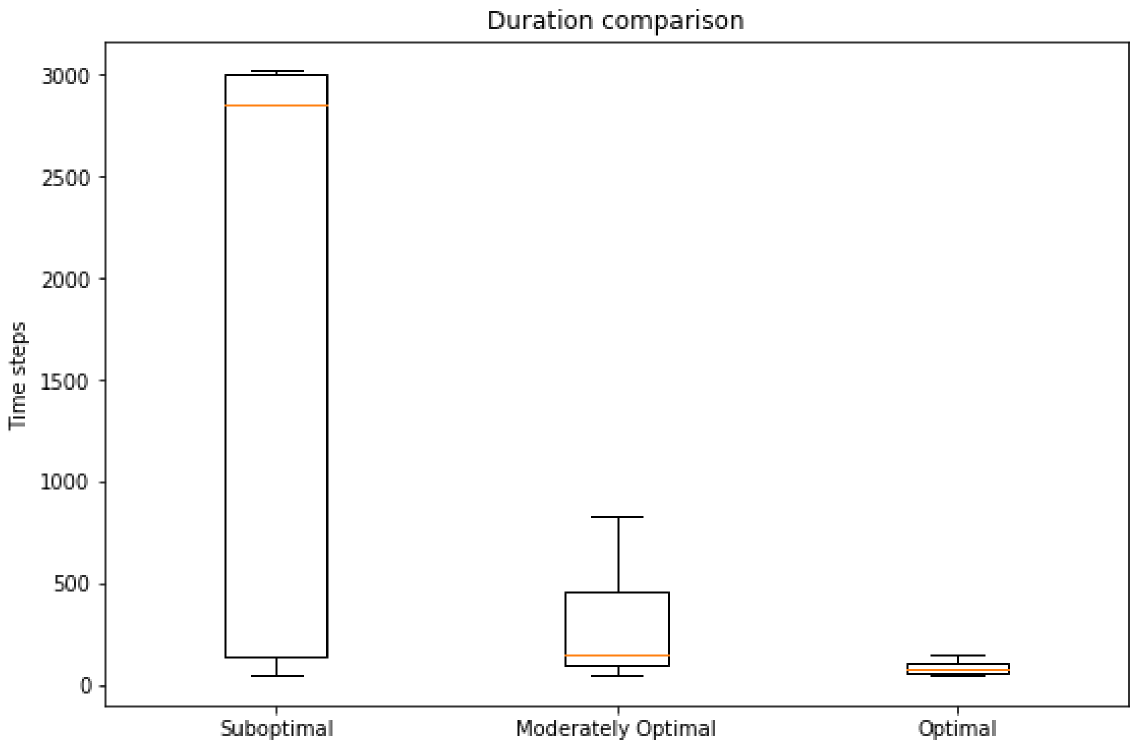


Figure 2: Metric sensitivity - Trip duration boxplots.

Boxplots depicting the trip duration for a vehicle's trip. Each boxplot represents data points for 40,000 trips (20 simulations, with 2,000 trips each).

Departure delay – Metric sensitivity

The amount of time a vehicle had to wait to start its trip. In the context of SUMO, this is measured as the sum of time steps that a vehicle had to wait from the time it was inserted into the simulation, to the time that it first moved.

Departure delays can occur in situations when there is no space in the roadway for the vehicle to begin its trip.

In the diagram below, we can observe that departure delay is sensitive to policy changes, and the results behave as one would expect, the best traffic control policies will result in a shorter departure delay; however, it is important to observe that this metric is not as sensitive as trip duration. We say that trip duration is a better metric because the departure delay values are very comparable between the optimal and the moderately optimal policies, an ideal comparison metric would reflect changes in optimality more clearly. If these two metrics are dependent, trip duration would be a better metric to use in our analysis.

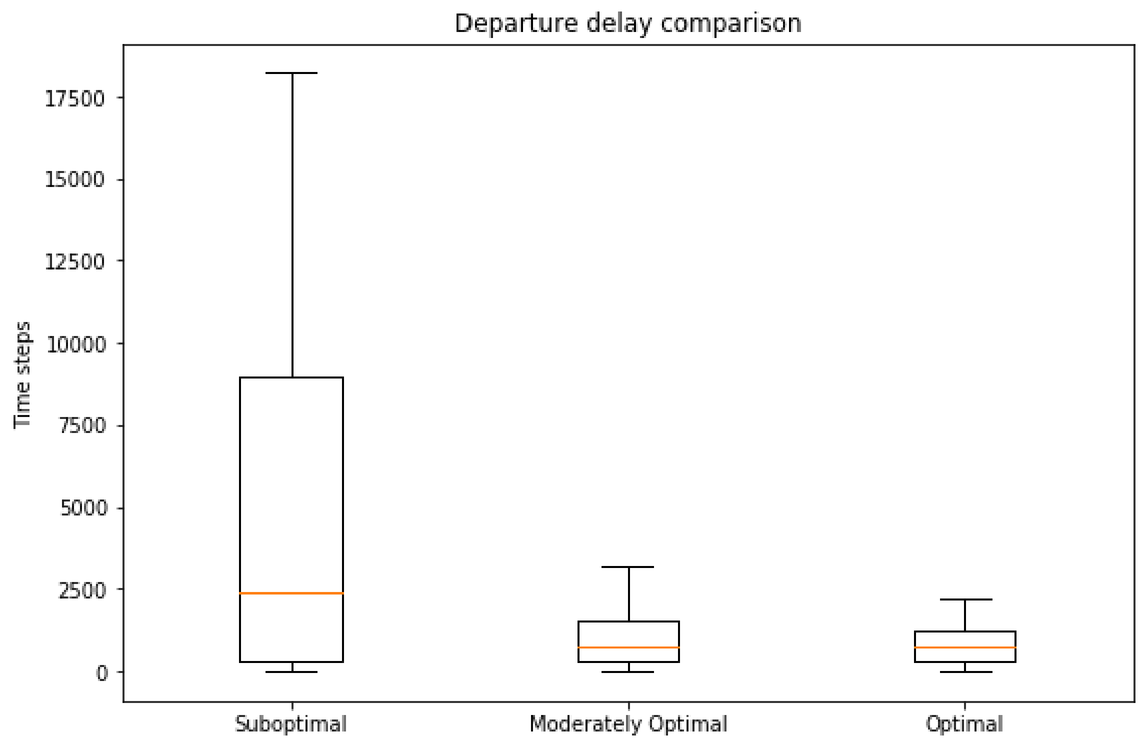


Figure 3: Metric sensitivity - Departure delay boxplots.

Boxplots depicting the departure delay a vehicle experienced during its trip. Each boxplot represents data points for 40,000 trips (20 simulations, with 2,000 trips each).

Time loss – Metric sensitivity

This is defined as the time that was lost due to driving below the ideal speed. The ideal driving speed is the speed at which a vehicle would be able to move if there were no cars blocking its path. This is measured as the difference between the time a vehicle would take to arrive at its destination driving at the ideal speed, and the actual time a vehicle took. In the diagram below, we can observe that this metric is also sensitive to policy changes; moreover, this metric is comparably sensitive to trip duration.

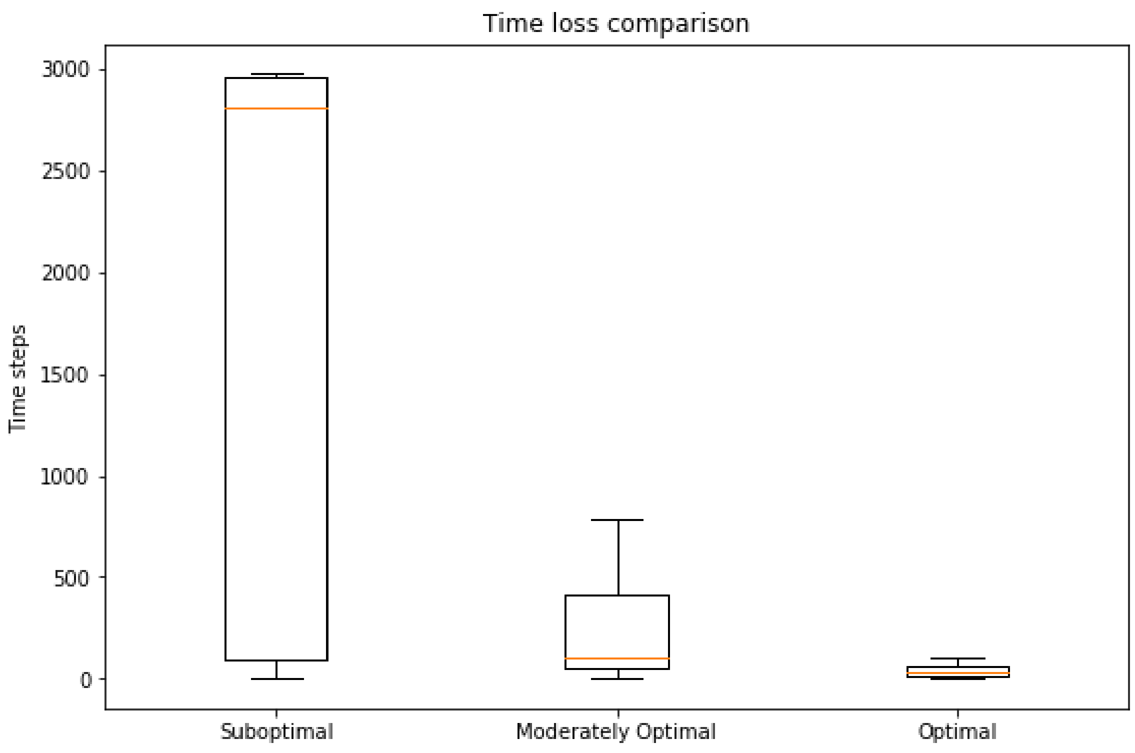


Figure 4: Metric sensitivity - Time loss boxplots.

Boxplots depicting the time loss a vehicle experienced during its trip. Each boxplot represents data points for 40,000 trips (20 simulations, with 2,000 trips each).

Carbon dioxide emissions – Metric sensitivity

This measurement is an estimation of the CO_2 emissions a vehicle would have generated during its time in the simulation. Intuitively, one would imagine that this SUMO metric is dependent on trip duration, given that this is true in the real world.

In the diagram below, we can observe that this metric is also sensitive to policy changes; moreover, this metric is comparably sensitive to trip duration and time loss. If these metrics are linearly dependent, we will use trip duration, due to its intuitiveness.

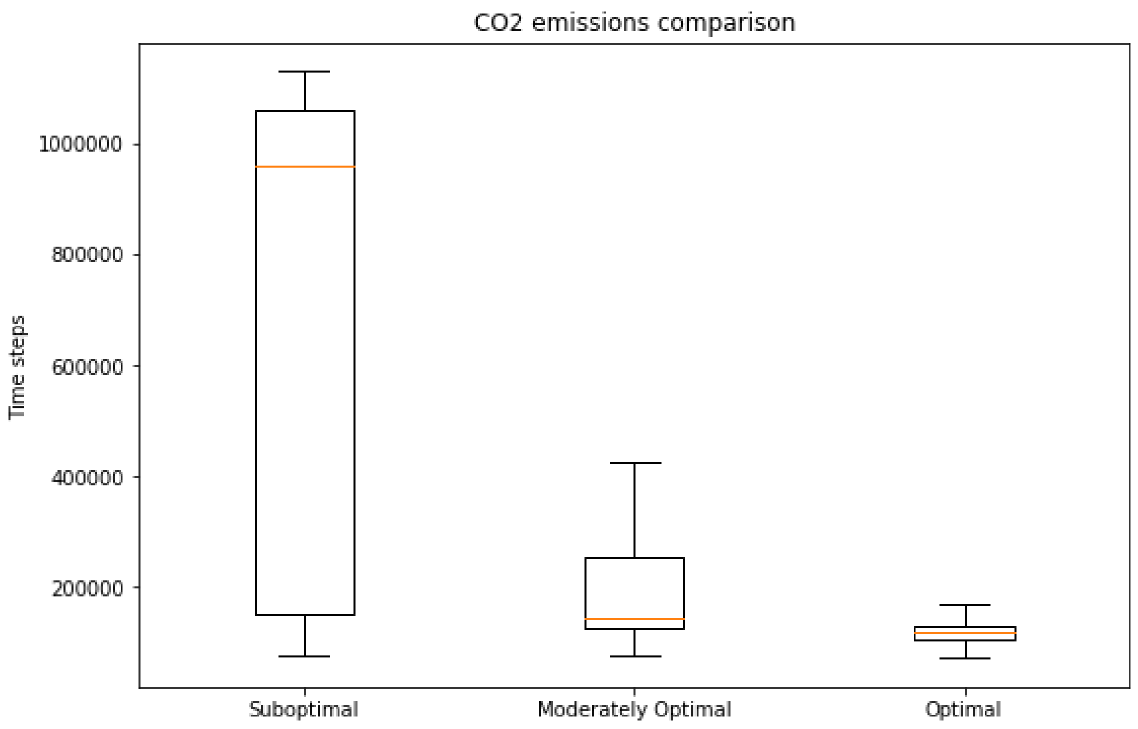


Figure 5: Metric sensitivity - CO_2 emissions boxplots.

Boxplots depicting the CO_2 emissions a vehicle generated during its trip. Each boxplot represents data points for 40,000 trips (20 simulations, with 2,000 trips each).

Wait steps – Metric sensitivity

The time the vehicle did not move. In the context of SUMO, this is the sum of the time steps the vehicle spent standing involuntarily due to other vehicles blocking its path.

In the diagram below, we can observe that this metric is also sensitive to policy changes; moreover, this metric is comparably sensitive to trip duration, time loss and CO2 emissions. Again, if these metrics are linearly dependent, we will use trip duration, due to its intuitiveness.

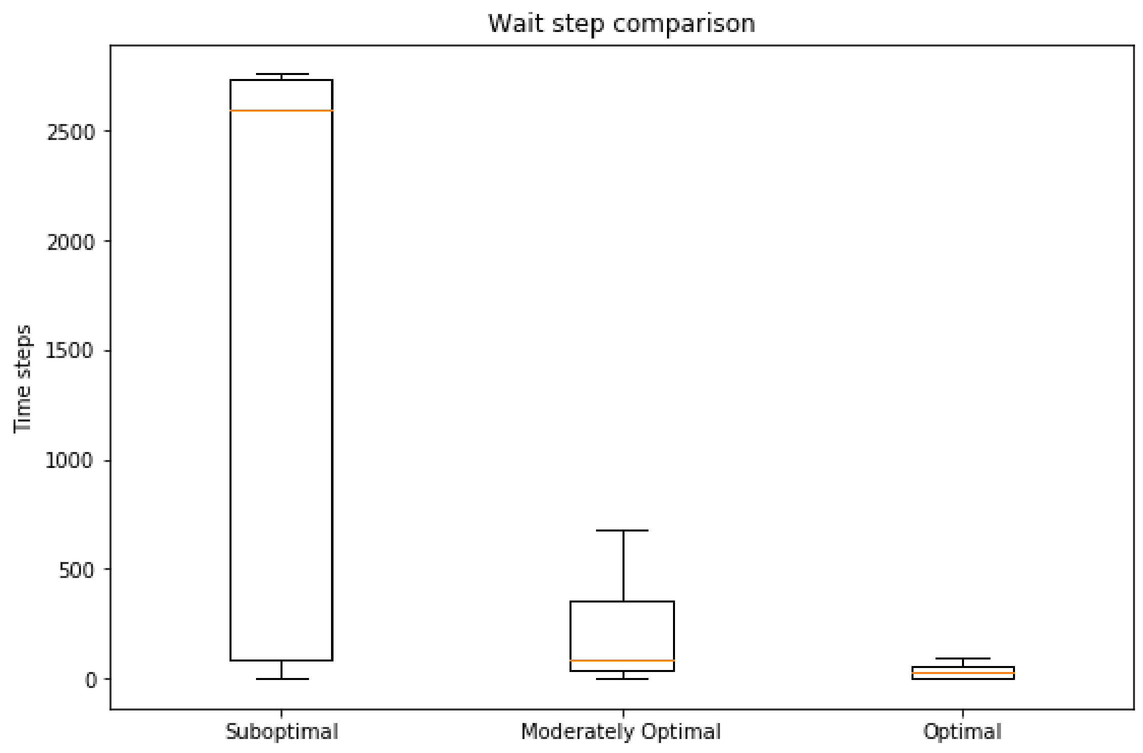


Figure 6: Metric sensitivity – Wait step boxplots.

Boxplots depicting the number of time steps a vehicle had to wait. Each boxplot represents data points for 40,000 trips.

Putting it all together, we have seen that the most useful metrics for comparing any two traffic control policies are *trip-duration, time loss, CO_2 emissions and wait steps*. *Departure delay* proved to be the least sensitive of the metrics evaluated; therefore, it will not be the preferred metric, provided that at least one of the four other metrics meets our needs, departure delay will not be used in any further policy evaluations done on this thesis.

In the next sub-section, we will evaluate the relationships between metrics, to understand how many of these metrics are needed.

Understanding metric relationships

While all of the metrics studied in the previous section are useful, we may not need all of them. If some of them are linearly dependent, then we will only need to use one of the metrics, given that this metric would give us the same information the others would.

We will compare pairwise plots for all of these quantitative metrics, to understand the relationships they have to each other, and conclude by identifying the metric or metrics that will be used for comparisons herein.

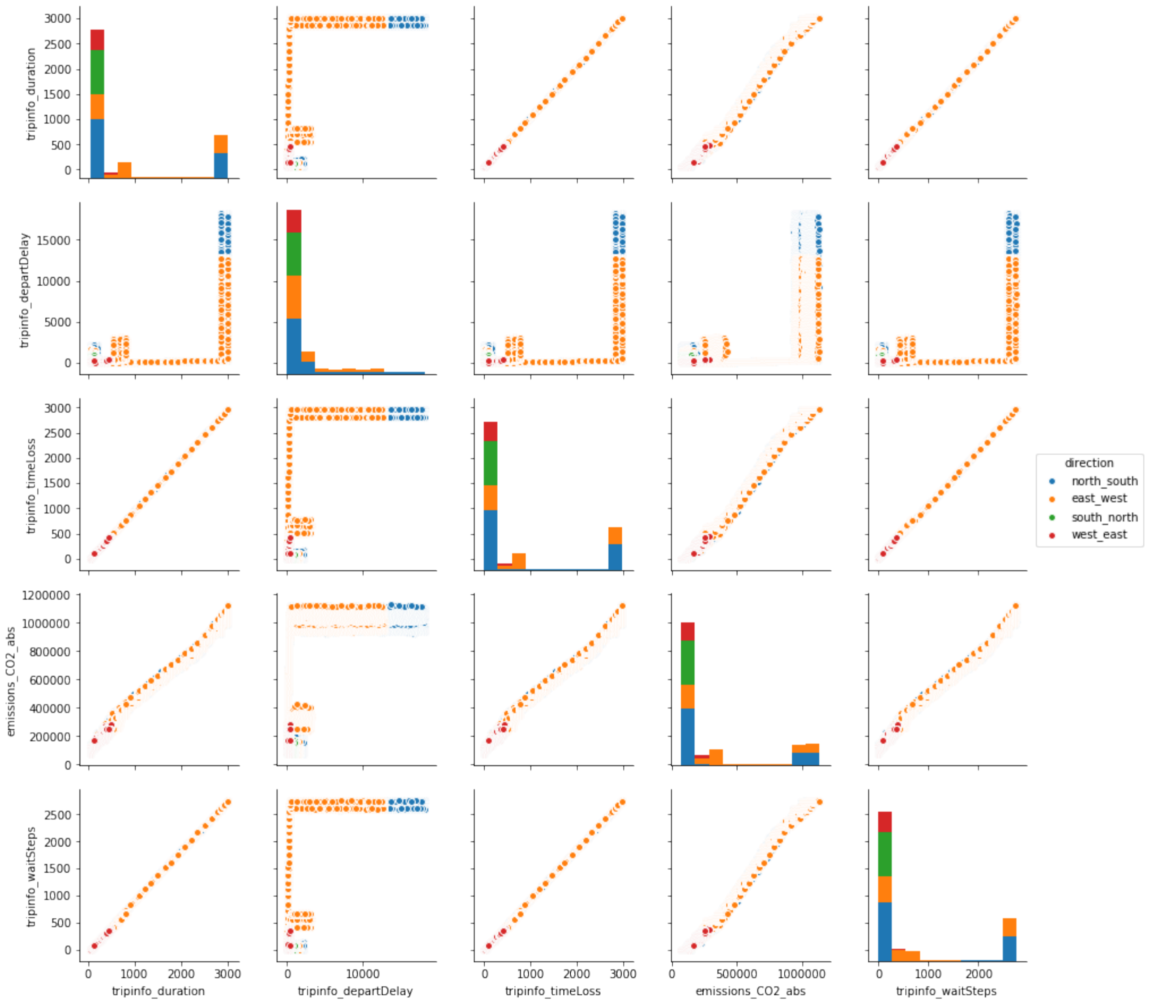


Figure 7: Metric relationships - Quantitative metric pairwise plots.

The diagonal shows the histograms for each metric, all other plots are the pairwise plots of two quantitative metrics that SUMO generates.

The pairwise plots above expose some expected trends in the data: there are different patterns, for the different groups that naturally arise due to the directions in traffic flow; as a reminder for the reader, the traffic flows from heaviest to lightest are as follows: *north-south(blue), east-west(orange), south-north(green), west-east(red)*.

We also observed a very strong relationship between trip duration, time loss, CO_2 emissions and wait steps. On the other hand, we found that departure delay appears to be in a category of its own. This is an interesting outcome, given that it also behaved differently than the other metrics, when evaluating metric sensitivity.

When inspecting the histograms for the four related metrics, we can observe that data follows the patterns of each traffic direction. The directions with the most traffic have better performance overall. We believe that this is a side effect of the two optimal policies, given that two-thirds of the vehicle data plotted above, was managed by these two policies. To explore this further, we would have to break out these statistics by policy and direction; however, this is beyond the scope of this thesis.

The suboptimal policy does a particularly poor job at managing the two heaviest traffic flows *north-south* and *east-west* and this becomes evident in the right side of the histograms for trip duration, time loss, emissions and wait steps, where we can see that some vehicles in these groups did particularly poorly on these metrics.

Departure delay stands as the most puzzling metric and we believe that its nature is the very reason for this. Departure delay is a measure of how quickly a vehicle first moved after being inserted into the simulation, all other events that occurred during a vehicle's trip are inconsequential to this metric. This is consistent with the findings obtained when evaluating metric sensitivity: a policy's optimality may not necessarily affect the behavior of this metric as much as it affects trip duration for example. Given these findings, we will not use departure delay for any evaluations done on this thesis.

Given that the four related metrics, *trip duration, time loss, CO_2 emissions and wait steps* appear to be the most useful metrics to measure a policy's optimality, and that they are nearly perfectly correlated, we will use *trip duration* as the main evaluation metric herein. Trip duration has proven its utility when comparing policy optimality, and is a desirable metric, given the widespread familiarity with this metric.

In order to make more consequential conclusions about traffic behavior, particularly pertaining to the correct usage and interpretation of departure delay, we would recommend a more complete evaluation of the metrics and their relationships; however, we believe this is beyond the scope of this thesis.

As a final remark, we evaluated several summary statistics (median, mean, maximum and minimum) and found that they showed similar behavior across these metrics and traffic policies (results not shown). We will use the average to summarize metrics and compare them across policies when applicable for this work.

In the next section, we will dive into the details of how we propose to manage traffic dynamically, by using Reinforcement Learning in a way that we believe is both efficient and practical.

Chapter VI.

Dynamic traffic management using Reinforcement Learning

Prior publications explored multiple approaches towards optimizing traffic flow, and most of the literature centered around the comparison of fixed-cycle controllers and adaptive controllers.

We consider that there are two large problem areas previous authors tackled, the *optimization of traffic policies*, and the *coordination of traffic controllers* to generate smooth flows, referred to as *green-waves*, by Wiering (2004).

This work will center around the optimization of traffic policies within a single intersection, and we will use Reinforcement Learning for this task. This approach shows great promise in the area of dynamic control, and as shown by Wiering (2004), is a feasible strategy for traffic management.

When stuck in traffic, we would like someone to figure out what is happening, and direct traffic in the way that makes the most sense for the current situation. Considering that promptly dispatching police officers to intersections suffering the greatest congestions is largely infeasible due to resource and logistic constraints, we believe that a technological solution would be highly desirable.

Reinforcement Learning is at the core of the new revolution in Artificial Intelligence, as it is the building block for Deep Reinforcement Learning, a technique so promising, that has been considered a promising foundational component towards Artificial General Intelligence, by some researchers, such as Arel (2012).

As stated in Chapter III, we believe that Reinforcement Learning is well suited for this problem and believe that good features representing state and rewards can be obtained through the mechanisms provided by the SUMO API. For example, the ability to obtain the mean speed of the vehicles about an intersection will be useful when computing rewards and the ability to poll for the number of vehicles that await on a given approach will be useful when deriving the environment state.

The Reinforcement Learning algorithm

Similarly to the approach described in Wiering (2004), we used Q-learning, a model-free Reinforcement Learning algorithmic technique for finite Markov decision processes, the main components of a Q-learning algorithm are: the learning rate, a constant called *alpha*, the discount rate for future rewards, a constant called *gamma*, quality estimate for a given action, given by the *reward function*, and the agent's model of what it has learned from its actions and the rewards it has obtained, called the Q-*table*.

The constant typically named *alpha*, the learning rate, which determines how much value the agent gives to the new information. This typically has a value from 0 to 1. An agent with a high learning rate will be more inclined to explore different actions, whereas an agent with a zero-valued alpha will behave deterministically, according to the policy it has learned. It is unclear how Wiering et al. (2004) chose the learning rate for their algorithm, as this is not mentioned. We chose to experiment with various learning rates and chose the one which produced the most satisfactory results, this will be explored in the next section, which discusses the training process for the Q-Learning algorithm.

The constant *gamma*, which determines how much value is placed on the future rewards. This typically has a value from 0 to 1. An agent that has a zero-valued gamma is said to be myopic, due to the fact that it only considers immediate rewards. Similarly, to the *alpha* constant, Wiering et al. (2004) do not mention how they chose this value. Just like *alpha*, this is a parameter we will select through the results of various experiments to understand how the agent's behavior responds to these parameters.

The *reward function* that computes the benefit that was obtained by an action, and the *Q-table*, a mapping of the estimated benefit or quality of an action, given a state and said action. The Q-table is how the agent encodes prior knowledge and is the main tool used when evaluating various potential actions for its current state. This is where our design decisions significantly diverge from the prior work. Prior work used a co-learning model that kept the Q-table state on a vehicle level, whereas we do so at the intersection level. Additionally, rewards in Wiering et al. (2004) are Boolean, describing if a car moved or not, and rewards in our algorithm are an integer, which is a function of the mean speed of all vehicles around the intersection.

In Wiering et al. (2004). the authors used a model relying on car-based controllers, and have a Q-table mapping which encodes states using a tuple of *(node, direction, place, destination),* and has two possible actions, either the light goes green or red. They call a traffic light a *node*. *Direction* is an identifier that represents in which lane a car is waiting, given that they use two-lane roads, this has 8 values. *Place* is the current location in the traffic grid. *Destination* is the place in the traffic grid which the vehicle needs to reach to meet its goal.

Contrary to the prior work, one of our design tenets is that the ideas explored here need to be realistically implementable at the moment. For example, a barrier for the implementation of the work in Wiering et al. (2004) is the coordination and information sharing between vehicles and intersections.

The *state* that the Q-table uses is encoded as a function of the number of cars that are waiting in each of the 4 incoming directions. Each direction can take a value from 0 to 9. The value representing traffic approaching from the north is encoded as the units, traffic approaching from the east is encoded as decimals digit, traffic from the south is encoded as hundredths digit, and traffic from the west approach is encoded in the thousandths digit. This means that at any point in time, our agent may be in one of 10,000 states. A slightly simplified version of the function that transforms incoming queue length to an integer between 0 and 9, and the state equation are provided below.

$$f(n) = Min\left(\left\lceil {}^{n}/_{k} \right\rceil, 9\right) With\ k = 5\ in\ the\ current\ implementation.$$

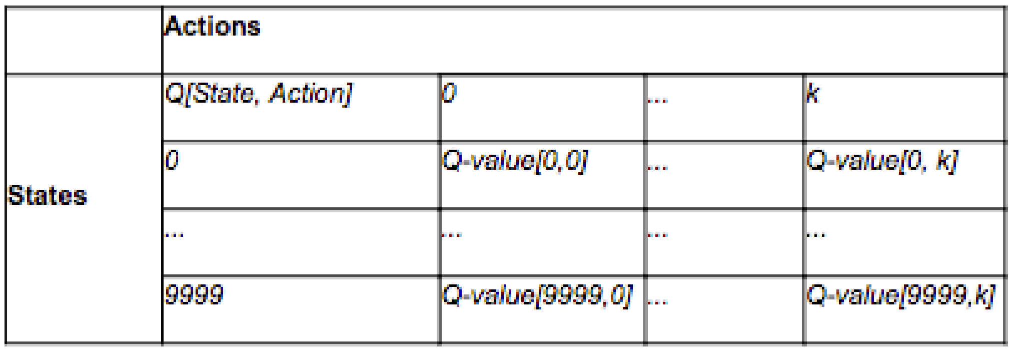
$$\begin{aligned} State &= f(queueLengthNorth) \\ &+ 10 \times f(queueLengthEast) \\ &+ 100 \times f(queueLengthSouth) \\ &+ 1000 \times f(queueLengthWest) \end{aligned}$$

Actions, defined by a set of atomic sequences, enable traffic in a given direction for some time period. We then notify vehicles in this direction that traffic flow will stop through the use of yellow lights. Finally, all lights are set to red for a brief period of time to ensure that the intersection is clear of vehicles. There are a fixed number of actions that an agent can take, and each action is designed so that we enable traffic in a particular direction or combination thereof, while guaranteeing the safety of vehicles that are traversing the intersection.

The *reward* is a function of the mean speed for all of the cars that flow through the intersection. The reward is used after an action completes, to update the *Q-value*, the quality estimate that is the agent's running estimate of the merits of taking an *action* whenever it finds itself in a given *state*. This running estimate is continuously updated as follows:

$$Q[State, Action] += Alpha \times \{Reward + Gamma\ \times \max(Q[state']) - Q[State, Action]\}$$

Putting it all together, our algorithm encodes *state* as an integer, which estimates the number of cars waiting in each cardinal direction (North, South, East, West), *actions* are a set of phases that ensure the safety of vehicles that traverse the intersection, and the *reward* is a function of the mean speed of the vehicles that traversed the intersection during the time the *action* was enabled. In order to allow the reader to solidify his mental model of this approach, and of Q-learning, below is a graphical representation of how the Q-table for this work looks like.



	Actions			
	Q[State, Action]	*0*	...	*k*
States	*0*	*Q-value[0,0]*	...	*Q-value[0, k]*
	...	...	...	...
	9999	*Q-value[9999,0]*	...	*Q-value[9999,k]*

Figure 8: Illustration of Q-table.

This table describes the Q-table that a Q-Learning algorithm uses. This table encodes the algorithm's knowledge of the world. As the algorithm receives a reward, the cells for the State and Action that were responsible for this reward will be updated. In the particular implementation presented in this thesis, there are 8 actions the agent can choose from. The possible actions are: North-South, East-West, South-North, West-East, NorthSouth+SouthNorth, EastWest+WestEast, LeftTurn-NorthToEastAndSouthToWest, LeftTurn-EastToSouthAndWestToNorth.

We believe that all of these decisions are sensible choices that could be implemented in the real world. For example, the environment information we need is just a function of the number of cars waiting at each approach of the intersection. As a reminder to the reader, this is used to derive the agent's representation of the environment state and used in the Q-table, which encodes the quality estimates that the agent has learned, and the speed at which vehicles cross the intersection, both of which can be obtained through induction loops. This is actually how SUMO models this information. For example, in order to obtain the number of vehicles waiting at an intersection or the mean speed of vehicles that crossed an intersection, one must poll the induction loop detectors for this information.

It may be possible that the authors of SUMO may have included functionality that is not readily available in modern induction loops, but we believe that this is unlikely, as there are researchers trying to solve the problem of how to obtain speed information through the use of just one induction loop, as mentioned in Lu et al. (2012).

In the next section, we will explore how we trained the algorithm, and how this training data lead to decisions about the parameters that we will use for *alpha*, the learning rate, and *gamma*, the discount factor for future rewards.

Reinforcement Learning agent training

Before the Reinforcement Learning algorithm can make advantageous decisions, we must first train it to understand how well it learns with different training settings and identify which are the best settings that lead to smarter agents with the most stable behavior.

In this section, we will go over the results of training the algorithm for 2,000 simulation episodes under different settings and discuss the behavior of each setting independently. As a reminder to the reader, each training episode is a simulation with 2,000 trips, resulting in 4 million trips in total. In this section, whenever we refer to a training or simulation episode, we can assume that this maps directly to one simulation.

As a reminder for the reader, this work will use two traffic patterns: the traffic pattern which represents a normal day's rush hour traffic on a downtown intersection, and another one which represents the traffic patterns for the same intersection, whenever an event such as construction disrupts traffic. We will train the algorithm using the first traffic pattern, the other one will be used later on to measure the adaptiveness of the algorithm.

All of the plots in this section represent the evolution of the agent's performance from an untrained state, to a trained state in which the agent has been exposed to various iterations under the same traffic pattern for various training episodes. The x-value is the episode number, and the y-value represents the average trip duration during a simulation episode.

The expectation of this exercise is to be able to measure how well the agent minimizes the average trip duration under various parameters and to evaluate the stability of its behavior. If a configuration produces great results for some training episodes, but its behavior does not show steadiness, this will be a configuration we deem undesirable, conversely, a desirable configuration would show that an agent's behavior converges towards optimal or close to optimal behavior.

We begin this exercise with a small learning rate *alpha* of 0.1 and a 0-valued discount factor *gamma*. As a reminder to the reader, this type of agent would be considered a myopic agent, given that it does not take into consideration its estimate of future rewards, it only cares about the immediate reward that it can get, and it does not place a lot of value on this immediate reward, by using a small learning rate, it discounts the reward heavily. This type of agent is expected to converge slowly towards the optimal solution.

Through the plot below, we can see that this configuration's behavior matches our expectations and improves its performance over the course of training. The algorithm learns very slowly with this configuration, and after extensive training still has large variability in its results, as we can see after 2,000 simulations. We expect that more aggressive learning configurations will mitigate this problem.

Additionally, we believe that if this configuration was permitting more episodes to learn, it would achieve a more stable behavior, but this is something we have yet to see, and which we will explore as we evaluate the other training configurations.

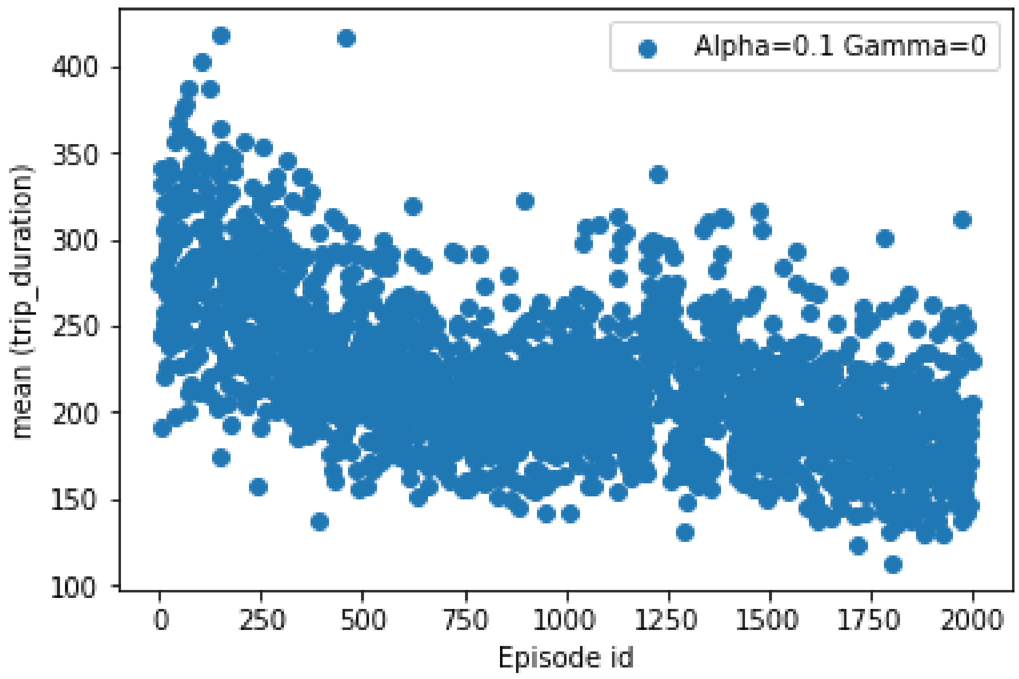


Figure 9. Agent training with alpha=0.1, gamma=0

In this configuration, we are training an agent that is said to be myopic, given that it does not use estimates of future rewards in its decisions. This agent's improvement is slow but stable.

Next, we continue to explore a configuration that uses the same small learning rate *alpha* of 0.1 and a small discount factor *gamma* of 0.1. This would not be considered a myopic agent, given that it is taking future rewards into consideration.

Despite the fact that this agent takes future rewards into consideration, we quickly begin to see non-optimal behavior. Around the 120th episode, this agent's behavior is undesirable and its selection of actions is non-optimal, so much that episodes do not complete. Actions which result in no movement were selected repeatedly, this can be seen in this graph through the x-axis, the fact that it is much shorter, means that we had to halt the training of this agent, given that simulations could not finish. Only120 simulation episodes were completed, compared to 2,000 as can be seen in the plot above.

The reason for this is a common problem in Reinforcement Learning, known as propagation of errors, and this happens due to the *gamma* discount factor. This does not mean that the *gamma* factor is detrimental to an agent's performance and that we should always make it zero. It just means that this factor should be used judiciously. By giving value to actions the agent made in an untrained state, we are skewing the behavior of the agent in undesirable ways.

Lavet et al. (2015) discuss the fact that the Gamma discount factor for future rewards can be a source of error propagation. They propose that this factor be incremented slowly as agents learn and that by doing this, learning can be accelerated, while minimizing the propagation of errors that would occur due to an untrained agent. At the moment, this kind of technique will not be used in this work, given that accelerating the learning rate of an agent is beyond the scope of this work; however, using this finding we will only use 0-valued discount factors to mitigate this problem.

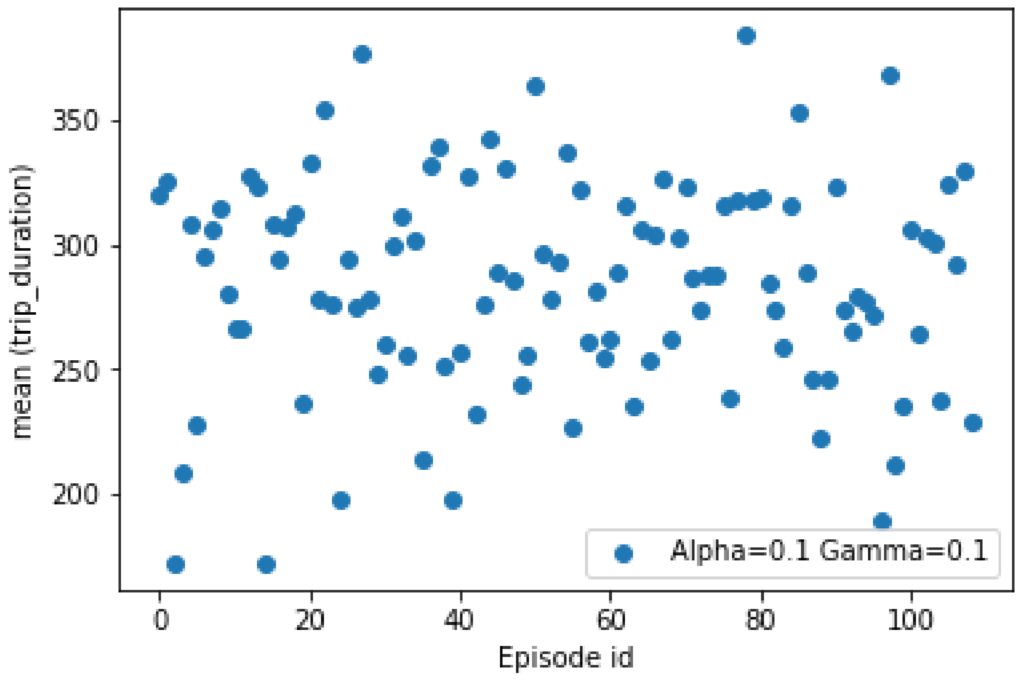


Figure 10. Agent training with alpha=0.1, gamma=0.1

In this configuration, we are training an agent that is not myopic and uses its estimate of future rewards in its decision making. This configuration is subject to error propagation, and around the 120th episode, simulations ran out of allotted time due to bad decisions taken by the agent. Simulations that ran out of allotted time do not have data points in this plot.

The next configuration we tried used an *alpha* learning rate of 0.5 and a *gamma* discount factor of 0, given the new information we discovered in the past experiment. This configuration is similar to our first configuration, so the agent can also be said to be myopic, but it is going to be a myopic agent that should learn from its experience faster than the agent in our first configuration.

When inspecting the plot below, we can observe that this configuration learns much faster than the first configuration, with an *alpha* of 0.1, heading towards a mean value of 200 much faster than the first configuration, around the 250th episode, while the first configuration experienced a slower descent towards this mean value.

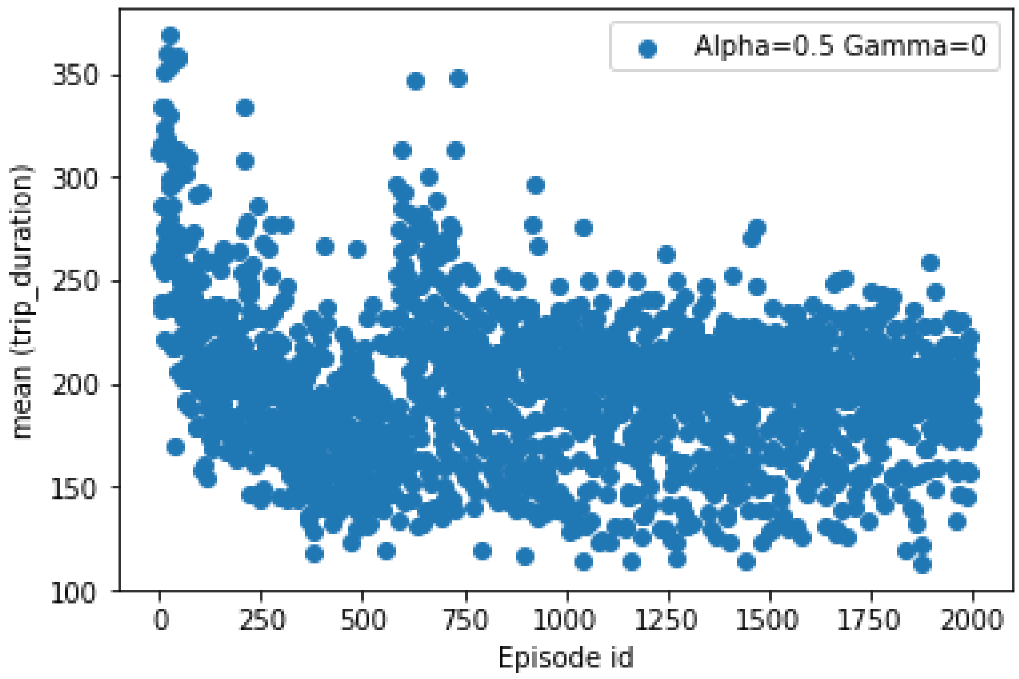


Figure 11: Agent training with alpha=0.5, gamma=0

In this configuration, we are training a myopic agent with a moderate learning rate of 0.5, we can see that this agent steers towards a mean trip duration of 200 around the 250th training episode, this is much faster than the slow descent experienced with the first configuration of alpha of 0.1, and gamma of 0.

The last configuration we will experiment with, uses an alpha of 0.9, and a gamma of 0, to avoid propagation of errors. We originally expected this very aggressive learning rate to be fast, but potentially unstable. This is partially true, during the first 200 episodes we see a large amount of variation in how the agent performs; however, around the 1000th episode we see that the agent's behavior is very consistent, achieving results that are very close to a mean trip duration of 200 time steps.

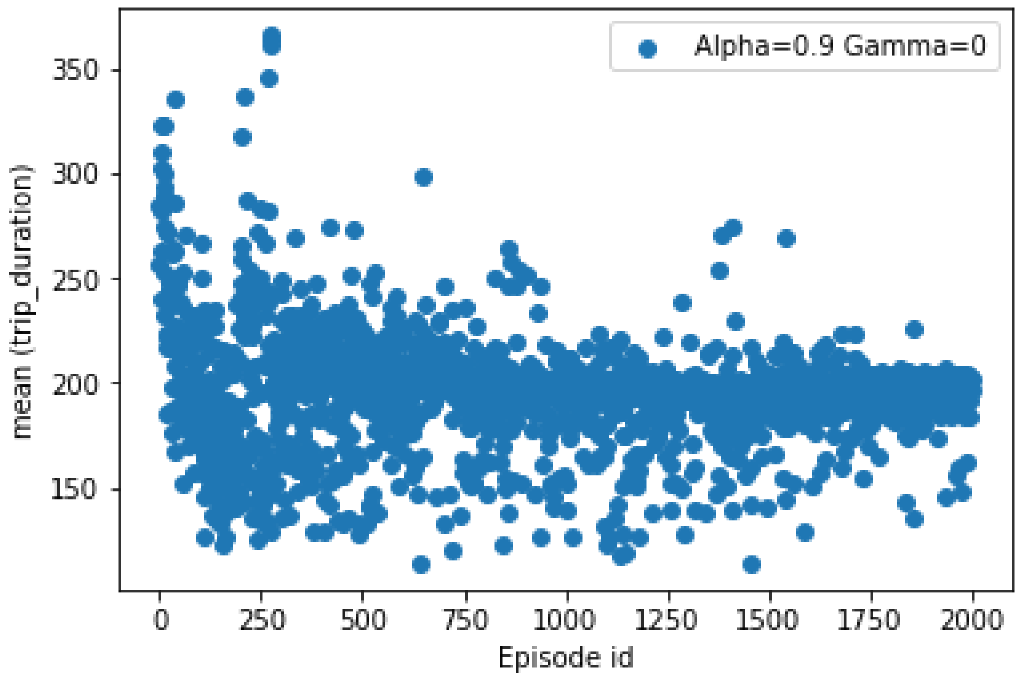


Figure 12: Agent training with alpha=0.9, gamma=0

In this configuration, we are training a myopic agent with an aggressive learning rate of 0.9. Despite the fact that this agent has large variations in the first 250 episodes, we see that the agent starts heading towards stability around the 1000th training episode.

Finally, as we prepare to conclude this chapter, it is helpful to recapitulate some of the findings to cement our understanding of the discoveries found here. To help the reader visualize some of these findings, *Figure 13* will present an overlay of the plots for the four training configurations.

There were a few surprises uncovered in this exercise, initially, we found that slow learning rates take a long time to achieve desirable values consistently, but overall, behave very well. Aggressive learning rates do not necessarily mean instability, as we have seen. After some time, the agent actually converges towards a desirable mean trip duration.

The most useful and interesting finding in this chapter was the fact that the discount factor for future rewards can cause undesirable behavior, and needs to be judiciously used, and adjusted dynamically. For example, in *Figure 13* we can see the divergence experienced with the configuration that used *Alpha=0.1* and *Gamma=0.1 (in orange)*. Under this setting, we could not complete the expected 2,000 training episodes. This finding leads to the decision of using a zero-valued gamma for the rest of this work, as optimizing for faster learning is beyond the scope of this thesis.

Finally, when deciding what value to use for the *alpha* learning rate, we would prefer an aggressive learning rate of 0.9 initially, provided that we use enough training episodes and verify convergence towards a desirable mean value; however, after the agent has been trained, we would probably use a conservative learning rate of 0.1, for stability. As we will see in the next section, this is exactly what we did when we tackled the problem of comparing the behavior of an optimal static policy, against this dynamic control agent.

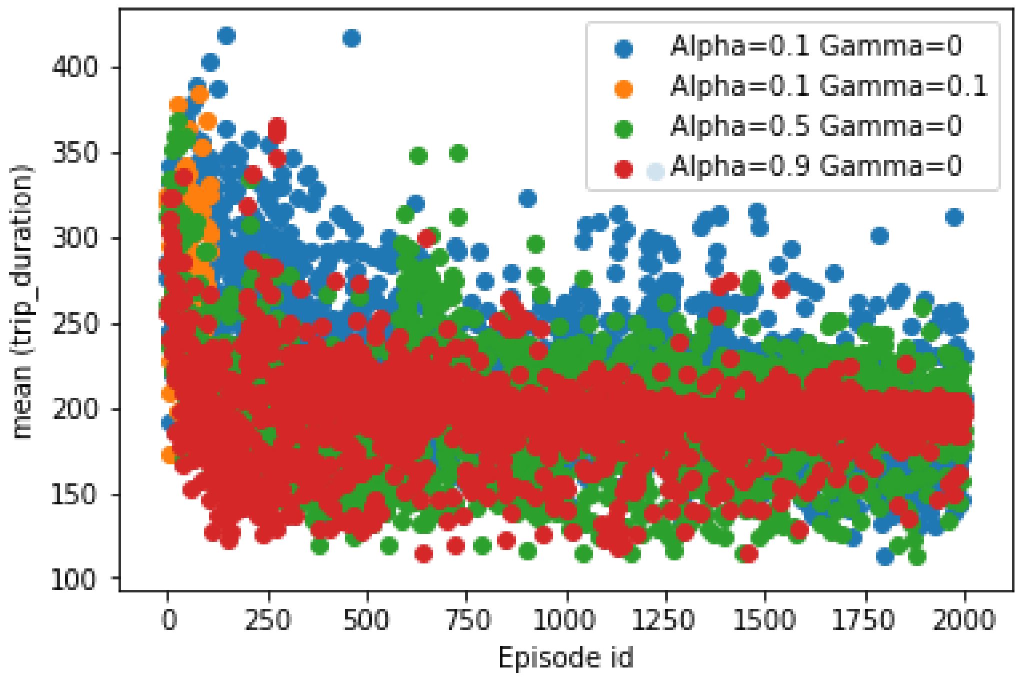


Figure 13: Agent training overlay plot.

This is the overlay plot of all of the combined training scenarios. Here we can see the contrast of how fast the aggressive learning rate achieves convergence, and how the slow learning rate still has some exploration to do in the last training episodes to further decrease variability in its performance.

Chapter VII.

Traffic management strategies head to head

It is now time for the main event, where we will evaluate the hypothesis of this thesis, and finally see how a carefully crafted static policy compares to a dynamic controller that uses Reinforcement Learning. The dynamic policy has been trained by now and should be well prepared for the challenge, but the static policy will not go down without a fight. It was crafted with complete knowledge of the traffic patterns which we will use, which means that it will not be an easy contender.

A few chapters back, in Chapter IV we described the simulation, the traffic grid, and the traffic patterns that dictate how vehicles are generated. We also described their origins and destinations, which in turn dictates how these vehicles arrive at different sections of the intersection. As a reminder to the reader, we will summarize some of this below.

We have a four-way intersection, each direction has three lanes, and left turns are permitted in this intersection, but not all traffic patterns have left turns.

We have two traffic patterns, the first one, we call our *normal traffic pattern*, and does not include any left turns. A total of 2,000 vehicles traverse the intersection under this setting, in this specific setting, no left turns are expected.

The second traffic pattern represents a disruption to the traffic infrastructure and can potentially cause significant disruption to the trip duration of commuters that have to cross this intersection. While this traffic pattern does not represent a significant increase in traffic, with 2,200 vehicles traversing the intersection, it does introduce left turns, which is a slight divergence from the expected setting, described above.

Comparing the trained agent against the optimal static policy

We ran 100 simulations using the traffic patterns for which the static policy was designed for on both the static policy, and the dynamic policy controlled by our Reinforcement Learning agent, and the results are in. Unsurprisingly, the static controller wins this round. This does not mean the demise of our dynamic traffic controller, it just means that given full knowledge of the traffic patterns, a carefully crafted policy performs very well.

Inspecting the box plots below, we can see that while the static controller indeed does better than the dynamic controller, part of this is a bit unrealistic, this policy was designed to be as close to optimal as possible.

The median in the two box plots is not too far away, being 71 for the static policy, and 143 for the dynamic policy, despite the fact that our dynamic control policy did not outperform the static policy, its results are nothing to be ashamed about, considering that no human supervision was involved in achieving them, apart from the initial parameter configuration.

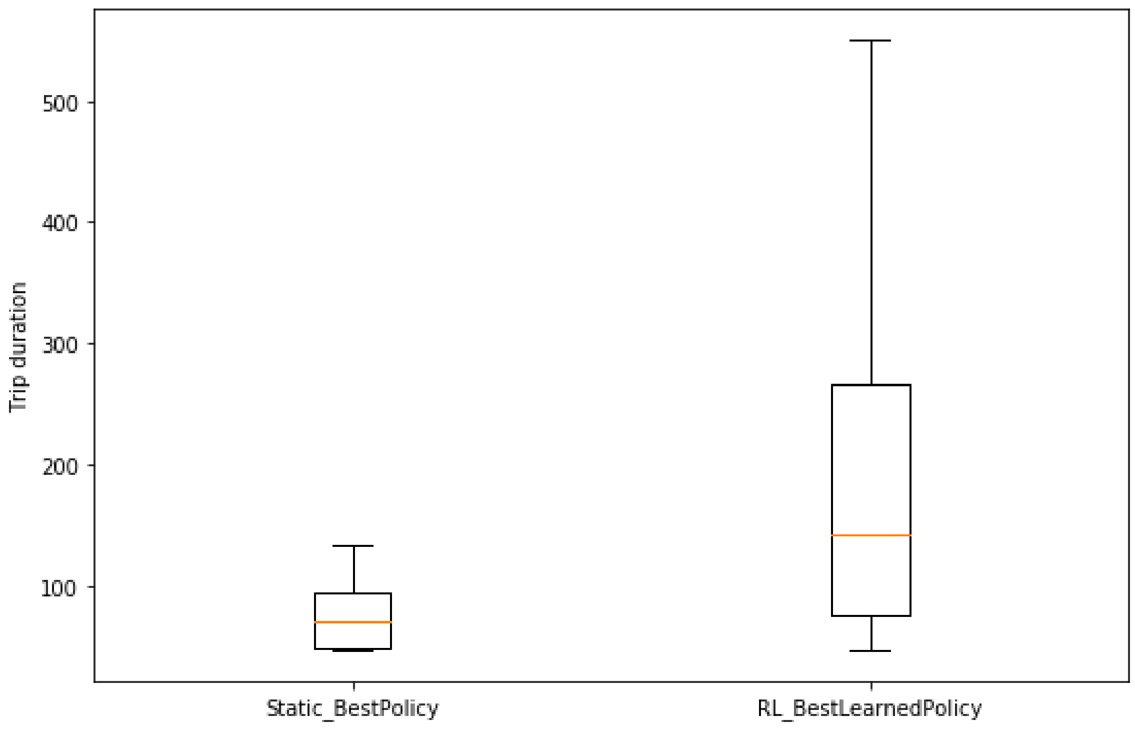


Figure 14: Box plots for policy comparisons under expected traffic conditions.

Under expected traffic conditions, a carefully crafted static policy beats a dynamic policy.

In the next section we will compare both of these policies using traffic patterns that none of these policies have been explicitly designed to manage. As a reminder to the reader, we trained the dynamic policy using the traffic settings we used in this section. It has not been exposed to the traffic patterns that will be used in next section.

Effects of unexpected traffic patterns

We ran 100 simulations under the different configurations which represent abnormal traffic flows that could have been caused by an accident, construction, or another type of event.

The static controller manages to clear up traffic but, does not do this efficiently, and drastically worsens the commute for all of the drivers of the vehicles that need to traverse this intersection. The median trip time has now shot up to 1,512 time steps, compared to the configuration this controller was designed for, this is not a trivial increment. Additionally, this controller results in very large trip duration variability, with some trips taking 12,000 time steps, more than 100 times what was experienced under its optimal setting.

On the other hand, we have that the dynamic controller performed quite well, despite not having any training for this specific configuration, it is able to adjust, and achieves a median trip duration of 357 time steps and far lower variability, when compared to the static policy, which means that most trips take the same amount of time to complete.

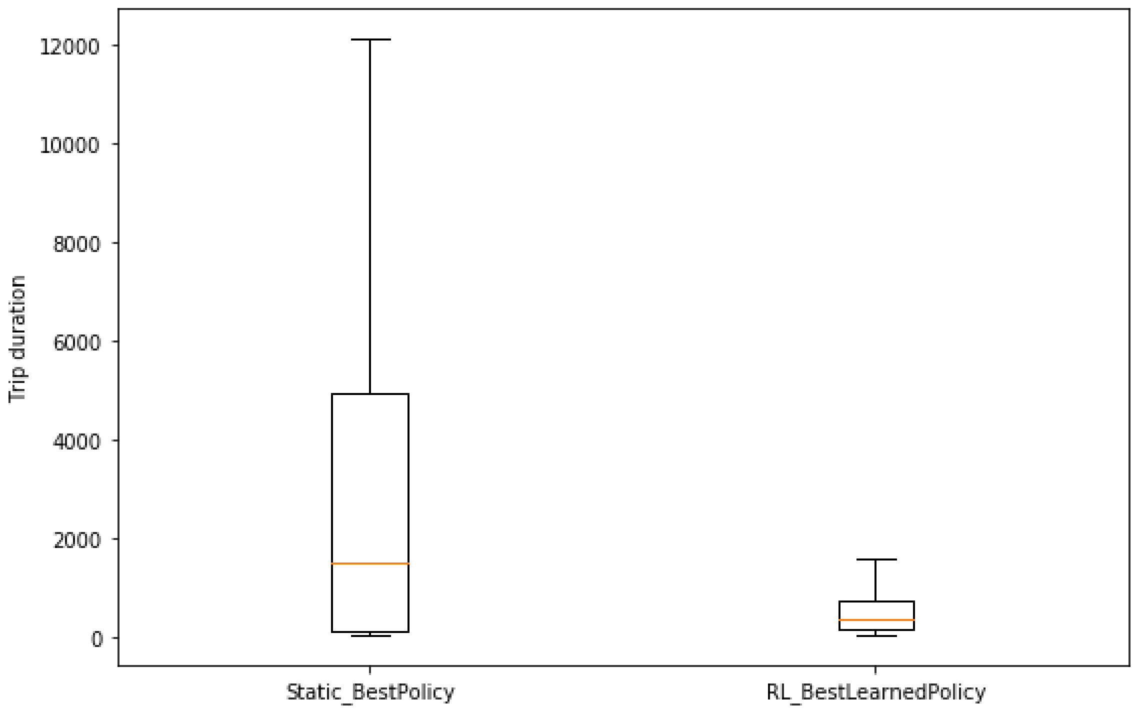


Figure 15: Box plots for policy comparisons under unexpected traffic conditions.

Under unexpected traffic conditions, dynamic control policies shine. Given the fact that we seldom have full knowledge of traffic flow volumes, and the fact that disruptions are part of life, dynamic policies clearly show some promise.

Now that we have seen the strengths and merits of the two types of traffic controllers, it is time for us to conclude this work, by elaborating on the results we have discovered in this study, and by outlining some future directions that may result from our findings.

Chapter VIII.

Conclusions

The work presented in this thesis explored the differences between static traffic control strategies and dynamic control strategies. We understood how to appropriately compare two traffic control strategies, and we explored how two different control strategies fare under the configuration they were designed for, and under unexpected circumstances.

Through the results presented in the previous chapter, we were able to partially confirm the hypothesis presented earlier, which stated that *dynamic control policies are a better traffic control mechanism, due to their adaptability.* Dynamic control policies, particularly the one presented in this work shine due to their ability to manage situations they had not previously been exposed to.

If we were to concisely summarize the characteristics of each class of controller, we would say that *dynamic controllers are able to manage a wider range of traffic configurations with near optimal performance but may not achieve maximum optimality for any of them*. Meanwhile, *static controllers are able to optimally manage a very specific traffic configuration but experience severe difficulties when managing configurations for which they have not been designed for*.

We believe that the fact that our controller did not achieve the same level of optimality as the static configuration does not mean that it is impossible for dynamic control policies to find the most adequate sets of decisions to manage traffic. We think of this as a minor setback, and we will shortly elaborate why.

One of our controller's priorities is ensuring the safety of vehicles traversing the intersection; therefore, when changing the way traffic flows, our controller stops all directions, despite the fact that there may be at least one direction which could have remained enabled, given that the next phase also enables it. This is an area of opportunity that could improve the smoothness of traffic flow when managing traffic dynamically, and we will speak more about this, and other promising opportunities in the next section.

It is clear that static traffic control policies are not without merit, they are the most common method for traffic control, and if carefully crafted and updated, they can perform very well; however, unpredictability is part of life, and this is where dynamic control policies shine, and this is the main reason we believe that the days of static traffic control policies are numbered. A day will come when dynamic control policies become the norm.

As a final remark for this section, it is worth mentioning that despite the fact that this work focused on the study of efficient control of signalized traffic intersections and the subject of our case study was a simple four-way intersection, we believe that this work will generalize well to flared and channelized intersections, given that these are just mechanisms to alleviate traffic close to an intersection.

Chapter IX.
Future directions

Despite the discoveries made in this work, much remains to be done in this field to achieve a better experience for travelers and commuters.

We can classify the opportunities we see for future work as follows: *traffic controller-based optimizations, car/driver-based optimizations, practical implementations*.

The work described in this thesis falls under the area of *traffic controller-based optimizations*. Some opportunities for improvements we see are related to the smoothness of traffic flows, that is, enhancements in traffic control strategies that would minimize the number of times a vehicle had to stop while traversing the network.

The controllers we implemented do not prioritize the smoothness of traffic, and sometimes stop traffic in situations where this is completely unnecessary. For example, if traffic is flowing in a given direction, and we enable a different phase, which also enables the current direction in which traffic is flowing, we do not need to halt the current direction, traffic can continue flowing. It is only when we determine that the next phase to enable has a conflict with at least one of the currently active directions that we must stop traffic. Given how close the comparison for the known traffic pattern was, we speculate and are optimistic that this may have been a reason why the dynamic controller could not match the static controller.

Smoothness can also be achieved through the coordination of adjacent intersections to minimize the number of times a vehicle has to stop. This work explored the control of traffic within a single intersection; however, it is yet to be determined how this would scale once all controllers within a traffic grid attempt to coordinate traffic. Would coordination happen seamlessly by each controller taking the best decision by just using information of the behavior of traffic that is local to it, or would intersections need to communicate in order for coordination to be achieved?

Perhaps the approach that most naturally lends itself to this is the work that Sanchez et al. (2010) did, by using genetic algorithms. It is unclear at the moment how we would use local knowledge created by controllers such as Reinforcement Learning to benefit the whole network. We think this is an area of great potential from both the research standpoint and its promise to make a sizeable impact.

The car/driver-based optimization that we think would have the most benefit is related to load-balancing. All traffic networks have a limited bandwidth, so we think that this is an area of opportunity in which we can borrow from the work done in computer networks, to optimize traffic flows within cities. This is currently done by proprietary technologies integrated into smartphones, such as Google maps. In order for this to have wide-spread adoption and more resounding impact, this would probably need the definition of an open protocol, so that all vehicles in the network are able to take advantage of this.

To the best of our knowledge, prior work appears to put little emphasis on how their findings could be implemented. This is a natural next step that may drive additional innovation. Existing open standards such as the National Transportation Communications for Intelligent Transportation Systems Protocol NEMA (2018) present an opportunity for defining an open architecture which cities could adopt as they see fit. Extending prior work so that it is deployable as cities see the need for it would be a great opportunity in the area of *practical implementations.*

Chapter X.

Final remarks

To conclude, this thesis illustrated that the use of readily available information about the traffic characteristics that are local to a traffic intersection in combination with a broadly used machine learning technique enabled a traffic control mechanism that is robust enough to adapt to drastic changes in traffic patterns. A broader evaluation of this methodology on more complex traffic networks and traffic patterns can pave the path forwards toward its deployment in cities. After all, it is only after a practical roll-out that this work will make a difference to our quality of lives.

Chapter XI.

References

Publications

Arel I. (2012) Deep Reinforcement Learning as Foundation for Artificial General Intelligence. In: Wang P., Goertzel B. (eds) Theoretical Foundations of Artificial General Intelligence. Atlantis Thinking Machines, vol 4. Atlantis Press, Paris

Findler, N. V. (n.d.). Automatic Rule Discovery for Field Work in Anthropology. Retrieved from https://pdfs.semanticscholar.org/bc91/d6aee79bb6c5fd2f9649063c5ffa46025a4d.pdf?_ga=2.177425445.1425997419.1494195246-1723602033.1494195246

Findler, N., & Stapp, J. (1992). A Distributed Approach to Optimized Control of Street Traffic Signals. Journal of Transportation Engineering, 118, 99–110. https://doi.org/10.1061/(ASCE)0733-947X(1992)118:1(99

François-Lavet, V., Fonteneau, R., & Ernst, D. (2015). How to Discount Deep Reinforcement Learning: Towards New Dynamic Strategies. Retrieved from http://arxiv.org/abs/1512.02011

GreenLightDistrict-Sourceforge (2018) The Green Light District sourceforge page. Retrieved from https://sourceforge.net/projects/stoplicht/

Kotusevski, G., Hawick, K.A. (2009). A Review of Traffic Simulation Software. Retrieved from https://www.researchgate.net/publication/228966705_A_Review_of_Traffic_Simulation_Software

Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 3(3), 247–254. https://doi.org/10.1109/JAS.2016.7508798

Liu, H., Oh, J.-S., & Recker, W. (2002). Adaptive Signal Control System with Online Performance Measure for a Single Intersection. Transportation Research Record, 1811(July), 131–138. https://doi.org/10.3141/1811-16

Lu, X.-Y., Varaiya, P., Horowitz, R., Guo, Z., & Palen, J. (2012). Estimating Traffic Speed with Single Inductive Loop Event Data. Transportation Research Record: Journal of the Transportation Research Board, 2308, 157–166. https://doi.org/10.3141/2308-17

Mahmud, K., & Town, G. E. (2016, June 15). A review of computer tools for modeling electric vehicle energy requirements and their impact on power distribution networks. Applied Energy. Elsevier Ltd. https://doi.org/10.1016/j.apenergy.2016.03.100

MASSDOT (2006) Project Development & Design Guide (Guidebook). Retrieved from https://www.massdot.state.ma.us/Portals/8/docs/designGuide/CH_6.pdf

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. https://doi.org/10.1038/nature14236

NEMA (2018) The NTCIP protocol page. Retrieved from https://www.nema.org/Technical/Pages/NTCIP.aspx

Ohno, K., & Mine, H. (1973). Optimal traffic signal settings-II. A refinement of Webster's method. Transportation Research, 7(3), 269–292. https://doi.org/10.1016/0041-1647(73)90018-X

Sanchez-Medina, J. J., Galan-Moreno, M. J., & Rubio-Royo, E. (2010). Traffic signal optimization in la Almozara District in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. IEEE Transactions on Intelligent Transportation Systems, 11(1), 132–141. https://doi.org/10.1109/TITS.2009.2034383

Sutton, R. S., & Barto, A. G. (1998). Reinforcement Learning: An Introduction. Cambridge, Mass: A Bradford Book.

TrafficSimulation-Wikipedia (2018) Traffic simulation. Retrieved from https://en.wikipedia.org/wiki/Traffic_simulation

TSIS-CORSIM (2018) The CORSIM project page. Retrieved from https://mctrans.ce.ufl.edu/mct/index.php/tsis-corsim/

Webster, F. V. (1958). Traffic signal settings. Road Research Technical Paper, No.39, Road Research Laboratory, London, 44.

Wen, W. (2008). A dynamic and automatic traffic light control expert system for solving the road congestion problem. Expert Systems with Applications, 34(4), 2370–2381. https://doi.org/10.1016/j.eswa.2007.03.007

Wiering, M., & Koopman, A. (2004). Intelligent Traffic Light Control. Transportation Research, 53(2004-029), 40–41. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.5180&amp;rep=rep1&amp;type=pdf

Software used or quoted in this work

Arena Simulation (2017), The Arena Simulation project page. Retrieved from https://www.arenasimulation.com/

OpenAI (2018) – The Open AI project page. Retrieved from https://gym.openai.com/

SimPy (2018), The SimPy Project page Retrieved from https://bitbucket.org/simpy/simpy/

SUMO-GitRepo, German Aerospace Center, Institute of Transportation Systems, SUMO GitHub repository (2018) https://github.com/eclipse/sumo

SUMO-ProjectPage, German Aerospace Center, Institute of Transportation Systems, SUMO project page (2018) http://sumo.dlr.de/index.html

The Matplotlib development team (2018), Matplotlib project page. Retrieved from https://matplotlib.org/

The NumPy developers (2018) Numpy project page. Retrieved from http://www.numpy.org/

The Pandas development team (2018) The Pandas project page. Retrieved from http://pandas.pydata.org/