NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
"COMPUTER SCIENCE"

MASTER THESIS

Comparative study on classical and modern ways of traffic signal control with the use of a simulator

Nikolaos K. Tsiougkranas
Alexandros C. Niarchos

***Supervisor*:** **Chamodrakas Ioannis,** Laboratory teaching staff

Athens

June 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΠΛΗΡΟΦΟΡΙΚΗ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Συγκριτική μελέτη κλασικών και σύγχρονων τεχνικών ελέγχου φωτεινών σηματοδοτών μέσω προσομοιωτή

Νικόλαος Κ. Τσιουγκράνας
Αλέξανδρος Χ. Νιάρχος

***Επιβλέπων*:** **Χαμόδρακας Ιωάννης,** Εργαστηριακό διδακτικό προσωπικό

ΑΘΗΝΑ

Ιούνιος 2023

MASTER'S THESIS

Comparative study on classical and modern ways of traffic signal control with the use of a simulator

Nikolaos K. Tsiougkranas

S.N.: cs3200004

Alexandros X. Niarchos

S.N.: cs3200002

Supervisor: **Chamodrakas Ioannis,** Laboratory teaching staff

Examination committee: **Emiris Ioannis,** Professor
Zisimopoulos Vasilis, Professor

Ιούνιος 2023

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Συγκριτική μελέτη κλασικών και σύγχρονων τεχνικών ελέγχου φωτεινών σηματοδοτών μέσω προσομοιωτή

Νικόλαος Κ. Τσιουγκράνας

Α.Μ.: cs3200004

Αλέξανδρος Χ. Νιάρχος

Α.Μ.: cs3200002

ΕΠΙΒΛΕΠΩΝ: **Χαμόδρακας Ιωάννης,** Εργαστηριακό διδακτικό προσωπικό

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Εμίρης Ιωάννης,** Καθηγητής
Ζησιμόπουλος, Βασίλειος, Καθηγητής

Ιούνιος 2023

ABSTRACT

Traffic signal control is an important challenge in urban planning and transport management, with significant implications for congestion, vehicle emissions, and travel times. Traditional traffic control algorithms are based on specific rules, static, and are not able to adapt to the different situations of traffic flow. This has led to the exploration of dynamic and machine learning algorithms to optimize traffic signal timings. This study presents a comparative analysis of five traffic control algorithms - classical methods like Webster's method, Greenwave, and Maxband, and more modern methods like Max Pressure and a DQN model, across a variety of traffic scenarios.

The implementation of classical methods is relatively straightforward. Webster's method aims to minimize delay by adjusting the cycle length of signals based on the critical lane volume and the saturation flow rate. Greenwave facilitates the coordination of traffic signals on arterial roads to create a "green wave" of successive green lights for vehicles. Maxband also optimizes arterial roads by adjusting green times to create maximum throughput.

The modern methods take a different approach. Max Pressure uses real-time data to adjust signal timings, favoring directions with larger vehicle queues. Meanwhile, the DQN model leverages reinforcement learning(RL) to learn optimal traffic signal timings based on traffic states, actions, and corresponding rewards. The DQN model's architecture includes an input layer that receives traffic state representation, hidden layers for capturing complex correlations between traffic states and actions, and an output layer for predicting optimum values for specific actions.

Evaluation of these methods is conducted through simulations using the SUMO traffic simulator across seven scenarios, ranging from a simple single intersection to a complex urban grid which resembles Manhattan. Performance metrics, including total delay and total simulation time for all the vehicles to exit the grid, are used to compare these methods.

Results reveal that the DQN method has better results over all other methods across all scenarios, demonstrating superior efficiency even in less complicated situations and showing flexibility in dealing with high complexity and traffic volume. DQN's adaptability and learning capability allow it to optimize signals based on past experiences and adaptive traffic flow dynamics, proving it to be a promising method for future traffic management.

While classical methods have their advantages, especially in certain circumstances, modern AI-driven techniques like DQN could lead to more efficient traffic management and reduction in waiting times. This study's findings indicate a potential shift in traffic signal control towards machine learning-based methods. However, further research is needed to investigate DQN's performance under other traffic scenarios and to determine how to integrate such systems into real-world traffic management with a focus also on the cost for installing and maintaining the system.

SUBJECT AREA: Transportation Engineering

KEYWORDS: traffic signal control, adaptive traffic management, machine learning, deep reinforcement learning, traffic flow optimization, urban traffic congestion, intelligent transportation systems, traffic simulation, traffic scenario analysis, comparative study, intersection control, queue length minimization, traffic delay reduction, traffic signal phase optimization

ΠΕΡΙΛΗΨΗ

Ο έλεγχος των φωτεινών σηματοδοτών κυκλοφορίας αποτελεί ένα καίριο πρόβλημα στον σχεδιασμό των πόλεων και στη διαχείριση των μεταφορών, με σημαντικές επιπτώσεις στη συμφόρηση, στις εκπομπές οχημάτων και στους χρόνους ταξιδιού. Οι παραδοσιακοί αλγόριθμοι είναι βασισμένοι σε κανόνες, συχνά στατικοί και ανίκανοι να προσαρμοστούν στην δυναμική κίνηση των οχημάτων. Αυτό οδήγησε στην προσπάθεια υλοποίησης αλγορίθμων μηχανικής μάθησης για την βελτιστοποίηση των χρόνων των φωτεινών σηματοδοτών. Η παρούσα μελέτη παρουσιάζει μια συγκριτική ανάλυση πέντε αλγορίθμων ελέγχου της κυκλοφορίας - παραδοσιακών μεθόδων όπως η μέθοδος Webster, ο Greenwave και ο Maxband, μαζί με σύγχρονες μεθόδους όπως ο Max Pressure και ένα μοντέλο μηχανικής μάθησης DQN - σε διαφορετικά σενάρια κυκλοφορίας.

Η υλοποίηση των παραδοσιακών μεθόδων είναι σχετικά απλή. Η μέθοδος Webster έχει ως στόχο στην ελαχιστοποίηση της καθυστέρησης των οχημάτων προσαρμόζοντας το μήκος του κύκλου των σηματοδοτών κυκλοφορίας βάσει του όγκου της κίνησης και του κορεσμού σε έναν δρόμο. Ο Greenwave διευκολύνει τον συντονισμό των φωτεινών σηματοδοτών στις αρτηριακές οδούς προκειμένου να δημιουργηθεί ένα "πράσινο κύμα" συνεχόμενων πράσινων φωτεινών σηματοδοτών για τα οχήματα. Ο Maxband βελτιστοποιεί επίσης τις αρτηριακές οδούς προσαρμόζοντας τους χρόνους των πράσινων φάσεων για να επιτευχθεί η μέγιστη δυνατή ροή στην κυκλοφορία.

Οι σύγχρονες μέθοδοι ακολουθούν διαφορετική προσέγγιση. Ο Max Pressure χρησιμοποιεί πραγματικά δεδομένα και σένσορες για να προσαρμόζει τους χρόνους των σηματοδοτών, δίνοντας προτεραιότητα σε κατευθύνσεις με υψηλή συγκέντρωση οχημάτων. Αντίθετα, το μοντέλο DQN αξιοποιεί την ενισχυτική μάθηση για να μάθει τους βέλτιστους χρόνους των φωτεινών φάσεων βάσει των καταστάσεων της κυκλοφορίας, των ενεργειών και των αντίστοιχων ανταμοιβών. Η αρχιτεκτονική του μοντέλου DQN περιλαμβάνει ένα επίπεδο εισόδου που λαμβάνει την αναπαράσταση της κατάστασης της κυκλοφορίας, κρυφά επίπεδα για την καταγραφή πολύπλοκων συσχετίσεων μεταξύ των καταστάσεων της κυκλοφορίας και των ενεργειών, και ένα επίπεδο εξόδου για την πρόβλεψη των καλύτερων δυνατών τιμών για συγκεκριμένες ενέργειες.

Η αξιολόγηση αυτών των μεθόδων πραγματοποιείται μέσω προσομοιώσεων χρησιμοποιώντας το πρόγραμμα προσομοίωσης κυκλοφορίας SUMO σε επτά σενάρια, που κυμαίνονται από ένα απλό μονό σημείο διασταύρωσης έως ένα πολύπλοκο αστικό πλέγμα που αντιγράφει μια περιοχή στο Μανχάταν. Χρησιμοποιούνται μετρικές απόδοσης, όπως ο συνολικός χρόνος καθυστέρησης και ο συνολικός χρόνος προσομοίωσης, για να γίνει η σύγκριση αυτών των μεθόδων.

Τα αποτελέσματα αποκαλύπτουν ότι ο DQN είχε καλύτερες επιδόσεις από όλες τις άλλες μεθόδους σε όλα τα σενάρια, επιδεικνύοντας υψηλή αποδοτικότητα και ευελιξία ακόμη και σε απλές καταστάσεις κίνησης και επιδεικνύοντας ανθεκτικότητα σε υψηλή πολυπλοκότητα και όγκο κυκλοφορίας. Η προσαρμοστικότητα και η δυνατότητα μάθησης του DQN του επιτρέπουν να βελτιστοποιεί τα σήματα βάσει προηγούμενων εμπειριών και της δυναμικής της κυκλοφορίας, καθιστώντας το έναν ελπιδοφόρο τρόπο για τη μελλοντική διαχείριση της κυκλοφορίας.

Παρά τα πλεονεκτήματα των παραδοσιακών μεθόδων, ειδικά σε συγκεκριμένες περιπτώσεις, οι σύγχρονες τεχνικές που βασίζονται σε τεχνητή νοημοσύνη, όπως ο DQN, θα μπορούσαν να οδηγήσουν σε πιο αποτελεσματική διαχείριση της κυκλοφορίας και μείωση των χρόνων αναμονής. Τα ευρήματα αυτής της μελέτης υποδεικνύουν μια δυνητική αλλαγή στον έλεγχο των φωτεινών σηματοδοτών κυκλοφορίας προς μεθόδους που βασίζονται σε μηχανική μάθηση. Ωστόσο, απαιτείται περαιτέρω έρευνα για να εξεταστεί η

απόδοση του DQN σε άλλα σενάρια κυκλοφορίας και του τρόπου ενσωμάτωσης αυτών των συστημάτων στην πραγματική διαχείριση της κυκλοφορίας και να δοθεί έμφαση και στο κόστος για εγκατάσταση και συντήρηση των συστημάτων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μηχανική Μεταφορών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Έλεγχος φωτεινών σηματοδοτών κυκλοφορίας, προσαρμοστική διαχείριση κυκλοφορίας, μηχανική μάθηση, βαθιά ενισχυτική μάθηση, βελτιστοποίηση ροής κυκλοφορίας, αστική συμφόρηση, έξυπνα συστήματα μεταφορών, προσομοίωση κυκλοφορίας, ανάλυση σεναρίων κυκλοφορίας, συγκριτική μελέτη, έλεγχος διασταυρώσεων, ελαχιστοποίηση μήκους ουρών, μείωση καθυστέρησης κυκλοφορίας, βελτιστοποίηση φάσεων φωτεινών σηματοδοτών.

ΠΕΡΙΕΧΟΜΕΝΑ

IMAGE CATALOG

PROLOGUE

This thesis is conducted as part of the “Computer Science” Master’s program of the National and Kapodistrian University of Athens.

As the world's population continues to grow and due to the urbanization of most countries, the efficient management of transportation is becoming more important. Traffic congestion, air pollution, and road accidents are among the most pressing issues faced by cities today. Addressing these challenges requires innovative and effective solutions that will improve traffic flow which will result in higher quality of life and lower environmental impact. One such aspect that is important to the efficient management of transportation systems is traffic signal control. This thesis aims to provide a comparative analysis of classical and modern traffic signal control techniques, using simulation tools to better understand their respective strengths and limitations.

The classical approach to traffic signal control has its roots in predetermined, fixed-time plans, based on historical data. While these methods have contributed significantly to the organization of urban traffic, they may not always be sufficient in addressing the dynamic and complex nature of today's transportation systems.

The modern era has seen the rise of intelligent traffic signal control techniques, which leverage cutting-edge technology, such as artificial intelligence, machine learning, and connected infrastructure, to optimize traffic signal timings dynamically. These innovations hold the potential to significantly enhance traffic flow efficiency, reduce congestion, and decrease the environmental impact of transportation systems.

By employing a simulator, this thesis will explore the performance of various classical and modern traffic signal control strategies under a range of traffic scenarios. The simulation-based approach will enable a better understanding of the practical problems and trade-offs associated with each method. Through this comparative study, we hope to provide valuable insights that can guide the development and implementation of more effective traffic signal control systems in the future.

During this thesis, we will go through history and innovations which will help us analyze and understand the importance of traffic management in shaping sustainable, smart, and efficient cities.

1. INTRODUCTION

The transportation field has evolved greatly over the past century, from the widespread adoption of automobiles and the urbanization process. These changes have had a significant impact on the way cities are planned and managed, with traffic signal control systems playing the most important role in maintaining order and safety on the roads. In this thesis, we explore classical and modern traffic signal control techniques, utilizing a simulator to analyze their performance under various traffic conditions and scenarios.

Chapter 2 sets the foundation for understanding traffic signal control systems by analyzing key parts such as phases and cycles, as well as providing an overview of various classical and modern traffic signal control algorithms. Additionally, it offers an introduction to neural networks, given their significance in modern control techniques. Finally, this chapter includes a review of the relevant literature in this field.

Chapter 3 focuses on the detailed implementation of the various traffic signal control algorithms discussed in the previous chapter. Here, the practical application of these algorithms is revealed across different traffic scenarios. Furthermore, this chapter presents the specifics of the custom-developed DQN implementation.

In Chapter 4 a comprehensive analysis of the extracted results from the various traffic signal control algorithms and the DQN implementation is conducted which leads us to conclusions for the performance of each algorithm in relation to different traffic scenarios.

In Chapter 5, the conclusion of the thesis is discussed. The challenges, limitations, and future potential work are analyzed.

Ultimately, this thesis seeks to contribute to the ongoing research on traffic signal control by providing an in-depth analysis of the different techniques(classical and more modern) available today. By examining their performance through simulation, we hope to offer valuable information that can inform the development and implementation of more effective traffic signal control systems in the future, leading to more efficient, eco-friendly, and sustainable urban environments.

2. HISTORY AND EVOLUTION OF TRAFFIC SIGNAL CONTROL SYSTEMS

Traffic signal control systems have played a crucial role in the management of urban traffic and have evolved significantly over time. From ancient methods of controlling traffic flow in huge cities in different cultures like Rome, Greece, China, and Egypt to the development of classical and modern algorithms that optimize signal timings, the history of traffic signal control systems is rich and diverse. In this chapter, we will briefly explore the history of traffic signal control systems, starting with their earliest origins and continuing through the classical and modern methods that are widely used today. We will discuss the various algorithms and techniques that have been developed to address the challenges of traffic management and improve traffic efficiency, focusing on their implementation and potential benefits.

2.1 History of traffic control

Before analyzing the classical and modern traffic signal control systems, it is crucial to understand the historical context from which these methods evolved. Traffic control has been necessary since the emergence of organized human settlements, with ancient and medieval societies devising innovative means to regulate the flow of pedestrians, animals, and vehicles. This sub-chapter will explore the origins of traffic signal control, focusing on the early attempts at managing traffic in ancient and medieval times, as well as the development of traffic signal controls from the 1800s to the present day.

The ancient world saw the rise of great civilizations, such as the Egyptians, Greeks, Romans, and Chinese which were characterized by bustling urban centers and complex road networks. Despite the absence of modern automobiles, these societies faced traffic challenges that necessitated the implementation of control systems[1-11]. In medieval times, cities and towns experienced significant growth, resulting in the need for traffic control systems to manage the movement of people, animals, and vehicles. Although these systems were not as advanced as modern traffic signals, several methods and regulations were implemented to maintain order and ensure the efficient flow of traffic[12-14]. As technology advanced, traffic signal controls evolved significantly since the 1800s, with interconnected systems, traffic signal controllers, and modern traffic signal control systems being developed to improve traffic flow and minimize congestion[15-18].

2.2 Basic Components of Classical Methods of Traffic Signal Control

In this chapter, we will delve deeper into classical methods of traffic signal control, examining their origins, applications, and the underlying principles that have shaped traffic management strategies over time. We will discuss fixed-time control and actuated control, highlighting their respective strengths and weaknesses, as well as the factors that have influenced their adoption and continued use in various traffic management scenarios.

By understanding these classical methods and their impact on traffic signal control, we can gain valuable insights into the factors that drive effective traffic management and the continued development of advanced traffic control systems. This understanding will provide a solid foundation for the analysis of modern and future traffic control technologies, which will be the focus of the subsequent chapters of this thesis.

2.2.1 Traffic Light Cycle

A traffic light cycle, also known as a signal cycle or cycle length, is the complete sequence of signal phases at a traffic light-controlled intersection, from the start of the green phase for one movement direction to the time when the same green phase is repeated. The traffic light cycle encompasses all the green, yellow, and red phases for each movement direction at an intersection, ensuring that vehicles and pedestrians are provided with the necessary time to safely navigate the intersection[18].

2.2.2 Traffic Light Phases

Traffic light phases, also known as signal timing, are a crucial aspect of traffic signal control systems. They determine the sequence and duration of each color displayed by the traffic lights at an intersection. An effective phase plan helps to ensure the smooth and safe flow of traffic, minimize congestion, and optimize overall efficiency. This section will provide a detailed explanation of the various traffic light phases and their functions[18].

1. The green phase is the period during which the correlated traffic is allowed to proceed through the intersection in a specific direction. The duration of the green phase can vary depending on the specific requirements of the intersection, such as traffic volume, speed limits, and the presence of pedestrians or cyclists. In classical algorithms this phase has fixed time but in more modern approaches the green phase may be extended or shortened in real-time based on traffic conditions, using traffic sensors or cameras [18].
2. The yellow phase serves as a transition between the green and red phases. It warns drivers that the signal is about to change to red, providing them with enough time to stop safely. The duration of the yellow phase is typically determined based on the speed limit and the width of the intersection, ensuring that drivers have sufficient time to stop [18].
3. The red phase is the period during which traffic must stop. Its primary purpose is to ensure safety by preventing collisions between vehicles traveling in different directions. The duration of the red phase may vary based on factors such as traffic volume, intersection geometry, and the presence of pedestrians or cyclists. In some cases, the red phase may include an "all-red" interval, during which all approaches to the intersection display a red signal to provide an additional safety buffer between conflicting phases[18].
4. The pedestrian phase is designed to provide safe crossing opportunities for pedestrians at signalized intersections. It typically includes a "walk" indication, during which pedestrians are allowed to enter the crosswalk, and a "flashing don't walk" indication, which warns pedestrians that the pedestrian phase is ending and they should not start crossing. The duration of the pedestrian phase is determined based on factors such as pedestrian crossing distance and walking speed, ensuring that pedestrians have enough time to safely cross the street. For this thesis, this phase will not be taken into consideration and analyzed further[18].
5. The left-turn phase, or right-turn phase for countries with opposite driving such as England, also known as a protected left-turn phase, is designed to facilitate safe and efficient left-turn movements at signalized intersections. It provides a dedicated green arrow signal for left-turning vehicles, allowing them to turn without conflicting with oncoming traffic or pedestrians. The left-turn phase can

be configured as leading (before the through movement), lagging (after the through movement), or split (separate from the through movement). The specific configuration and duration of the left-turn phase depend on factors such as traffic volume, intersection geometry, and safety considerations [18].

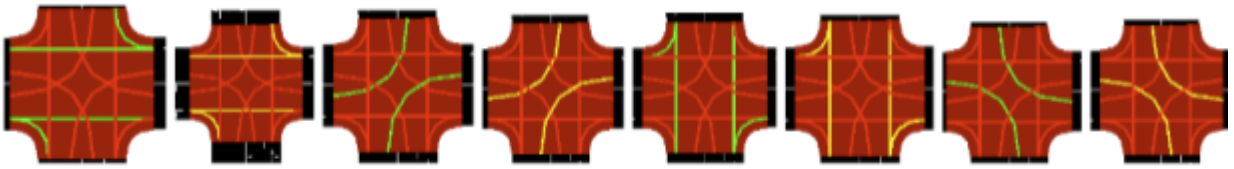

Image 1: All possible phases of a four-way intersection

2.2.3 Fixed-Time Control Traffic Light System

Fixed-time control, also known as pre-timed control or fixed-cycle control, is a classical method of traffic signal control in which the green, yellow, and red phases are displayed for predetermined durations at each intersection approach. These durations, or cycle lengths, are typically calculated based on historical traffic patterns, peak demand periods, and other relevant factors. Fixed-time control systems are relatively simple to design, implement, and maintain, making them a popular choice for traffic management in various contexts.

In a fixed-time control system, the cycle length, green time allocation, and phase sequence are pre-determined and remain constant throughout the day, regardless of fluctuations in traffic demand. The cycle length is typically divided into different phases, with each phase providing green time for one or more movement directions [18]. The green time allocation is determined based on factors such as traffic volume, pedestrian demand, and intersection geometry.

To optimize traffic flow, fixed-time control systems often use different cycle lengths and green time allocations for different times of the day, such as morning and evening peak periods or off-peak hours. These changes can be implemented manually or automatically using a pre-programmed schedule.

Fixed-time control traffic light systems offer several advantages, including:

1. Simplicity: The straightforward design and operation of fixed-time control systems make them relatively easy to implement, maintain, and troubleshoot.
2. Predictability: The consistent cycle lengths and green time allocations provide predictable traffic patterns, which can help drivers anticipate signal changes and reduce the likelihood of accidents.
3. Efficiency: When properly designed, fixed-time control systems can efficiently manage traffic flow during predictable demand periods, such as morning and evening rush hours.

Despite their advantages, fixed-time control traffic light systems also have some limitations, including:

1. Inflexibility: Fixed-time control systems are unable to adapt to real-time changes in traffic demand, which can lead to inefficiencies during periods of unexpected congestion or low traffic volumes.
2. Increased Delay: Due to their inflexible nature, fixed-time control systems may cause unnecessary delays for drivers and pedestrians, particularly during off-peak hours when traffic volumes are lower.

3. Environmental Impact: The lack of adaptability in fixed-time control systems can result in increased idling time, fuel consumption, and vehicle emissions.

2.2.4 Actuated Control Traffic Light System

Actuated control is a more dynamic method of traffic signal control that adjusts signal timings in response to real-time traffic conditions. By using sensors, such as inductive loops, video cameras, or microwave detectors, actuated control systems can detect the presence of vehicles and pedestrians and adjust the green time allocation accordingly. This adaptive approach allows for more efficient traffic management, reducing delays and improving overall traffic flow.

In an actuated control traffic light system, sensors are installed at each intersection approach to detect the presence and demand of vehicles and pedestrians. When a sensor detects a vehicle or pedestrian waiting at the intersection, it sends a signal to the traffic signal controller, which then adjusts the green time allocation for the corresponding phase.

There are two primary types of actuated control systems: semi-actuated and fully actuated.

Semi-actuated control systems: These systems use sensors on the minor street approaches and pedestrian crossings, while the major street operates on a fixed-time basis. Semi-actuated control systems are commonly used at intersections where the traffic volume on the minor street is significantly lower than that on the major street.

Fully actuated control systems: In fully actuated control systems, sensors are installed on all approaches and pedestrian crossings, allowing the signal controller to adapt the green time allocation for all phases based on real-time traffic demand. Fully actuated control systems are typically used at complex intersections with high traffic volumes and variable demand patterns.

Actuated control traffic light systems offer several advantages, including:

1. Responsiveness: By adapting to real-time traffic conditions, actuated control systems can reduce delays and improve traffic flow, particularly during periods of unexpected congestion or low traffic volumes.
2. Efficiency: Actuated control systems can allocate green time more efficiently, prioritizing phases with higher traffic demand and minimizing the time spent waiting at red lights.
3. Environmental benefits: The adaptive nature of actuated control systems can reduce vehicle idling time, fuel consumption, and emissions by minimizing delays and optimizing traffic flow.

Despite their advantages, actuated control traffic light systems also have some limitations, including

1. Complexity: Actuated control systems are more complex than fixed-time systems, requiring the installation, maintenance, and calibration of sensors and communication equipment.
2. Cost: The increased complexity of actuated control systems can result in higher implementation and maintenance costs compared to fixed-time systems.
3. Sensor limitations: Sensor technologies can sometimes be affected by weather conditions, dirt, or vehicle positioning, potentially leading to inaccuracies in traffic detection and signal control[18].

2.3 Classical traffic control strategies

Now that all the basic components are covered we will explore three classical traffic signal control methods that have significantly contributed to the optimization of traffic flow in urban areas. These methods continue to be relevant in various traffic management scenarios. The three classical methods we will discuss in this thesis are the Webster Algorithm, Greenwave, and MAXBAND.

2.3.1 Webster Algorithm

The Webster Algorithm is a widely used method for calculating the optimal cycle lengths and green time allocations for fixed-time cycles. The algorithm aims to maximize the traffic flow by minimizing the total delay at an intersection. This algorithm continues to be a very relevant and popular method for traffic signal timing optimization, especially for isolated intersections or simple traffic networks[18][19].

It calculates the optimal cycle length (C) and green time allocation for each phase (G) using the following steps:

1. Determine the saturation flow rate (S) for each approach or movement direction at the intersection. The saturation flow rate is the maximum number of vehicles that can pass through the intersection per unit of time when the signal is green.
2. Calculate the flow ratio (v/S) for each approach or movement direction, which represents the demand for green time as a proportion of the saturation flow rate.
3. Compute the lost time (L) for each phase. Lost time accounts for the time wasted during the start and end of each green phase when vehicles are accelerating or decelerating. The lost time is typically estimated using field observations or derived from engineering guidelines.
4. Determine the total flow ratio (Σ(v/S)) and the total lost time (ΣL) for the intersection by summing the flow ratios and lost times for all approaches or movement directions.
5. Calculate the optimal cycle length (C) using the following formula: $C = 1.5 * L + 5 / (1 - \Sigma(v/S))$
6. Compute the optimal green time allocation (G) for each phase by multiplying the cycle length (C) with the flow ratio (v/S) for the respective approach or movement direction.

The calculated cycle length and green time allocations can be implemented in a fixed-time traffic signal control system to optimize the traffic flow and minimize delays at the intersection.

Despite its widespread use, the Webster Algorithm has some limitations:

1. Applicability: The algorithm is primarily suitable for isolated intersections or simple traffic networks. In more complex networks or situations where coordinated signal timings are required, alternative optimization methods may be more appropriate.
2. Assumptions: The Webster Algorithm assumes that traffic demand remains constant throughout the cycle and does not account for variations in demand or fluctuations in traffic conditions.

3. Fixed-time control: As the algorithm is designed for fixed-time traffic signal control systems, it does not consider the potential benefits of adaptive or traffic-responsive control methods

2.3.2 Greenwave

Greenwave traffic control, also known as traffic signal coordination or progression, is a classical traffic signal control method that synchronizes traffic signals along a corridor or arterial road to create a continuous "green wave" for vehicles. By coordinating signal timings, the green wave allows vehicles to travel through a series of intersections with minimal stops, reducing delays, and improving overall traffic flow. Greenwave traffic control is particularly effective in managing traffic along corridors with a dominant direction of flow, such as in downtown areas or during peak hours[18].

The Greenwave traffic control strategy is based on adjusting signal offsets, which are the time differences between the start of the green phase at adjacent intersections. The key steps in implementing a Greenwave traffic control strategy are:

1. Determine the critical direction of flow: Identify the direction of flow with the highest traffic demand, which will be the primary focus of the green wave.
2. Calculate travel time between intersections: Measure or estimate the time it takes for a vehicle to travel between adjacent intersections at the desired speed or posted speed limit.
3. Set cycle length: Establish a common cycle length for all intersections along the corridor. This can be determined using optimization methods, such as the Webster Algorithm, or through engineering guidelines and field observations.
4. Adjust signal offsets: Determine the optimal signal offset for each intersection along the corridor based on the travel time between intersections. The offset should be set such that when a vehicle arrives at the downstream intersection, the green phase for the critical direction of flow is just beginning.
5. Implement and monitor: Apply the calculated cycle lengths and signal offsets to the traffic signal controllers and monitor the system's performance to ensure the green wave is functioning as intended. Adjustments may be needed to account for changes in traffic patterns or demand.

Greenwave traffic control offers several benefits, including:

1. Reduced delays: By synchronizing traffic signals, Greenwave traffic control reduces the number of stops and delays for vehicles traveling along the corridor, improving overall travel time.
2. Improved traffic flow: The green wave facilitates smoother traffic flow along the corridor, which can help mitigate congestion and increase the capacity of the road network.
3. Enhanced fuel efficiency and reduced emissions: With fewer stops and more consistent travel speeds, vehicles can operate more efficiently, resulting in reduced fuel consumption and lower emissions.
4. Improved safety: Smooth traffic flow reduces the likelihood of rear-end collisions due to sudden stops or erratic driving behavior.

Despite its advantages, Greenwave traffic control also has some limitations:

1. Limited applicability: Greenwave traffic control is most effective along corridors with a dominant direction of flow, and its benefits may be diminished in areas with complex traffic patterns or balanced traffic demands.
2. Fixed-time control: Greenwave traffic control relies on fixed-time signal timings and does not account for real-time traffic fluctuations, limiting its adaptability to changing traffic conditions.
3. Potential side-street delay: The focus on optimizing signal timings for the main corridor may result in increased delays for side-street traffic, as they may have to wait longer for a green phase.

2.3.3 MAXBAND

MAXBAND is a classical traffic signal control method developed by Professors Nathan H. Gartner and Nicos Geroliminis in the 1970s. It focuses on coordinating traffic signals along an arterial road or corridor to maximize the bandwidth of green time for the critical direction of flow. By optimizing the green time bandwidth, the MAXBAND method aims to minimize stops and delays for vehicles traveling along the corridor, resulting in smoother traffic flow and improved overall efficiency[20][21].

MAXBAND utilizes an optimization algorithm to determine the optimal signal offsets and cycle lengths for a given corridor. The key steps in implementing MAXBAND are as follows:

1. Determine the critical direction of flow: Identify the direction of flow with the highest traffic demand, which will be the primary focus of the green-phase bandwidth optimization.To identify the critical direction, it is necessary to consider the travel demand patterns in both the inbound and outbound directions. During peak hours, the inbound roads may have a higher travel demand as people travel to work or other central destinations. In contrast, during the evening rush hour, the outbound roads may have a higher travel demand as people head home or leave the central area.
2. Establish a common cycle length: Determine a common cycle length for all intersections along the corridor. This can be calculated using optimization methods such as the Webster Algorithm, or through engineering guidelines and field observations.
3. Calculate the optimal signal offsets: Using the MAXBAND optimization algorithm, determine the signal offsets that maximize the bandwidth of green time for the critical direction of flow along the corridor. The optimization process considers both inbound and outbound roads, adjusting the signal offsets and cycle lengths to create a continuous green wave in the critical direction. This coordination can be applied separately for different periods, addressing the varying travel demands between inbound and outbound roads during peak and off-peak hours.
4. Implement and monitor: Apply the calculated cycle lengths and signal offsets to the traffic signal controllers, and monitor the system's performance to ensure that the MAXBAND strategy is functioning as intended. Adjustments may be needed to account for changes in traffic patterns or demand.

The MAXBAND traffic control method offers several benefits, including:

1. Reduced delays: By maximizing the bandwidth of green time along the corridor, MAXBAND minimizes stops and delays for vehicles traveling in the critical direction, improving overall travel time.

2. Improved traffic flow: The optimized signal coordination achieved with MAXBAND facilitates smoother traffic flow along the corridor, helping to mitigate congestion and increase the capacity of the road network.
3. Enhanced fuel efficiency and reduced emissions: With fewer stops and more consistent travel speeds, vehicles can operate more efficiently, resulting in reduced fuel consumption and lower emissions.

Despite its advantages, the MAXBAND method also has some limitations:

1. Limited applicability: MAXBAND is most effective along corridors with a dominant direction of flow and may be less effective in areas with complex traffic patterns or balanced traffic demands.
2. Fixed-time control: Like the Greenwave method, MAXBAND relies on fixed-time signal timings and does not account for real-time traffic fluctuations, limiting its adaptability to changing traffic conditions.
3. Potential side-street delay: Similar to the Greenwave method, the focus on optimizing signal timings for the main corridor may result in increased delays for side-street traffic, as they may have to wait longer for a green phase.

2.3.4 Conclusion

In conclusion, classical traffic control methods have played a significant role in managing traffic flow and reducing congestion in urban environments. These methods, such as the Webster algorithm, Greenwave, and MAXBAND, have provided the foundation for our understanding of traffic signal control systems and their impact on urban mobility.

Throughout this chapter, we have explored the principles, methodologies, advantages, and limitations of these classical methods. We have seen how fixed-time control provides simplicity and ease of implementation, while actuated and traffic-responsive control methods offer more adaptability to real-time traffic conditions. Additionally, signal coordination strategies, such as the Webster algorithm, Greenwave, and MAXBAND, have demonstrated their effectiveness in improving traffic flow and reducing delays along arterial corridors.

However, we have also identified several limitations of classical traffic control methods, including their reliance on pre-determined signal timings, no adaptability to changing traffic patterns in different scenarios, and significant inefficiencies in managing complex urban road networks. These limitations have boosted the development of modern traffic control systems, which leverage sensors, and intelligent algorithms to enhance overall network efficiency.

2.4 Modern Methods of Traffic Signal Control

In the modern era of rapid urbanization when the number of vehicles in big cities has escalated, the need for more modern and adaptive traffic signal control systems has skyrocketed. These systems use traffic sensors, clever algorithms, and more in order to improve their adaptability over varying traffic scenarios.

We will begin by examining several modern traffic control systems in detail, including the MAX-Pressure algorithm and the DQN algorithm. Throughout this chapter, we will discuss the advantages of these modern systems, such as their ability to adapt and generally improve the overall network efficiency. However, we will also address their

limitations, such as dependency on sensors, computational complexity, and potential challenges in a real-world implementation, which included the cost of installation and maintenance.

2.4.1 Traffic-Responsive Control

Traffic-responsive control is an advanced method of traffic signal control that not only adapts to real-time traffic conditions but also considers broader network-wide traffic patterns. By continuously monitoring and analyzing traffic data from multiple intersections, traffic-responsive control systems can optimize signal timings to improve overall network performance. This strategic approach helps to reduce congestion, minimize delays, and enhance the efficiency of the entire transportation system.

Traffic responsive control systems rely on a central traffic management center, which collects and processes traffic data from various sources, such as sensors, cameras, and traffic counters, across multiple intersections. The system employs advanced algorithms and optimization techniques to determine the most efficient signal timings based on the current traffic conditions and anticipated demand patterns.

The central traffic management center can then send updated signal timings to individual traffic signal controllers at each intersection. This coordination among intersections enables traffic-responsive control systems to optimize traffic flow and reduce the impact of congestion on the broader network.

There are several traffic responsive control strategies, including:

1. Adaptive signal control: This strategy uses real-time traffic data to adjust signal timings at individual intersections, ensuring that each intersection operates at maximum efficiency.
2. Traffic signal coordination: This strategy synchronizes signal timings among a series of intersections, creating a "green wave" effect that allows vehicles to travel through multiple intersections with minimal stops.
3. Traffic network optimization: This strategy optimizes signal timings across the entire traffic network, considering the interaction between intersections and the broader traffic patterns to maximize overall network performance.

Traffic responsive control traffic light systems offer several advantages, including:

1. Network-wide optimization: By considering the entire traffic network, traffic responsive control systems can maximize overall efficiency, reducing delays and congestion across multiple intersections.
2. Real-time adaptability: Traffic responsive control systems can quickly adapt to changing traffic conditions, ensuring that signal timings remain optimized even during unexpected events, such as incidents or special events.
3. Enhanced performance: By optimizing signal timings and coordinating multiple intersections, traffic-responsive control systems can improve overall traffic flow, reduce travel times, and enhance the overall performance of the transportation system.

Despite their advantages, traffic responsive control traffic light systems also have some limitations, including:

1. Complexity: Traffic responsive control systems require a sophisticated central traffic management center, advanced algorithms, and communication

infrastructure, making them more complex and challenging to implement and maintain than traditional control methods.

2. Cost: The increased complexity of traffic-responsive control systems can result in higher implementation, maintenance, and operational costs compared to other traffic control methods.
3. Data reliance: Traffic-responsive control systems rely heavily on accurate and timely traffic data, making them susceptible to inaccuracies or failures in data collection and communication systems.

2.4.2 MAX-Pressure Traffic Control Algorithm

The MAX-Pressure traffic control algorithm is a modern, adaptive traffic signal control method that aims to minimize the total queue length across all approaches at an intersection. Developed by Pravin Varaiya and his colleagues, this method is based on a distributed control strategy and dynamically adjusts signal timings in response to real-time traffic conditions. By minimizing the total queue length, the MAX-Pressure algorithm can reduce delays, improve traffic flow, and enhance the overall efficiency of urban road networks.

To continue into the better understanding of the Max Pressure algorithm, the mathematical description of the algorithm will be analyzed.

Let's denote the set of intersections as I, and the set of all links (or approaches) as L. Each intersection $i \in I$ is associated with a set of incoming links Li_{in} and a set of outgoing links Li_{out}. Traffic sensors collect the vehicle counts $n_l(t)$ for each link $l \in L$ at each time step t. The pressure $P_l(t)$ for each link $l \in L$ at each time step t is calculated as the difference between the cumulative inflow and outflow:

$$P_l(t) = \sum(a \in Li_{in} [n_a(t)]) - \sum(b \in Li_{out} [n_b(t)])$$

where $n_a(t)$ and $n_b(t)$ represents the cumulative number of vehicles arriving at the upstream approach a and departing from the downstream approach b, respectively.

The signal timings are updated at each time step t to prioritize the link l with the maximum pressure:

$$l = \text{argmax}_l \{P_l(t)\}$$

The green phase is extended or granted to the link l to facilitate vehicle departure.

The MAX-Pressure algorithm uses traffic sensor data, such as vehicle counts and occupancy measurements, to calculate the "pressure" at each approach of an intersection. The pressure, as described earlier, is defined as the difference between the cumulative number of vehicles arriving at the upstream approach and the cumulative number of vehicles departing from the downstream approach. The algorithm updates signal timings to prioritize the approach with the highest pressure, thereby minimizing the total queue length at the intersection. The main steps in implementing the MAX-Pressure algorithm are as follows:

1. Obtains real-time traffic measurements from sensors to monitor vehicle arrivals and departures at each approach of the intersection.
2. For each phase of the traffic light, calculates the pressure by taking the difference between the number of vehicles arriving at the upstream approach and the number of vehicles departing from the downstream approach.

3. At each time step, compares the pressures for all approaches and prioritizes the approach with the highest pressure by extending or granting its green phase. Adjust the signal timings dynamically to minimize the total queue length at the intersection.
4. Continuously monitor the performance of the MAX-Pressure algorithm and adjusts the parameters as needed to account for changes in traffic patterns or demand.

The MAX-Pressure traffic control algorithm offers several benefits, including:

1. Adaptability: Unlike fixed-time control methods, such as Greenwave and MAXBAND, the MAX-Pressure algorithm dynamically adjusts signal timings in response to real-time traffic conditions, making it more adaptable to changing traffic patterns and demands.
2. Distributed control: The MAX-Pressure method is based on a distributed control strategy, which allows for efficient and scalable implementation in large-scale urban road networks.
3. Reduced delays and improved traffic flow: By minimizing the total queue length at intersections, the MAX-Pressure algorithm can reduce delays, improve traffic flow, and enhance overall network efficiency.
4. Robustness: The MAX-Pressure algorithm is robust to variations in traffic demand and can effectively handle both under-saturated and over-saturated traffic conditions.
5. Several studies have been conducted to evaluate the performance of the MAX-Pressure algorithm in various traffic scenarios and compared to other traffic control methods. These studies have found that the MAX-Pressure algorithm generally performs well in reducing delays, improving traffic flow, and enhancing overall network efficiency.

Despite its advantages, the MAX-Pressure method also has some limitations:

1. Dependency on traffic sensors: The performance of the MAX-Pressure algorithm depends on the availability and accuracy of real-time traffic sensor data. Inaccurate or missing sensor data may reduce the effectiveness of the algorithm.
2. Computational complexity: The MAX-Pressure algorithm requires the continuous calculation of pressures and dynamic updating of signal timings, which can be computationally intensive, especially for large-scale networks.
3. MAX-Pressure algorithm optimizes an intersection and does not take account of the neighboring intersections. As a result, it is not suitable for complex intersection clusters where neighboring intersections affect each other.

Comparison with other traffic control methods: In several studies, as will be discussed later in detail, the MAX-Pressure algorithm has been compared with other traffic control methods both classical and modern and has shown that it can outperform the other methods by reducing delays and improving traffic flow. Especially in scenarios with very congested intersections.

Large-scale implementation: The MAX-Pressure algorithm has been successfully implemented and tested in large-scale urban road networks, demonstrating its scalability and adaptability to different traffic scenarios. This large-scale implementation also highlights the potential of the MAX-Pressure method in real-world traffic control applications.

Robustness and efficiency: The MAX-Pressure algorithm is robust to variations in traffic demand and can effectively handle both under-saturated and over-saturated traffic conditions. This robustness and efficiency make the MAX-Pressure method a promising traffic control solution for urban road networks.

In conclusion, MAX-Pressure is a serious candidate for modern traffic control systems due to its distributed control and adaptability in different traffic scenarios despite its dependency on traffic sensors and the higher computational complexity [22][23][24].

2.4.3 Deep Q-Networks (DQN) for Traffic Control

In this thesis in order to optimize the traffic signal control problem we implemented a DQN algorithm that utilizes deep reinforcement learning (DRL) techniques. The versatility of DQN is apparent in multiple applications which solve a vast range of problems from games to autonomous vehicles. By using this advanced DRL technique we aim to develop a system that will be able to be trained and address efficiently multiple different traffic scenarios which will reduce traffic congestion and improve the overall network efficiency.

In this section, we will discuss about the DQN method, analyzing its foundations in DRL. We will explain the key concepts of Q-learning, DNNs, and the integration of both techniques to form the DQN. We will then describe the specific DQN architecture and training process used, detailing the state representation, action space and reward function.

Following the introduction, we will present the implementation of the DQN-based traffic control system, discussing the necessary modifications and adaptations to tailor the method for traffic signal control. We will also outline the simulation environment used to evaluate the performance of the DQN solution and compare it to classical and other modern traffic control methods.

2.4.3.1 Q-learning

In order to make clear what DQN is we must first analyze what Q-learning is. It is a non-model-bound, off-policy RL algorithm whose goal is to find the best policy for decision-making problems by learning the Q-values, which in simpler terms are an action-value function. These values are saved in the Q-table. As a result, the agent is enabled to make optimal decisions without the need for a complete model of the environment. The algorithm gets a specific state, estimates the expected reward per action, and makes the action with the most profitable reward[25].

In Q-learning, the agent maintains a Q-table that stores the Q-values for each state-action pair. The Q-values are updated iteratively using the Bellman equation:

$Q(s, a) \leftarrow Q(s, a) + \alpha * [r + \gamma * \max_{a'} Q(s', a') - Q(s, a)]$

- s: current state
- a: action taken
- r: immediate reward received after taking the action
- s': next state after taking the action
- a': possible actions in the next state
- α: learning rate ($0 < \alpha \leq 1$) determining the extent to which new information overrides old information

- γ: discount factor ($0 \leq \gamma < 1$) representing the importance of future rewards compared to immediate ones

The algorithm begins with an arbitrary initialization of Q-values and iteratively refines them through interactions with the environment. The agent selects actions based on the current Q-values, often using an exploration strategy such as ε-greedy to balance exploration and exploitation. As the agent continues to interact with the environment, the Q-values converge to their optimal values, which represent the expected cumulative reward for each state-action pair following the optimal policy.

Once the Q-values have converged, the agent can make optimal decisions by selecting the action with the highest Q-value for each state. Q-learning has been widely applied to various decision-making problems, from robotics to game playing, and has proven to be a powerful and effective RL algorithm[26].

While Q-learning with Q-tables has proven effective in many RL problems, it suffers from a few limitations that make it unsuitable for large-scale or high-dimensional problems:

1. Scalability: The size of the Q-table grows exponentially with the number of states and actions in the environment. For problems with a large state space or many possible actions, the Q-table can quickly become infeasible in terms of memory requirements and computational complexity [26].
2. Generalization: Q-tables store and update Q-values for each individual state-action pair, which means they do not generalize well to unseen states or situations. In contrast, neural networks are capable of learning continuous, smooth functions that can generalize to similar but unseen states, enabling them to handle larger state spaces more effectively [32].
3. Sample efficiency: Q-learning with Q-tables can be slow to converge, as it requires many samples (state-action-reward-next state tuples) to update the Q-values accurately. Neural networks, especially DNNs, can learn complex representations and generalize from a smaller number of samples, leading to faster convergence and better overall performance [29].

To address these limitations, Q-learning can be combined with neural networks, leading to the DQN algorithm [27]. By replacing the Q-table with a neural network, the DQN algorithm can efficiently learn the action-value function in large-scale and high-dimensional problems. The neural network approximates the Q-values, taking the state as input and producing Q-values for each possible action as output. This approach significantly reduces the memory requirements and enables the algorithm to generalize better to unseen states, leading to improved performance and faster convergence in complex environments.

2.4.3.2 Deep Neural Network

A DNN is a type of artificial neural network that consists of multiple layers of interconnected nodes or neurons, which enables the network to learn complex, hierarchical representations of input data [30]. DNNs have gained significant attention in recent years due to their ability to achieve state-of-the-art performance in various domains, including image recognition, natural language processing, and RL [31].

DNNs consist of an input layer, multiple hidden layers, and an output layer. The input layer receives the raw data, while the output layer produces the final predictions or decisions. The hidden layers, which contain the majority of the network's neurons,

transform the input data into increasingly complex and abstract representations through a series of non-linear transformations [32].

Each neuron in a DNN receives input from multiple neurons in the previous layer, computes a weighted sum of these inputs, applies a non-linear activation function, and passes the result to the neurons in the next layer. The weights and biases of the neurons are the learnable parameters of the network, which are adjusted during the training process to minimize the difference between the network's predictions and the ground truth.

DNNs are typically trained using the backpropagation algorithm, which calculates the gradient of the loss function with respect to each weight and bias by applying the chain rule of calculus [33]. The gradients are then used to update the parameters using an optimization algorithm such as stochastic gradient descent (SGD) or one of its variants [34].

The depth of a neural network, or the number of hidden layers, is a crucial factor in its ability to learn complex and hierarchical representations. Deep networks have been shown to outperform shallow networks in various tasks, as they can learn to extract and compose high-level features from raw input data [35].

2.4.3.3 Neurons

A neuron is the most important building block of a neural network. It is a computational element that connects to other neurons forming layers and is inspired by the biological neurons of the brain and works very similarly. It receives input from other neurons or external sources (if the neuron is at the input layer), processes the information, and passes the output to other neurons or generates a final output for the network (if the neuron is at the output layer)[36].

Each neuron in an artificial neural network performs the following operations:

Input aggregation: The neuron receives input from multiple sources, which can be neurons in the previous layer or external input data. Each input is multiplied by a corresponding weight, and the weighted inputs are then summed together with an additional bias term. The weighted sum can be expressed mathematically as:

$z = \sum(wi * xi) + b$

where z is the weighted sum, wi is the weight for the i-th input, xi is the i-th input, and b is the bias term.

Activation: After computing the weighted sum, the neuron applies a non-linear activation function to the result. This function transforms the input, introducing non-linearity into the network, and allowing it to learn and represent complex patterns and relationships in the data. Common activation functions include the sigmoid, hyperbolic tangent (tanh), and rectified linear unit (ReLU) [37].

$a = f(z)$

where a is the activation, f is the activation function, and z is the weighted sum calculated earlier.

Output: The activation value 'a' is the output of the neuron, which is passed to other neurons in the subsequent layer or used as the final output of the network. In the case of multi-class classification problems, the output layer often employs a softmax activation function, which converts the activations of the neurons into probabilities that

sum to 1. This helps in determining the most likely class or decision for a given input [38].

Neurons in a neural network are typically organized into layers, including the input layer, one or more hidden layers, and the output layer. The input layer is responsible for receiving the raw data which resemble the problem state, while the output layer generates the final predictions or decisions which resemble the action space. The hidden layers, which contain the majority of the network's neurons, perform a series of non-linear(most of the time) transformations to learn and extract features from the input data [32].

The learning process in a neural network involves adjusting the weights and biases of the neurons to minimize the difference between the network's predictions and the actual target values. This is typically achieved through a process called backpropagation, which will be analyzed later, and computes the gradients of the loss function with respect to the weights and biases and updates them using an optimization algorithm [39].

2.4.3.4 Weights

Weights in a neural network are numerical values associated with the connections between neurons, which determine the strength or influence of the connections. They play a critical role in the learning process, as they are adjusted during training to minimize the difference between the network's predictions and the actual target values. Weights enable the network to adapt and capture the underlying patterns and relationships in the input data [32].

In an artificial neural network, each neuron receives input from other neurons or external sources. The input is multiplied by its corresponding weight before being summed together with other weighted inputs and a bias term.

Weights serve several purposes in a neural network:

1. Importance of connections: The weights determine the significance of a connection between neurons. A larger weight implies that the input has a greater impact on the output, while a smaller weight indicates a weaker influence [38].
2. Learning process: During training, the weights are updated to minimize the error between the network's predictions and the actual target values. The learning algorithm, such as gradient descent, adjusts the weights based on the gradients of the loss function with respect to the weights [39].
3. Feature extraction: The weights, especially in DNNs, help the network learn hierarchical representations of the input data. Lower-level features are combined and transformed by the weights in subsequent layers to form higher-level, more abstract features [30].
4. Function approximation: The weights, in combination with activation functions, enable the neural network to approximate complex, non-linear functions and relationships between inputs and outputs [40].

2.4.3.5 Activation Function

An activation function in a neural network is a non-linear function applied to the weighted sum of inputs at each neuron. The activation function introduces non-linearity into the network, allowing it to learn and represent complex patterns and relationships in

the input data. It is a crucial component of neural networks, as without non-linearity, the network would be limited to modeling only linear relationships, significantly reducing its expressive power [32].

Mathematically, the activation function as mentioned before takes the form:

a = f(z)

where a is the activation, f is the activation function, and z is the weighted sum calculated as the sum of the product of the input values (xi) and their corresponding weights (wi) along with the bias term (b).

There are several popular activation functions used in neural networks:

1. Sigmoid function: The sigmoid function, also known as the logistic function, maps the input values to a range between 0 and 1. It is defined as:

   f(z) = 1 / (1 + exp(-z))

   The sigmoid function is smooth and has a simple derivative, which is useful for gradient-based optimization algorithms. However, it can suffer from the vanishing gradient problem, where the gradients become very small during backpropagation, causing slow learning [37].

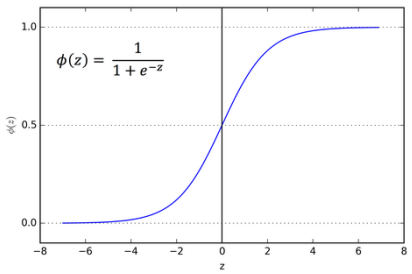

Image 2: Sigmoid activation function

2. Hyperbolic tangent (tanh) function: The tanh function is similar to the sigmoid function, but it maps the input values to a range between -1 and 1. It is defined as:

   f(z) = (exp(z) - exp(-z)) / (exp(z) + exp(-z))

   The tanh function shares many properties with the sigmoid function, but its output is zero-centered, making it more suitable for certain applications.

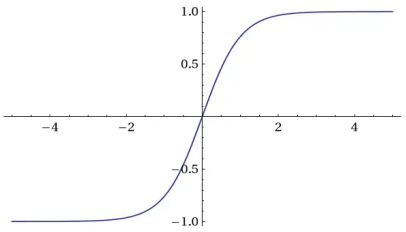

Image 3: Tanh activation function

3. Rectified Linear Unit (ReLU) function: The ReLU function is a piecewise linear function that maps the input values to a range between 0 and positive infinity. It is defined as f(z) = max(0, z)

The ReLU function is computationally efficient and helps alleviate the vanishing gradient problem. However, it can suffer from the dying ReLU problem, where some neurons become inactive during training, producing zero gradients [41].

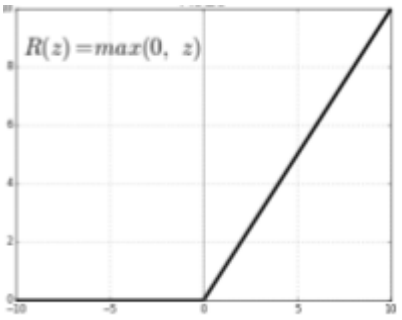

Image 4: ReLU activation function

Activation functions play a vital role in determining the output of neurons and the overall behavior of the network. They help the network learn complex, non-linear relationships and improve the network's capacity to generalize and make predictions based on the input data.

2.4.3.6 Error function

Error function, also known as the loss function or cost function, is a measure of the discrepancy between the network's predictions and the actual target values. The objective of the training process is to minimize the error function. Different error functions are used for different tasks:

1. Mean Squared Error (MSE): Commonly used for regression tasks, the MSE calculates the average squared difference between the predicted and actual values:

   MSE = (1/n) * Σ(yi - ŷi)^2

   where yi is the actual target value, ŷi is the predicted value, and n is the number of samples.

2. Cross-Entropy Loss: Widely used for classification tasks, the cross-entropy loss measures the difference between the predicted probability distribution and the actual probability distribution:

   Cross-Entropy = -Σ(yi * log(ŷi))

   where yi is the actual probability distribution (usually one-hot encoded) and ŷi is the predicted probability distribution.

The choice of error function depends on the problem being solved, as different functions have different properties and sensitivities to prediction errors. The error function is one of the key components for the back-propagation step.

2.4.3.7 Back-propagation

Back-propagation is an algorithm used in the training of artificial neural networks and is in simple terms the learning procedure. It is based on the chain rule of calculus and is used to compute the gradients of the loss function based on the weights of the network. These gradients are then used by an optimization algorithm, such as gradient descent, to update the weights and biases, minimizing the error between the network's predictions and the actual target values [33].

The backpropagation algorithm involves the following steps:

1. Forward pass: The input is passed through the network to compute the output (predictions) for each neuron. This involves calculating the weighted sum of inputs and applying the activation function for each neuron in each layer.
2. Compute the error: The error or loss function quantifies the difference between the network's predictions and the actual target values. Common loss functions include mean squared error for regression tasks and cross-entropy loss for classification tasks [38].
3. Backward pass: First the calculation of the gradient for each neuron in the output layer is done and then propagates the gradients backward through the network, layer by layer.
4. Update the weights and biases: The optimization algorithm, such as gradient descent updates the weights and biases using the computed gradients. This process is repeated for multiple iterations or episodes until the network converges to a satisfactory level of performance or the error reaches a minimum threshold [32].

Backpropagation is an extremely efficient method for training neural networks, as it allows for the calculation of the gradients in linear complexity. The method has been widely adopted in most deep-learning fields.

2.4.3.8 DQN(deep Q-learning network)

DQN is an extension of the Q-learning algorithm that combines the power of DNNs with the principles of RL. DQN was introduced by Volodymyr Mnih and his colleagues at DeepMind in 2013 and demonstrated a significant breakthrough in RL by achieving human-level performance on a wide range of Atari games [28].

In RL, an agent interacts with an environment to learn the optimal policy that maps states to actions, maximizing the expected cumulative reward over time. Q-learning is a popular RL algorithm that learns an action-value function, Q(s,a), representing the expected value of taking action in state s and following the optimal policy thereafter.

The DQN algorithm extends the traditional Q-learning approach by approximating the action-value function Q(s,a) using a DNN instead of a Q-table. This neural network, called the Q-network, takes the state as input and outputs the Q-values for all possible actions.

The DQN algorithm involves the following key components and modifications to the standard Q-learning approach:

1. Experience Replay: DQN stores the agent's experiences (state, action, reward, next state) in a replay buffer. During training, random mini-batches of experiences are sampled from the buffer to update the Q-network. This approach breaks the correlation between consecutive experiences and improves the

stability and convergence of the learning process [42]. Since experience replay was a very important aspect of our training we will delve into it later.

2. Target Network: To stabilize the learning process, DQN uses a separate target network, which is a copy of the Q-network but with fixed parameters. The target network is used to compute the target Q-values for the loss function. The parameters of the target network are updated periodically by copying the parameters from the Q-network [32].
3. Loss Function and Optimization: DQN minimizes the mean squared error between the predicted Q-values and the target Q-values, computed using the target network and the immediate reward:

   Loss = (Q(s, a) - (r + γ * max_a' Q_target(s', a')))^2

   where Q(s, a) is the predicted Q-value, r is the immediate reward, γ is the discount factor, and Q_target(s', a') is the target Q-value from the target network. The optimization is performed using gradient descent or its variants (e.g., RMSProp, Adam).
4. Exploration-Exploitation Trade-off: DQN uses an ε-greedy policy to balance exploration and exploitation. With probability ε, the agent takes a random action to explore the environment, while with probability 1-ε, the agent selects the action with the highest Q-value (exploitation). The value of ε is gradually decayed over time to encourage more exploration initially and more exploitation as the agent learns the optimal policy.

The DQN algorithm has been a significant milestone in the development of RL algorithms and has inspired numerous extensions and improvements, such as Double DQN, Dueling DQN, and Prioritized Experience Replay.

2.4.3.9 Experience Replay

Experience Replay is an essential component of the DQN algorithm, which addresses critical challenges in training DRL agents. In RL, an agent interacts with an environment and learns from its experiences to improve its decision-making policy over time. However, when using DNNs to approximate the Q-values, several issues arise that can hinder learning:

1. Temporal correlations: In traditional RL, the agent's experiences are obtained sequentially, and the training data are highly correlated. This correlation violates the assumption of independent and identically distributed (i.i.d.) data, which many optimization algorithms, such as stochastic gradient descent, rely upon. The correlated data can lead to inefficient learning and instability in the training process.
2. Overfitting: Due to the limited number of experiences available, the agent may overfit to a small set of recent experiences, leading to a biased policy that does not generalize well to unseen situations.
3. Catastrophic forgetting: As the agent updates its Q-network, it may overwrite the knowledge gained from previous experiences, leading to a phenomenon known as catastrophic forgetting. This forgetting can result in the agent repeatedly learning and forgetting the same experiences, which hampers the learning process.

Experience Replay addresses these challenges by maintaining a replay buffer, which stores the agent's experiences as tuples (state, action, reward, next state). During

training, the agent samples random mini-batches of experiences from the buffer to update its Q-network. This approach has several benefits:

1. Breaking temporal correlations: By sampling random experiences from the buffer, Experience Replay breaks the correlations between consecutive experiences. This approach approximates the i.i.d. assumption, enabling more efficient and stable learning.
2. Reducing overfitting: Random sampling from a diverse set of experiences helps to prevent overfitting by providing a broader view of the environment and encouraging the agent to learn a more general policy.
3. Mitigating catastrophic forgetting: The replay buffer retains past experiences, allowing the agent to revisit and reinforce the knowledge gained from previous interactions with the environment. This approach mitigates the issue of catastrophic forgetting and helps the agent to learn more effectively.

In summary, Experience Replay plays a crucial role in improving the stability, efficiency, and robustness of the learning process in DRL algorithms, such as DQN[32][42].

2.5 Related work

This chapter will include a comprehensive overview of the related work in the field of traffic signal control, specifically focusing on adaptive and learning-based approaches. The objective of this review is to provide the reader with a clear understanding of the state-of-the-art techniques, the underlying principles, and the challenges faced by researchers and practitioners in this domain. We begin by discussing traditional methods, such as fixed-time and actuated traffic signal control, which have been widely implemented in practice. Subsequently, we delve into the more recent advances in adaptive traffic signal control (ATSC) algorithms, highlighting their contributions and limitations. The approach will be on a per paper basis in order to be more clear what the results and the contribution of each paper for the field.

2.5.1 Deep Reinforcement Q-Learning for Intelligent Traffic Signal Control with Partial Detection [43]

In this paper, the authors present a novel deep Q-learning model for traffic signal control at single intersections with partial detection over connected vehicles. They introduce a new state representation for partially observable environments called partial Discrete Traffic State Estimator (DTSE) and a new reward function for traffic signal control, total squared delay. The proposed model is evaluated against two existing actuated traffic signal control algorithms, Max Pressure and Self-Organizing Traffic Lights (SOTL), in a two-step comparative analysis by episode mean total delay. The model shows improved performance over Max Pressure and SOTL in full detection, particularly in more complex intersection configurations with varying traffic demands and for 4-phase programs. The model also performs efficiently for connected vehicle (CV) penetration rates as low as 20% in partially observable environments. In conclusion, the proposed deep Q-learning model offers a more efficient and adaptive approach to traffic signal control compared to existing actuated methods, with acceptability and optimality thresholds estimated at 20% and 40% CV penetration rates, respectively. The authors suggest several avenues for future research, including exploring probabilistic estimation methods, generalizing partial DTSE across multiple intersection sizes, developing reward functions based only on CVs, improving performance at low CV penetration rates, extending the model to decentralized multi-agent RL for coordinated traffic light grids, and validating the model on real-world simulated networks.

2.5.2 An Open-Source Framework for Adaptive Traffic Signal Control [44]

This study optimized hyperparameters and compared the performance of different controllers, such as Max-pressure, DQN, DDPG, Uniform, Webster's, and SOTL. The results showed that hyperparameters significantly impact the performance of the controllers, especially learning-based controllers. The Max-pressure algorithm achieved the best performance, while the SOTL controller exhibited the worst results. The DQN and DDPG controllers demonstrated interesting performance differences, with the DQN performing well during the demand peak but poorly during low-traffic demand periods. In contrast, the DDPG controller achieved low queues and delay at the beginning and end of the simulation but was surpassed by the DQN controller during the demand peak. The study highlights the importance of understanding the sensitivity of adaptive traffic signal controllers to hyperparameters and suggests future research to investigate the performance difference between RL traffic signal controllers and explore richer function approximators and RL algorithms to improve their performance.

2.5.3 Using a Deep Reinforcement Learning Agent for Traffic Signal Control [45]

The main goal of this paper is implementing the Deep Q-Learning Traffic Signal Control Agent (DQTSCA). The experiments were conducted with each training epoch representing 1.25 hours of simulated traffic. The intersection geometry and traffic movements were designed to mimic real-world conditions, and the agent was trained using a biologically-inspired process known as experience replay. The study also developed a shallow neural network Traffic Signal Control Agent (STSCA) for comparison purposes. The results showed that the DQTSCA outperformed the STSCA in three out of four traffic metrics, reducing average cumulative delay by 82%, average queue length by 66%, and average travel time by 20%. The improved performance is attributed to the use of the DTSE and the deep architecture employed in the DQTSCA.

2.5.4 Multi-intersection Traffic Optimisation: A Benchmark Dataset and a Strong Baseline[46]

The authors explore a variety of traffic signal control methods, with a focus on their proposed Edge-Weighted Graph Convolutional RL (EGU-RL) model. EGU-RL leverages the advantages of graph convolutional networks (GCNs) to capture spatial relationships between traffic junctions and RL to learn optimal traffic signal control policies. The model is designed to jointly control multiple traffic junctions while maintaining computational efficiency, achieving better performance and generalization compared to other methods like fixed-rule controllers, random controllers, and other multi-agent RL models. Competing methods include fixed-rules controller, random controller, auction-based model, single-agent for single-junction multi-agent RL model (MARL-s), and grouped multi-agent RL model (MARL-g). The experiments are conducted on various maps of Suzhou, Manhattan, and synthetic scenes. The EGU-RL model demonstrates promising results, outperforming other methods in most cases. It can reduce total vehicle waiting time by over 80% compared to fixed-schedule controllers and surpasses other methods by a large margin, especially in complex environments. The EGU-RL model has fewer parameters and demonstrates better generalization ability to unseen environments. In ablation studies, removing either the edge-weighted graph convolutional encoder or the unified structure decoder significantly reduces the model's performance, indicating the importance of these components.

2.5.5 Reinforcement Learning Benchmarks for Traffic Signal Control[47]

In this paper the experimental section is divided into three parts. First, the authors validate their benchmark algorithms' implementation by comparing their results to previous publications. Second, they present a comparative study between state-of-the-art approaches in realistic traffic scenarios. Finally, they draw general conclusions that characterize the algorithms and their performance. In the validation phase, the researchers closely replicate the scenarios presented in the original publications of the algorithms, such as MPLight and (Federated Multi-Agent Actor-Critic)FMA2C, and compare their performance to previously reported results. MPLight is a model-free, DRL algorithm designed for traffic signal control in urban environments. It employs a single-agent approach and utilizes a DQN to learn the optimal traffic signal timings based on the current traffic state. MPLight focuses on minimizing the traffic delay by making decisions on traffic signal timings at each intersection. FMA2C is another RL-based traffic signal control algorithm that aims to optimize traffic signal timings in urban networks to improve traffic flow and reduce congestion. FMA2C uses a coordinated multi-agent approach, where multiple traffic signal controllers in the network communicate and cooperate to achieve global optimization. The algorithm employs a federated learning framework, allowing agents (traffic signals) to learn local policies based on their local observations and experiences. These local policies are then periodically shared and aggregated to update the global policy, which in turn is distributed back to the agents. This federated learning approach enables the agents to learn and adapt to different traffic scenarios while maintaining coordination and cooperation among them. They find that the implemented algorithms in their study exhibit performance trends consistent with the original publications. In the comparative study, the authors evaluate the benchmark RL controllers on a range of realistic traffic scenarios without tuning their hyperparameters for each scenario. The results suggest that decentralized control algorithms are more robust than coordinated control algorithms when considering realistic traffic scenarios. The authors speculate that this is due, in part, to the efficient data aggregation of the independent learners utilizing convolutional layers. Future work should examine merging this efficient data representation with state-of-the-art coordinated control approaches. It is highlighted the importance of the time elapsed before an algorithm reaches its best performance, as this is crucial for real-world implementation. In this regard, the MPLight algorithm presents a trade-off of reliability and converged performance for learning speed, while FMA2C requires a large amount of time to achieve its best performance on challenging synthetic coordination problems.

3. IMPLEMENTATION AND EVALUATION OF TRAFFIC SIGNAL CONTROL METHODS

The previous chapters have provided an overview of the history of traffic signal control, classical methods, and modern techniques like MAX-Pressure and techniques incorporating DRL, specifically the DQN algorithm. In this chapter, we present the implementation and evaluation of the traffic signal control methods discussed earlier in the thesis. We will describe the simulation environment used to model realistic traffic conditions, along with the parameters and configurations of each control method. Furthermore, we will analyze what performance metrics we used to evaluate the effectiveness of each method in managing traffic flow, reducing congestion, and minimizing delays.

3.1 Implementation details

3.1.1 Webster

The standard Webster algorithm calculates the optimal cycle duration based on static saturation flows and divides the total available green time of an intersection into its phases proportional to these predefined saturation flows of each phase. However, this approach is not effective in various real-world traffic scenarios where traffic conditions are dynamic and constantly changing.

In order to address this issue, we have implemented a dynamic Webster algorithm that adapts better to real traffic conditions by continuously monitoring traffic data in order to change the cycle durations. First, the vehicle counts and saturation flows are initialized with default values per intersection. Then, at regular intervals of a few hundred steps, we keep a record of the vehicles that have passed through an intersection at that time, in order to calculate the new saturation flows and cycle durations of each intersection. Finally, the total available green time is distributed to each phase proportionally to their saturation flows.

These steps are repeated during the simulation to make sure that the cycle duration is adapting to current traffic conditions and it allows for more optimal traffic management. By taking this approach, the Webster algorithm is able to handle multiple traffic scenarios more efficiently.

3.1.2 Greenwave - MAXBAND

The Greenwave and Maxband algorithms are specifically designed for managing traffic flow along arterial roads, where the intersections are closely spaced, and the primary goal is to minimize the number of stops and delays experienced by vehicles traveling in a coordinated manner. Due to the specialized nature of these algorithms, their implementation differs from the more general Webster algorithm. In both methods, the offset between intersections, determined by considering the distance and average speed between intersections, is adapted while keeping the cycle length fixed.

Specifically MAXBAND algorithm also takes into account the volume of incoming traffic added from previous intersections and from intersecting roads to optimize its offset and traffic cycle phases in order to maximize the bandwidth of green time for the critical direction of flow. For instance, for a downstream intersection along the arterial road, the volume of vehicles will likely be higher since cars join from the connecting side streets. Therefore, MAXBAND carefully factors these traffic influxes into its calculation of optimal offsets and the distribution of green time across phases.

Finally the scenarios for these two algorithms have to be very specific and generally are difficult to be compared with the rest of the algorithms since in different scenarios their results are far worse since they are designed to optimise the flow in arterial roads not small non-linear intersections.

3.1.3 MAX-pressure

The Max Pressure is designed to maximize the throughput at an intersection by selecting the phase that alleviates the most pressure between incoming and outgoing lanes. In this implementation, the calculation and decision of the next phase are performed after a minimum green time of 10 simulation steps, ensuring that the traffic signal control system continuously adapts to the changing traffic conditions. By having a

minimum green time ensures that we do not have big red - yellow to green time ratio which results in worse traffic conditions overall.

Specifically every 10 simulation steps the algorithm calculates the phase pressures for each phase per traffic light and makes a decision to change the active phase based on the maximum phase pressure. If the current phase is about to change then a yellow phase is added in between the old and the new phase to ensure a smooth transition.

Finally, MAX-pressure is a very versatile algorithm that can be adapted to any scenario and is not limited to specific scenarios, thus we were able to compare it with all the other algorithms.

3.1.4 DQN

3.1.4.1 Architecture

The architecture of the DQN we employ in our traffic signal control system follows a sequential structure. This means that the layers are organized in a straightforward sequence where the output from one layer serves as the input to the next. The network is constructed using six layers in total, with each layer serving a distinct purpose.

- Input Layer: The first layer, which acts as the input layer to the network, consists of a volatile number of neurons which is directly connected to the dimensions of our state representation for the traffic situation.
- Hidden Layers: Following the input layer, our network includes four fully connected hidden layers, each containing 100 neurons. These hidden layers also use the ReLU activation function. The role of these layers is to discover and represent the intricate correlations between our traffic states and the respective actions.
- Output Layer: The final layer in our network serves as the output layer. This is a dense layer with the number of neurons equivalent to the number of possible actions in our traffic signal control system, which are defined by the number of different phases. This layer uses a linear activation function, meaning the outputs of the neurons are used directly without any further transformation. Each output corresponds to the predicted Q-value for a specific action, given the input state.
- Compilation: Once the structure of our model is defined, it needs to be compiled for training. We utilize MSE as our loss function, a standard approach for regression tasks, and particularly suitable for Q-learning where we aim to minimize the disparity between the predicted and target Q-values. For optimization, we use the Adam optimizer with a specified learning rate. Adam, an adaptive learning rate optimizer, is highly effective in training deep learning models. The model also tracks accuracy as a metric, which can provide useful insight into the learning process of the network.

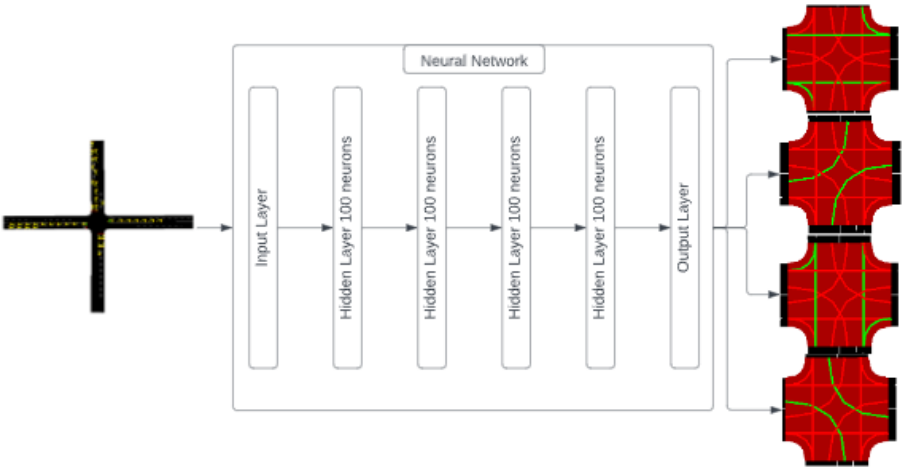

Image 5: Our neural network architecture

3.1.4.2 State representation

In our DQN implementation for traffic signal control, the state is represented by a two-dimensional grid for each incoming road, with each road possibly having multiple lanes. The intersection in focus is dissected into various incoming lanes, and each lane is further segmented into equal parts. Each segment or grid cell corresponds to a particular portion of the lane, and the value in each cell is a count of the current number of vehicles occupying that specific segment. This approach effectively transforms the real-world traffic scenario into a structured numerical representation. The grid-based state encapsulation provides a scalable and detailed view of the traffic situation at the intersection. This comprehensive representation captures the spatial distribution and congestion levels on the incoming lanes, providing an efficient input state to the DQN for decision-making on traffic signal control.

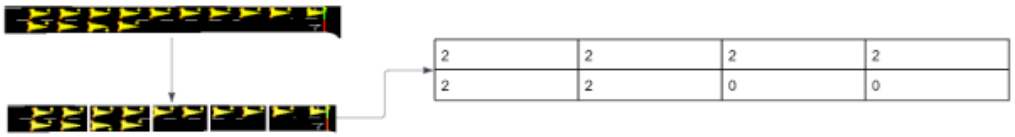

Image 6: How the state of a road representation is quantified

3.1.4.3 Action representation

Our action space is represented by the different green and partial green phases of the traffic signal at the intersection. Each phase is a unique configuration of the traffic signal control system. For instance, for a specific incoming road with two potential directions, the traffic control system can assume a variety of phases. These phases can range from all green lights (indicating clear passage), all red lights (indicating a halt), all yellow lights (signifying an impending change in signal), to partial green phases (where, for instance, a road conflicting with the opposing one might still have the green light). Only green and partial green phases constitute distinct actions that the DQN agent can take. For every given state of the traffic situation, the agent chooses an action - i.e., a phase

of traffic lights - that it expects to be most effective according to its current policy. This approach of defining actions provides a direct mapping from the agent's decisions to the operations of the real-world traffic signal system, enabling a practical and intuitive framework for traffic signal control.

3.1.4.4 Reward

The reward function plays a critical role in guiding the agent towards optimal actions. The reward is intricately tied to the queue length in each lane at the agent's intersection, with the goal of the DQN agent being to minimize these queue lengths, thus promoting smooth and efficient traffic flow. Specifically, we quantify the total queue length for each state-action pair. The reward assigned to the agent is the negative of this total queue length. The negative sign is crucial because it instructs the DQN to maximize its cumulative reward, which effectively means minimizing the total queue length at the intersection. This reward value is subsequently utilized by the Adam optimizer - an algorithm for first-order gradient-based optimization of stochastic objective functions. The optimizer uses this signal to adjust the weights of the neural network. By using the negative of the total queue length as a performance indicator, the neural network is effectively guided to learn the most beneficial traffic signal phases for reducing congestion at the intersection. As a result, the RL agent consistently refines its policy, becoming more proficient at controlling the traffic signals as it accumulates more experience.

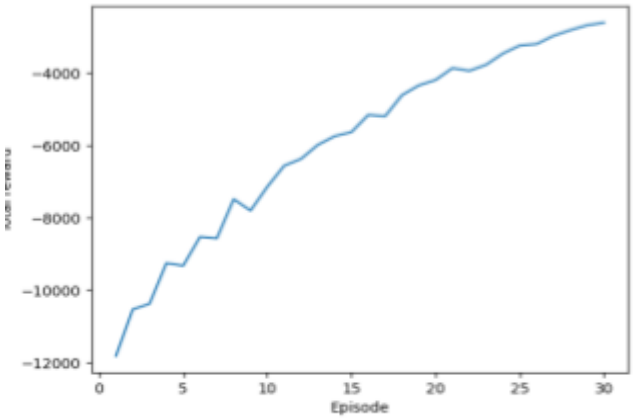

Image 7: Reward function results during the 30 episodes of training

3.1.4.5 Training

Each simulation run, or "episode", lasts for 5000 steps inside of SUMO, and the entire training process comprises 30 such episodes. In our model, every traffic signal is managed by a separate DQN agent. These agents interact with the traffic environment and learn from it concurrently, allowing for a distributed and efficient approach to managing complex traffic scenarios. The training procedure leverages the concept of experience replay, a technique widely used in DQN models.

An "experience" is a fundamental unit of information that the DQN agent uses to learn and update its policy. Each experience is a tuple that consists of four elements: a state, an action, a reward, and the subsequent state.

- The "state" is the current representation of the traffic situation at an intersection, captured as a 2-dimensional grid for each incoming road, as discussed earlier.

- The "action" is the decision made by the DQN agent in response to the current state, represented by the selected traffic signal phase.
- The "reward" is the feedback received after executing the action, computed as the negative of the total queue length at the intersection.
- The "next state" is the traffic situation resulting from the execution of the action, again represented in the form of a 2-dimensional grid.

Over the course of each episode, the agents' experiences are stored in a memory pool. From this experience pool, 100 experiences are sampled randomly to form a mini-batch. The DQN agent then makes predictions for the Q-values based on the current state and updates its estimates using the observed reward and the maximum predicted Q-value for the next state, effectively following the Q-learning update rule. This stochastic batch update process is repeated 20 times, allowing the network to iteratively refine its estimates of the Q-values. This method of training allows for a more stable and robust learning process, by breaking the correlation between experiences and enabling the agent to learn from a diverse set of past actions and outcomes. As a result, the trained DQN agent is equipped to effectively control the traffic signals, adapting to the dynamics of the traffic flow and making decisions that minimize queue lengths at the intersections.

3.2 Evaluation Techniques

Now we will discuss about the evaluation techniques and methods used to compare the performance of the five traffic control algorithms, Webster's method, Greenwave, Maxband, Max Pressure, and DQN. To obtain a deep understanding of the strengths and weaknesses of each algorithm, a variety of scenarios will be created to simulate different traffic conditions and environments. Furthermore, specific scenarios will be designed to highlight the unique characteristics and advantages of each algorithm. The comparison will be based on performance metrics to ensure a fair assessment of each method.

3.2.1 Evaluation Technique

The evaluation of the five traffic control algorithms will be conducted through SUMO which allows the creation of realistic traffic scenarios with customizable parameters, such as network configurations, different traffic patterns, and signal timings. The software will also facilitate data collection on various performance metrics, providing a quantitative basis for the comparison of the algorithms.

3.2.2 Performance Metrics

To ensure a fair comparison, the algorithms will be evaluated based on performance metrics. These metrics will provide a quantitative assessment of each algorithm's effectiveness in managing traffic flow and minimizing delays. The key performance metric is the total delay experienced by vehicles at signalized intersections. Another metric used for the arterial road scenario was the total simulation time. In order to simulate an arterial road just like a real-world one, you want maximum congestion. The total delay performance metric does not work because there is no infinite space inside of the simulation environment to have all the vehicles at once. So the best use is to check when the simulation will finish(the last time a vehicle was still inside of the simulation)

3.2.3 Scenarios Analysis and Comparison

To carry out a comparative evaluation and comparison of the traffic control algorithms, an assortment of scenarios has been developed. Each of these scenarios is designed to simulate diverse traffic conditions and environments, incorporating various levels of complexity, road infrastructure, and traffic congestion. The performance metrics collected from the simulations will be analyzed to determine the effectiveness of each algorithm in various scenarios. The comparison will involve examining the performance of each algorithm across all scenarios, and identifying the strengths and weaknesses of each method. The analysis will also highlight the unique advantages of specific algorithms in specialized scenarios, showcasing their potential applications in real-world traffic management.
Scenario 1: This represents the simplest case, involving a single intersection where two roads intersect at a 90-degree angle, each road having two incoming lanes for each side.

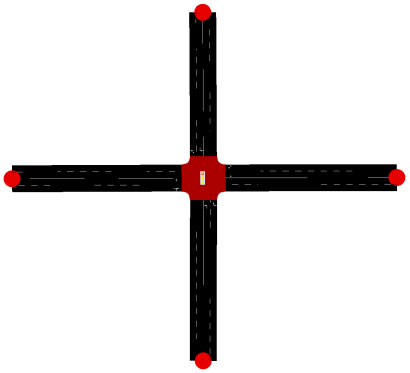

Image 8: Single intersection with two lines each

Scenario 2: Building on the first scenario, this features a double intersection, mirroring the structure of the single intersection but with one more intersection next to the first one.

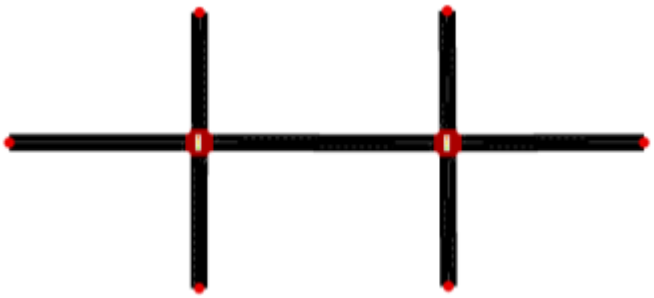

Image 9: Double connected intersection

Scenario 3: Building on the first scenario, this scenario features an intersection with unbalanced traffic. Involving a single intersection where two roads intersect at a 90-degree angle, one road having three lanes and the other one with two lanes.

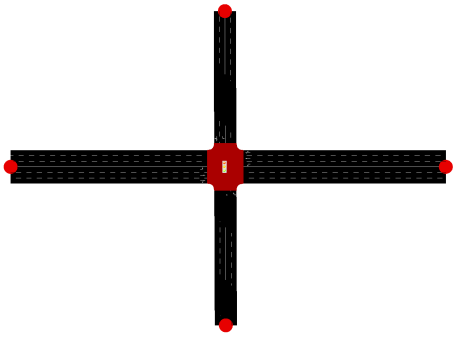

Image 10: Unbalanced single intersection with three lanes on the main road and two lanes on the vertical

Scenario 4: Introducing a higher degree of complexity, this scenario mimics a block of congested roads in city centers, simulated through a 3x3 grid of intersections.

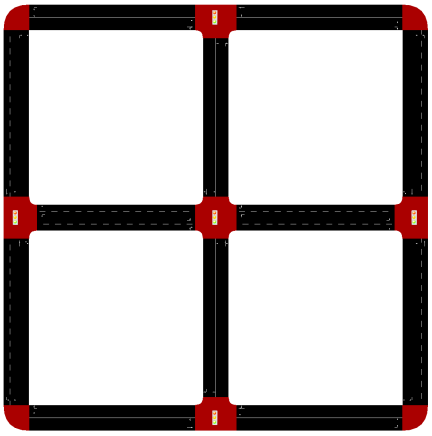

Image 11: 3x3 grid with two lanes on each road

Scenario 5: To further escalate the complexity and closely simulate real-world traffic conditions, this scenario utilizes a 4x4 grid of intersections, reflecting highly congested city blocks.

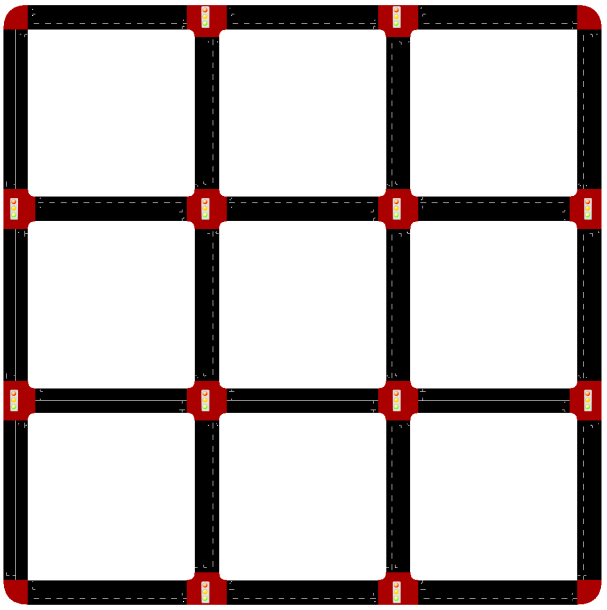

Image 12: 4x4 grid with two lanes on each road

Scenario 6: This scenario is designed to replicate an arterial road, which features a three-lane road intersecting with two different two-way roads at two distinct points.

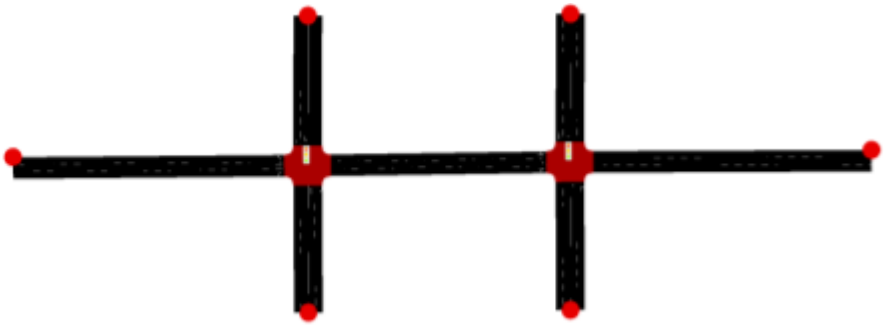

Image 13: Arterial road with one way three lane road and two verical two lane road

Scenario 7: Finally, the most complex scenario draws from reality, replicating a large-scale urban traffic scenario from the center of Manhattan with 18 intersections of varying sizes.

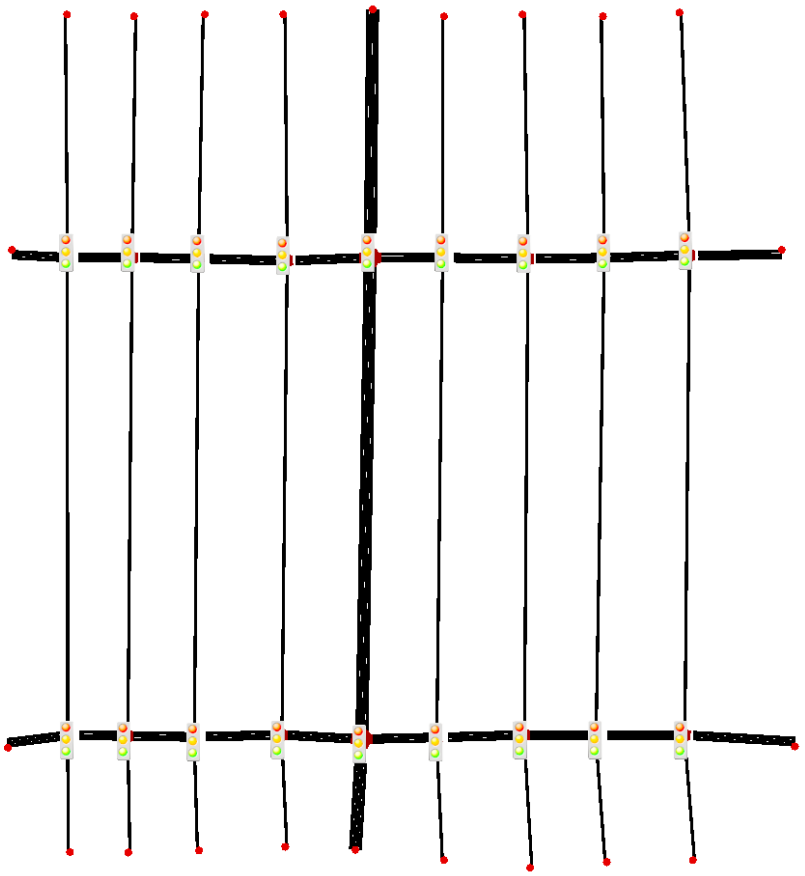

Image 14: Real case scenario, depicting neighborhood in Manhattan

4. RESULTS AND DISCUSSION

Following the implementation details, we will analyze the results of the comparative analysis, highlighting the strengths and weaknesses of each method under different traffic conditions and scenarios. This analysis will provide insights into the suitability of each approach for various traffic management applications and help identify potential areas for improvement and future research. Finally, we will discuss the implications of our findings, drawing conclusions about the overall performance of classical and modern traffic signal control methods, and providing recommendations for traffic management practitioners and policymakers. This chapter aims to bridge the gap between theory and practice, ultimately contributing to the development of more efficient and adaptive traffic signal control systems for our increasingly urbanized world.

Scenario 1: The simplest scenario witnessed the lowest waiting time with the DQN algorithm, reducing waiting time by 7.2% compared to Max Pressure, and 27.3% compared to the classical Webster method. This demonstrates the efficacy of modern AI techniques, even in less complicated traffic situations.

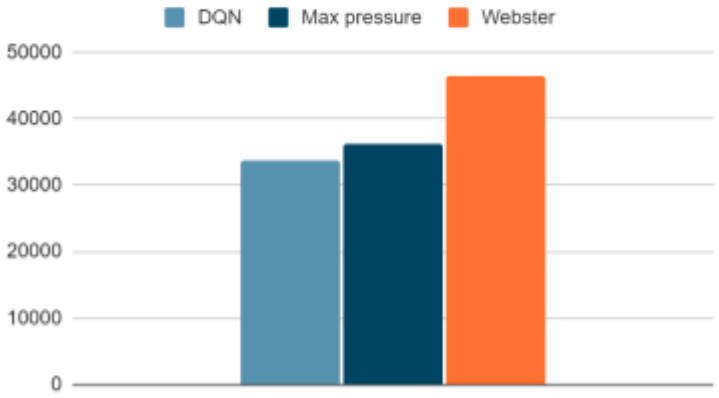

Image 15: Scenario 1 - Single intersection results

Scenario 2: Similar to the first scenario, DQN performed the best. It reduced waiting time by 51.7% compared to Max Pressure, and 28.1% compared to Webster. It further underscores the potential of DQN for larger, yet not overly complex, traffic systems.

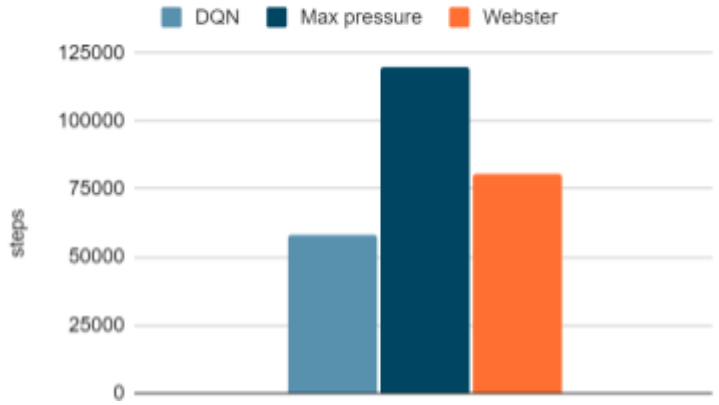

Image 16: Scenario 2 - Double connected intersection results

Scenario 3: In this scenario, DQN outperformed both other algorithms, with waiting times less by 11.5% and 23.7% than Max Pressure and Webster, respectively. It indicates that DQN can handle unbalanced traffic more efficiently.

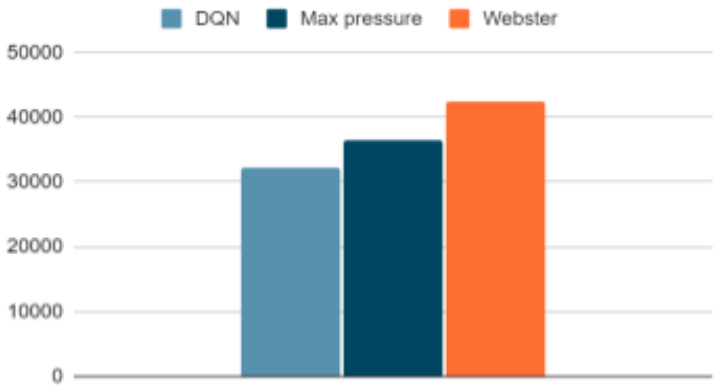

Image 17: Scenario 3 - Unbalanced intersection results

Scenario 4: DQN's performance remains superior even in more complex scenarios like the 3x3 grid. It reduced waiting time by 19.5% compared to Max Pressure and by 40.5% compared to Webster. The DQN's ability to learn and adapt from larger data sets makes it suitable for managing dense urban traffic.

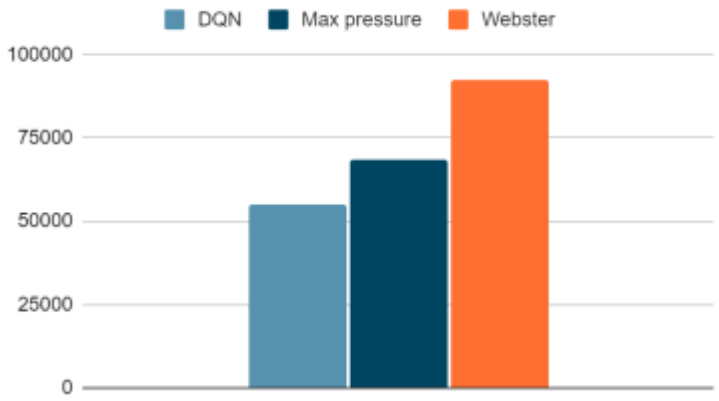

Image 18: Scenario 4 - 3x3 grid results

Scenario 5: For the highly congested 4x4 grid, DQN again delivered the lowest waiting times. It performed 33.6% better than Max Pressure and 49.3% better than Webster. This result strengthens the argument for DQN's robustness in dealing with high complexity and traffic volume.

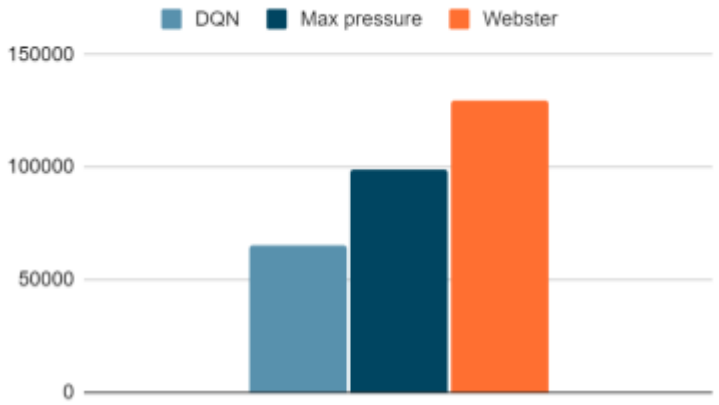

Image 19: Scenario 5 - 4x4 grid results

Scenario 6: DQN performed the best, followed closely by Max Pressure and Webster, which showed identical results. Interestingly, both Greenwave and Maxband, dedicated arterial road algorithms, outperformed by DQN. It demonstrates that DQN's adaptive learning capability is beneficial, even in specific traffic layouts like arterial roads.

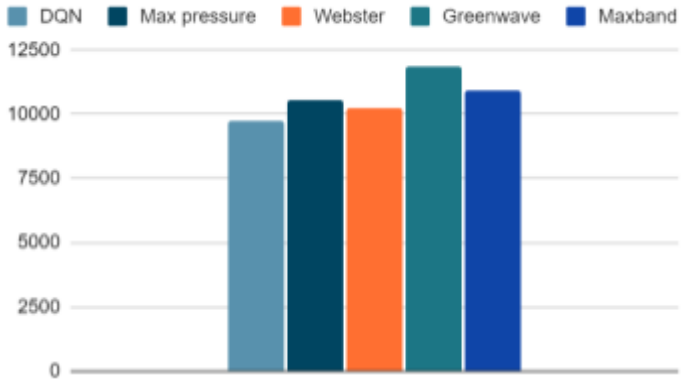

Image 20: Scenario 6 - Arterial road results

Scenario 7: In our most complex scenario, DQN again provided the lowest waiting time, outperforming Max Pressure by 48.3% and Webster by 55.9%. Despite the complexity and diversity of the Manhattan scenario, DQN demonstrated consistent superiority, indicating its utility for large-scale, real-world urban traffic management.

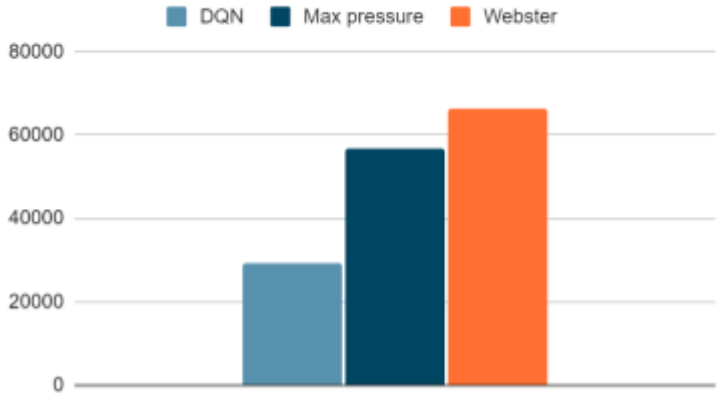

Image 21: Scenario 7 - Real case results

In conclusion, the DQN algorithm demonstrated superior performance across all scenarios. It outperformed both the dynamic Webster algorithm and the more modern Max Pressure algorithm. DQN's adaptability to traffic flow and its learning capability, which allows it to optimize signals based on past experiences, contribute to its effectiveness. This implies a potential shift in traffic signal control towards machine learning-based methods like DQN. While classical methods have their advantages and are useful under certain circumstances, modern AI-driven techniques could lead to more efficient traffic management and a reduction in waiting times. However, further research is needed to investigate DQN's performance under other traffic scenarios, and how to most efficiently integrate such systems in real-world traffic management.

5. CONCLUSIONS

This thesis presents a comparative study of five distinct traffic control algorithms, ranging from classical methods such as Webster's method, Greenwave, and Maxband, to dynamic approaches, namely the Max Pressure method and a Deep Q-Network (DQN) model. The research encapsulates a variety of traffic scenarios simulated via SUMO traffic simulator, and utilizes performance metrics such as total delay and total simulation time.

Upon evaluating the results, the DQN model consistently outperformed all other methods across all traffic scenarios. The modern AI-based DQN approach showcased superior efficiency, robustness, and adaptability, reducing waiting times significantly compared to other methods, even in highly complex scenarios. These findings indicate a potential shift in traffic signal control towards machine learning-based methods, particularly those deploying reinforcement learning like DQN.

5.1 Challenges

However, the implementation of DQN isn't without its challenges. The method incurs substantial installation costs, and the complexity of training the model can increase in exceptionally high traffic scenarios, particularly when traffic is congested due to blockage. Moreover, this method demands servers with high availability and reliable fallback mechanisms to ensure consistent performance. Lastly, the reliance on sensors, which are subject to potential issues and costly repairs, could lead to chaotic traffic situations if not managed properly.

5.2 Limitations

Some limitations that we had to follow in order to be able to conduct the simulations were:

- One type of vehicles: Standard size vehicles which is given by SUMO. Also no pedestrians were included in the simulations. As a result the generic traffic data were not real world derived but they were automatically created.
- Most of the intersections were 4-way intersections with two lanes in order to simplify the traffic scenarios.
- No special scenarios like accidents, constructions and emergency vehicles were included in the simulation scenarios.
- Fixed-phase durations: Fixed minimum green time of 10 simulation steps and fixed yellow phase time of 3 simulation steps.
- The DQN agents have to be trained for each scenario specifically because the traffic lights have different input and output states.

5.3 Future work

In the future we should delve into further research and experimentation in diverse traffic scenarios, including possible strategies to mitigate the aforementioned challenges. This research is vital before such systems can be seamlessly integrated into a real-world traffic light management system.

In detail, some real-world tests with a bigger variety of vehicles, extreme scenarios, multiple phase configurations and complex road networks could be conducted. Moreover, our DQN approach follows the pattern of a single agent per traffic light in order to simplify the state and action spaces. In theory, the ideal approach could be to have a single agent for the whole network which will have the context of the whole network and will be able to make the optimal decisions but this is extremely hard to scale in complex urban scenarios. Another approach to mitigate this scaling issue could be having agents that control a cluster of traffic lights, this could potentially solve the scaling issue because the state and action spaces would be limited.

Despite these challenges, the potential of the DQN model in optimizing traffic management is undeniable. Its robustness and adaptability, and its ability to learn from past experiences and adapt to dynamic traffic flows, make it a promising contender for future traffic management.

In conclusion, while classical methods hold their advantage in specific circumstances, the introduction of AI and machine learning in traffic signal control can revolutionize the industry, offering more efficient traffic management and substantially reducing waiting times. The future of urban traffic management appears to be taking a significant shift towards AI-based solutions, driving into an era of smarter, more responsive cities.

ABBREVIATIONS - ARCHIVES - ACRONYMS

SUMO	Simulation of Urban MObility
DQN	Deep Q-Network
DNN	Deep Neural Network
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
Tanh	Tangent hyperbolic
ReLU	Rectified Linear unit
MSE	Mean Squared Error
i.i.d.	independent and identically distributed
DTSE	Discrete Traffic State Estimator
DDPG	Deep Deterministic Policy Gradient
SOTL	Self-organizing Traffic Lights
CV	Connected Vehicles
DQTSCA	Deep Q-network Traffic Signal Control Agent
STSCA	Swallow network Traffic Signal Control Agent
DTSE	Discrete Traffic State Encoding
EGU-RL	Edge-Weighted Graph Convolutional Reinforcement Learning
GCN	Graph Convolutional Network
MARL-s	single-junction multi-agent RL
MARL-g	grouped multi-agent RL
FMA2C	Feudal Multi-agent Advantage Actor-Critic

REFERENCES

[1] Bresson A. (2016). The Making of the Ancient Greek Economy: Institutions, Markets, and Growth in the City-States.
[2] de Grummond N.T. (2006). Encyclopedia of the History of Classical Archaeology.
[3] Camp J. (2001). The Archaeology of Athens. New Haven: Yale University Press.
[4] Baines J., Málek J. (2000). Cultural Atlas of Ancient Egypt.
[5] Kemp B. J. (2006). Ancient Egypt: Anatomy of a Civilization.
[6] Silver M. (1991). Economic Structures of Antiquity.
[7] Casson L. (1994). Travel in the Ancient World.
[8] Aldrete G. S. (2004). Daily Life in the Roman City: Rome, Pompeii, and Ostia.
[9] Laurence R. (1999). The Roads of Roman Italy: Mobility and Cultural Change.
[10] Liu D. (2005). The Silk Road in World History.
[11] Needham J. (1965). Science and Civilization in China, Volume 4: Physics and Physical Technology, Part 2: Mechanical Engineering.
[12] Britnell R.H. (1996). Markets, Trade, and Economic Development in England and Europe, 1050-1550.
[13] Gies F., Gies J. (1994). Life in a Medieval City.
[14] Lilley K.D. (2002). Urban Life in the Middle Ages, 1000-1450.
[15] Bagwell P.S. (1994). The Transport Revolution from 1770.
[16] McShane C. (1999). Down the Asphalt Path: The Automobile and the American City.
[17] Miller G. (2006). The Automobile: A Chronology of Its Antecedents, Development, and Impact.
[18] Roess R.P., Prassas E.S., McShane W.R. (2011). Traffic Engineering (4th ed.).
[19] Webster F.V. (1958). Traffic Signal Settings. Road Research Technical Paper No. 39. London: Her Majesty's Stationery Office.
[20] Nathan H. Gartner, John D. C. Little, Gabbay H. (1975). Optimization of Traffic Signal Settings by Mixed-Integer Linear Programming: The MAXBAND Program. Transportation Research Record, 9(4), 321-343.
[21] Gartner N.H., Little J.C., Kelson M.D. (1984). MAXBAND: A versatile program for setting signals on arteries and triangular networks. Transportation Research Record Journal of the Transportation Research Board, 795, 40-46
[22] Zohdy I.H., Rakha H.A. (2013). Intersection Management via Vehicle Connectivity: The Intersection Cooperative Adaptive Cruise Control System Concept. Journal of Intelligent Transportation Systems, 20, 17-32.
[23] Varaiya P. (2013). Max pressure control of a network of signalized intersections. Transportation Research Part C: Emerging Technologies, 36, 177-195.
[24] Aboudolas K., Papageorgiou M., Kouvelas A. (2010). A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. Transportation Research Part C: Emerging Technologies, 18, 680-694.
[25] Watkins C.J., Dayan P. (1992). Q-learning. Machine Learning, 8, 279-292.
[26] Sutton R.S., Barto A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.
[27] Mnih V., Kavukcuoglu K., Silver D., Graves A., Antonoglou I., Wierstra D., Riedmiller M. (2013). Playing Atari with deep reinforcement learning. https://arxiv.org/abs/1312.5602
[28] Mnih V., Kavukcuoglu K., Silver D., Rusu A.A., Veness J., Bellemare M.G., Hassabis D. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529-533.
[29] Arulkumaran K., Deisenroth M. P., Brundage M., Bharath A. A. (2017). A brief survey of deep reinforcement learning. https://arxiv.org/abs/1708.05866
[30] LeCun Y., Bengio Y., Hinton G. (2015). Deep learning. Nature, 521, 436-444.
[31] Schmidhuber J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
[32] Goodfellow I., Bengio Y., Courville A. (2016). Deep Learning. MIT Press.
[33] Rumelhart D.E., Hinton G.E., Williams R.J. (1986). Learning representations by back-propagating errors. Nature, 323, 533-536.
[34] Kingma D.P., Ba J. (2014). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980
[35] Krizhevsky A., Sutskever I., Hinton G.E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems.
[36] McCulloch W.S., Pitts W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5, 115-133.
[37] Glorot X., Bordes A., Bengio Y. (2011). Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.
[38] Bishop C.M. (2006). Pattern Recognition and Machine Learning. Springer.
[39] Rumelhart D.E., Hinton G.E., Williams R.J. (1986). Learning representations by back-propagating errors. Nature, 323, 533-536.

[40] Hornik K., Stinchcombe M., White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2, 359-366.
[41] He K., Zhang X., Ren S., Sun J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision.
[42] Lin L.J. (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. Machine Learning, 8, 293-321.
[43] Ducrocq R., Farhi N.(2021). Deep Reinforcement Q-Learning for Intelligent Traffic Signal Control with Partial Detection https://arxiv.org/pdf/2109.14337.pdf
[44] Genders W., Razavi S. (2019). An Open-Source Framework for Adaptive Traffic Signal Control https://arxiv.org/pdf/1909.00395v1.pdf
[45] Genders W., Razavi S.(2016). Using a Deep Reinforcement Learning Agent for Traffic Signal Control https://arxiv.org/pdf/1611.01142.pdf
[46] Wang H., Chen H., Wu Q., Ma C, Li* Y., Shen C.(2019). Multi-intersection Traffic Optimisation: A Benchmark Dataset and a Strong Baseline https://arxiv.org/pdf/2101.09640.pdf
[47] Ault J., Sharon G. (2021). Reinforcement Learning Benchmarks for Traffic Signal Control https://people.engr.tamu.edu/guni/Papers/NeurIPS-signals.pdf