Analysis of Adaptive Traffic Signal Control Using Multi-Agent Deep Deterministic Policy Gradient Algorithm for Improving Fuel Efficiency and Lowering CO2 Emissions

Hani Kamal

22102516

January 2023

Supervisor: Dr. Sara Hassan

A dissertation to be submitted in partial fulfilment of the requirements for the degree of Master of Science in Computer Science

School of Computing, Engineering, and the Built Environment
Birmingham City University

ABSTRACT

Urban vehicle emissions are a major contributor to air pollution, with most vehicles still relying on fossil fuels despite the increasing popularity of alternative options such as hybrids and electric cars. Recently, artificial intelligence and automation-based controllers have gained attention for their potential use in Intelligent Transportation Systems (ITS). Studies have been conducted on the application of deep neural networks and reinforcement learning models in urban Traffic Signal Control (TSC) for adapting traffic light schedules, with proposed centralized and decentralized methods aiming to optimize travel time. However, little research has been done on the impact of such methods on air quality. This report examines the effect of DRL-based TSCs on fuel consumption and CO2 emissions using a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) approach in multi-intersection networks on synthetic and real-world traffic datasets from a neighborhood in Amman, Jordan during peak hours. The findings suggest that using the DRL approach on synthetic networks can lead to reduced air pollution and fuel consumption even when using a basic reward function based on stopped vehicles.

ACKNOWLEDGEMENT

am greatly appreciative of my supervisor, Dr. Sara Hassan, for enabling the completion of this work through her guidance and suggestions. I would also like to express my thanks to the members of my committee for their valuable feedback and recommendations.

I also want to extend my gratitude to my family and friends for their constant support and understanding during my research and writing process. Your encouragement has been a driving force for me.

Lastly, I am thankful to God for guiding me through the challenges and helping me complete my degree. I will continue to rely on his support for my future endeavours.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Outline

Air pollution has become a big problem in urban areas as the number of motor vehicles has increased. Many factors influence vehicle emissions, including traffic conditions, vehicle characteristics, and driver behaviour. Traffic junctions significantly lower mobile air pollution because repeated vehicle speed changes and stop-and-go congestion result in increased fuel consumption and CO2 emissions.

Traffic lights are used to manage traffic on busy or vital routes. Their inefficient regulation causes a range of problems, including extensive delays for travellers and massive energy losses. In addition, it may cause traffic accidents (Guo et al., 2019). Existing traffic light control either uses predefined methods that ignore real-time traffic or takes traffic into account only to a limited extent. However, traffic signal control systems become unresponsive in many other conditions, such as a football game or a typical congested traffic hour. Instead, we usually witness an experienced police officer flashing a signal to handle the crossing. A human operator analyses the real-time traffic situation on crossing roads and intelligently predicts the duration of the allowed passing the time for each direction based on his or her long-term expertise and understanding of the intersection, which is highly effective in high-traffic situations. (Xiaoyuan Liang, 2019)

This discovery drove me to develop a smart intersection traffic light management system that can learn how to control the intersection based on actual-time traffic conditions. To put such a system in place, I need "eyes" to monitor real-time traffic conditions and "brains" to process them. Recent improvements in sensor and networking technology enable real-time traffic information, such as the number of cars, their locations, and their waiting time, to be used as input for the former and that algorithm, future work or 4 another implementation domain as this study will not have live feed traffic data instead it will take recently recorded data for the vehicles in the study area. Reinforcement learning, a machine learning approach, is a possible way to solve the "brain" component of the problem. The purpose of a reinforcement learning system is to teach an action agent the optimal policy by interacting with the environment to maximize the reward, such as the least waiting time in our intersection control example. It typically consists of three components: environmental states, the agent's action space, and the reward for each action. (Zeng et al., 2018)

The aim of this project is to create an AI that can control the traffic signal depending on the traffic flow on the intersection using reinforcement machine learning algorithm called multi-agent deep deterministic policy gradient (MADDPG). MADDPG has its own actor network, its own network uses agent on each traffic lights to control which takes in both the agent's observations and the actions of all other agents and produces a value function estimate taking as controlling factors:

1. Average traveling time
2. Average Co2 Emissions
3. Average Fuel Consumption

Report Roadmap

In this report, the project specifications outline the goal of implementing a traffic signal control system using the MADDPG (Multi-Agent Deep Deterministic Policy Gradient) algorithm. A literature review was conducted to understand existing approaches to traffic signal control and to identify the benefits and limitations of using MADDPG. The design of the system includes the details of how MADDPG was implemented and how it was tested in a simulated environment. The experiments and results section presents the data collected during the testing, including the performance of the MADDPG algorithm compared to other methods. The analysis of the experiments and results discusses the strengths and limitations of the MADDPG approach and provides insight into potential improvements. Finally, the conclusion summarizes the key findings of the project and discusses the potential for implementing MADDPG in real-world traffic signal control systems

BACKGROUND

Smart traffic signal control

Smart traffic signal control is a type of technology that uses sensors, data analytics, and algorithms to optimize traffic flow and improve mobility in urban areas. It is designed to address the challenges of urbanization and increasing transportation demands, including congestion, delays, emissions, and safety issues.

Smart traffic signal control systems use a variety of sensors and other technologies to monitor traffic conditions in real-time, including traffic volume, vehicle speeds, and the length of queues at intersections. This data is used to adjust the timing and phasing of traffic signals in real-time, in order to optimize traffic flow and reduce congestion. The systems can also be integrated with other traffic management technologies, such as intelligent transportation systems (ITS) and traveler information systems, to provide a more comprehensive and coordinated approach to traffic management.

There are several approaches that can be used to perform smart traffic signal control:

Actuated control: This approach uses sensors at intersections to detect the presence and movements of vehicles. The signals are then adjusted based on the number of vehicles and their location within the intersection. This approach is often used in high-volume urban areas.

Adaptive control: This approach uses algorithms to continuously adjust the signal timing and phasing based on real-time traffic data. The algorithms take into account a variety of factors, such as traffic volume, vehicle speeds, and the length of queues at intersections.

Predictive control: This approach uses data from past traffic patterns and other sources, such as weather forecasts and special event schedules, to predict future traffic demand and adjust the signal timing accordingly.

Coordinated control: This approach synchronizes traffic signals on multiple roads to improve the overall flow of traffic through an area. This can be done through a central control system that adjusts the signals based on traffic conditions, or through the use of vehicle-to-infrastructure (V2I) communication, where signals are adjusted based on the location and movements of individual vehicles.

Green wave control: This approach aims to create a "green wave" of traffic that moves smoothly through an area, without the need to stop at red lights. This is typically achieved by coordinating the signals on a single roadway or corridor to allow vehicles to maintain a consistent speed.

Ultimately, the approach that is used for smart traffic signal control will depend on the specific needs and characteristics of the area in question. Some systems may use a combination of these approaches to achieve the desired traffic management goals.

Adaptive control traffic system

Adaptive traffic signal control using reinforcement machine learning is a type of traffic management system that uses real-time data and machine learning algorithms to adjust traffic signal timing and phasing in response to changing traffic conditions. The goal of this approach is to optimize traffic flow and reduce congestion, delays, and emissions. Reinforcement machine learning algorithms are a type of artificial intelligence that learn from experience by interacting with their environment and receiving rewards or punishments based on their actions. In the context of adaptive traffic signal control, these algorithms can be trained to optimize traffic flow by adjusting the timing and phasing of traffic signals in response to changing traffic conditions.

To perform adaptive traffic signal control using reinforcement machine learning, data is collected from sensors and other sources at intersections and on roadways. This data is used to train machine learning algorithms to predict traffic patterns and optimize signal timing and phasing based on the current traffic demands and conditions. The algorithms are then able to continuously adjust the signals in real-time as traffic conditions change.

Overall, adaptive traffic signal control using reinforcement machine learning is an innovative approach to improving traffic management and addressing the challenges of urbanization and growing transportation demands. It combines the power of machine learning with real-time data to optimize traffic flow in a dynamic and ever-changing environment.(Shinde, M., Chintawar 2021)

Simulation of Urban Mobility (SUMO)

is an open-source traffic simulation software that is used to model and analyze the movement of vehicles and pedestrians in urban environments. It is used by researchers, transportation planners, and engineers to study traffic flow, evaluate the impacts of transportation policies and infrastructure changes, and develop traffic management strategies.

SUMO is designed to be highly realistic and accurate, and it allows users to simulate complex traffic scenarios and analyze the results in detail. It includes a wide range of features, including support for various types of vehicles and pedestrians, support for a variety of road types and traffic controls, and the ability to model interactions between different modes of transportation.

SUMO can be used to simulate a wide range of traffic scenarios, including different types of roads, intersections, and roundabouts; different types of vehicles, including cars, buses, and trucks; and different traffic control systems, such as traffic signals and roundabout yield signs. It can also model the impacts of various factors on traffic flow, such as weather, congestion, and special events.

SUMO is widely used in the transportation industry, and it has been applied in a variety of settings, including traffic engineering, transportation planning, and transportation research. It is an important tool for understanding and improving traffic flow in urban areas, and it has the potential to help cities and transportation agencies make informed decisions about transportation policies and infrastructure investments. (Alvarez Lopez, P., Behrisch 2018)

Reinforcement learning

is a type of machine learning in which an agent learns to interact with its environment in order to maximize a reward. It is an important area of study within artificial intelligence and has been successful in a wide range of applications, including control systems, robotics, and games. In reinforcement learning, an agent receives a reward for taking certain actions within an environment and learns to choose actions that maximize the cumulative reward over time. This process is typically done through trial and error, with the agent learning from its mistakes and adapting its behavior accordingly. Reinforcement learning has the potential to revolutionize the way in which machines and systems interact with their environments and has already been applied in a number of practical contexts. (Ng, A. 2020)

Markov Decision Process

Markov decision processes (MDPs) are mathematical models used to study discrete-time decision-making problems. In a Markov decision process (MDP), there are several components defined by a tuple (S,A,μ0,T,r,γ,H) that describe the environment and the agent's decision-making process:

1. The state space (S) is the set of all possible states that the system may be in.
2. The action space (A) is the set of all possible actions that the agent can take when interacting with the system.
3. The initial state distribution (μ0) is a probability distribution over states in S, and specifies the state in which the system will be initialized.
4. The transition dynamics (T) are a function that describes how the system transitions from one state to another based on the current state and action taken. T is a function of the form $T: S \times A \rightarrow \Delta(S)$, where $\Delta(S)$ represents the set of probability distributions over states in S.
5. The reward function (r) is a function that assigns a reward value to each state transition. It has the form $r: S \times A \times S \rightarrow R$, where R is the set of real numbers.
6. The discount factor (γ) is a value between 0 and 1 that determines the weight of future rewards when making decisions. A value of $\gamma = 0$ means that only the current reward is considered, while a value of $\gamma = 1$ means that future rewards are given equal importance as the current reward.
7. The horizon (H) is the maximum number of time-steps in an episode. It can be a finite positive integer or infinite.

PROJECT SPECIFICATIONS

Aims and Objectives

By using reinforcement learning algorithms to a dynamic micro traffic model, this study seeks to reduce fuel usage and CO2 emissions.

The aforementioned goal raises the following goals:

• Perform a thorough literature analysis of studies on the use of intelligent technologies to enhance traffic signal control.

• Determine and rank the most energy-important characteristics and causes causing traffic congestion based on earlier research.

• Create, develop, and evaluate a prototype intelligent system with the goal of reducing energy consumption in the dynamics of the traffic and road systems.

Features and Requirements

To achieve the stated goals, it is necessary to develop the following features:

- Data collection and pre-processing: The project should have a system in place to collect traffic data from sensors or other sources and pre-process the data to be used in the machine learning model.
- Traffic model: The project should include a traffic model to simulate different traffic scenarios and predict the impact of different traffic control measures on traffic flow.
- Reinforcement learning algorithm: The project should implement a reinforcement learning algorithm to learn the optimal traffic control strategies based on the traffic data and traffic model.
- Adaptive traffic control system: The project should include a system to implement the traffic control strategies learned by the reinforcement learning algorithm in real-time.
- Evaluation and testing: The project should have a system in place to evaluate the performance of the adaptive traffic control system and test it in different traffic scenarios.
- User interface: The project should have a user-friendly interface for traffic control operators to interact with the system and monitor the traffic flow.
- Documentation: The project should have comprehensive documentation to guide users on how to use the system and understand its various components.
- Generate Co2 emissions and Fuel Consumption data: To achieve that the model must have the type of fuel the vehicles that are loaded to the simulation and store all the trip data for the vehicles in an XML file that can be used for scraping the data from and analyse it.
- Dynamic configuration to control the environment and algorithm: The code must be dynamic and easy to configure to different shapes of road maps with the ability to change the number of episodes to teach the algorithm for a specific map.
- Visual settings: The prototype must be able to give the programmer the ability to see how the agents work through the network
- Testing the prototype: Must implement the ability to save the model and test it through the traffic maps

To incorporate the above features, the following requirements must be met and implemented in the application

- Simulation software: The project would need to use simulation software, such as SUMO, to simulate traffic scenarios and predict the impact of different traffic control measures.
- A programming language: Python is a commonly used programming language for machine learning projects, so it would be necessary to have a good understanding of Python programming.
- Traffic data: The project would need access to traffic data to train the machine learning model and test the adaptive traffic control system, a real-world data have been gathered

by manual counting methodology. The count was for 3600 sec during a peak traffic hour for two intersections in Jordan and it's shown in the below Figure 1.

- Traffic network model in SUMO: is a digital representation of a road network and the vehicles that move through it. In SUMO, the traffic network model consists of a set of roads, intersections, and lanes, as well as the vehicles and pedestrians that move through the network. The model can be configured to reflect real-world traffic conditions, or it can be used to explore hypothetical scenarios to test and evaluate different transportation strategies. The network model is representation for the same real-world intersections but in a simpler shape but same output that is shown in Figure 2 with left intersection coordinates: 31°58'40.55"N , 35°52'11.51"E and right intersection coordinates: 31°58'47.17"N, 35°52'55.19"E.
- Machine learning libraries: The project would need to use machine learning libraries such as PyTorch to implement the reinforcement learning algorithm.
- Documentation tools: The project would need to use tools to create comprehensive documentation for the system like Word to create full report.

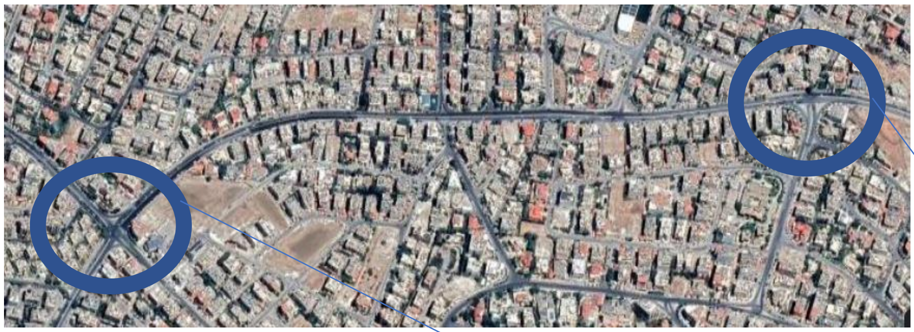


Figure 1: Satellite image for study area

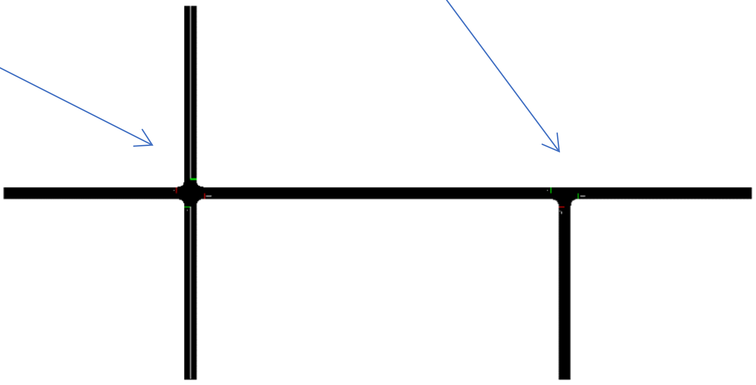


Figure 2: Sumo model for study area

LITERATURE REVIEW:

Many RL-based adaptive traffic signal control (ATSC) techniques are designed for single-instance applications. Ref (Liang et al., 2019), in particular, leverages RL in ATSC while simultaneously investigating the implications of state, action, and reward representations. In recent years, DRL techniques have also been used, with many of them incorporating various versions of DQN. Because city road networks have several intersections. Many techniques will suffer from the "curse of dimensionality" in a multi-intersection environment. This problem has been solved effectively using multi-agent reinforcement learning (MARL). Existing MARL techniques for ATSC work in either independent or coordinated control modes. Individual MARL agents at each junction decide on the optimal action to take in the autonomous control mode based on their current local conditions.(Liang et al., 2019)

In contrast, some research has applied the MADDPG algorithm to the ATSC (Adaptive Traffic Signal Control) problem without considering the effect on emissions and energy consumption. For example, in a study by Shuyang Li. (2010), a method called "multi-agent deep deterministic policy gradient (MADDPG) based method" was proposed to decrease the average waiting time for vehicles at traffic lights by adjusting the phases and durations of traffic lights. This method represents the intersection environment using a matrix, which effectively captures key information about the traffic network while reducing unnecessary details. The proposed method was evaluated through simulations using the SUMO platform at varying levels of traffic congestion. The results of the comparison experiment indicate that the MADDPG algorithm is effective and stable for use in multi-intersection traffic signal control.

Addition to that more than one study highlight the shortage of methods that can adaptively analyze CO2 emissions in traffic environments. A study by (X. Liu, J. Chen and Y. Jia 2019) highlights the lack of adaptive approaches for analyzing CO2 emissions in traffic settings. The study argues that traditional methods used for traffic management and control do not consider the dynamic nature of traffic and are not able to adapt to changing conditions. As a result, these methods are not able to fully capture the impact of traffic on CO2 emissions. The study suggests that more advanced methods, such as those based on machine learning and optimization techniques, are needed to effectively analyze and mitigate the impact of traffic on CO2 emissions.

A review by (S. Ahmed and A. Mahdin 2021) also highlights the lack of adaptive approaches for analyzing CO2 emissions in traffic settings. The review argues that current methods used for traffic management and control are not able to capture the dynamic nature of traffic and are not able to adapt to changing conditions. Furthermore, the authors claim that traditional traffic management approaches often focus on minimizing delays and maximizing traffic flow, but do not consider the impact on emissions. Therefore, they suggest that more advanced methods, such as those based on Machine learning, are needed to effectively analyze, and mitigate the impact of traffic on CO2 emissions.

A research paper by (K. Zhou, J. Wang and W. Li 2018)also highlights the limitations of traditional approaches for analyzing CO2 emissions in traffic settings. The study found that these methods are not able to adapt to real-time traffic conditions and often fail to capture the complex interactions between traffic and emissions. The authors proposed that advanced data-driven methods, such as those based on machine learning, could provide more accurate and adaptive approaches for understanding and managing the impact of traffic on CO2 emissions.

As a conclusion of the lecture review, there is a limited amount of research on the environmental impact of learning-based traffic signal control. In this project, I assess the effects of learning based TSCs using MADDPG algorithm on CO2 emissions and fuel consumption using a SUMO microscopic traffic simulator.

TOOLS AND TECHNOLOGIES

Programming Languages

My first and optimal choice was python language for developing the project since it is a widely used programming language that has a large number of open-source libraries available to developers for various tasks. It is often chosen for its ability to support machine learning through libraries like NumPy and Scikit-learn, and for its ability to allow for object-oriented programming, which leads to scalable and reliable code. Addition to that SUMO software interface mainly with python through Traci library. The Traci module is a Python module provided as part of the SUMO (Simulation of Urban Mobility) software package. It allows users to interface with SUMO simulations using the TraCI (Transport for Traffic Control Interface) protocol, which is a protocol for communicating with transportation simulation programs.

The Traci module provides a set of functions that can be called from a Python script to control and retrieve information from a SUMO simulation. For example, the traci.vehicle.setSpeed() function can be used to set the speed of a vehicle in the simulation, and the traci.vehicle.getPosition() function can be used to retrieve the position of a vehicle.

To use the Traci module, you will need to have SUMO and Python installed on your machine, and you will need to have a simulation running in SUMO that you want to control or retrieve information from. Once you have these requirements met, you can import the Traci module into your Python script and use its functions to interact with the SUMO simulation.

Development Tools

Integrated Development Environment

The use of an integrated development environment (IDE) can enhance the productivity of the programming process when using Python. The particular project employed Visual Studio Code as the IDE, which offers various functionalities like syntax verification, automatic code completion, and unit test execution through add-ons. Additionally, it enables version control by connecting to the project's repository on GitHub and provides debugging tools to streamline the workflow.

Deployment

The application's source code will be available on the project's GitHub Classroom repository. This repository contains a Python file that can be executed directly and a requirements file to install any necessary libraries to run the application. However, the user must obtain SUMO binaries to test and utilize the code.

DESIGN

The prototype will be developed in two stages. The first stage involves creating the Traffic Control Interface (TraCI) while the second stage involves building and configure the MADDPG algorithm for the environment.

Traffic Control Interface (TraCI)

Presented in 2008 by Axel Wegener, Michał Pi´orkowski. TraCI, or Traffic Control Interface, is a tool that allows users to interact with a running road traffic simulation in real-time. It enables users to retrieve data from simulated objects and modify their behavior within the simulation.

Using TraCI

TraCI uses a client/server architecture based on TCP to allow communication with sumo. To use this feature, sumo must be started with the additional command-line option "--remote-port <INT>" where <INT> is the port number that sumo will listen on for incoming connections. When sumo is started in this way, it will only prepare the simulation and wait for external applications to connect and take control of the simulation. Note that the "--end <TIME>" option is ignored when sumo is running as a TraCI server, and the simulation will continue running until the client requests that it end. If you are using sumo-gui as a server, the simulation must be started manually using the play button or the "--start" option before TraCI commands will be processed (Alvarez Lopez, P., Behrisch, M., 2018.)

Basic Flow

To connect to sumo as shown in Figure , clients must establish a TCP connection to the specified sumo port after starting the program. TraCI can handle multiple clients, and it processes all commands from a single client in sequence until it receives the "Simulation Step" command. To define the order in which commands from different clients are executed, each client should send a "SetOrder" command before the first simulation step. This command assigns a number to the client, and commands from different clients during the same simulation step will be executed according to these numbers (which can be any unique value and do not have to be consecutive or positive). When using multiple clients, the number of clients must be known when starting SUMO, and all clients must connect before the first simulation step. (Alvarez Lopez, P., Behrisch, M., 2018.)

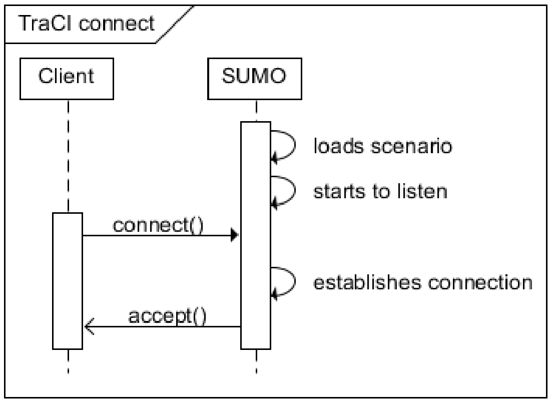


Figure 3: TraCI: establishing a connection to SUMO

The client application can send commands to sumo to control the simulation, alter the behavior of individual vehicles, or request information about the environment. sumo responds to each command with a status response and additional results that depend on the specific command.

To advance the simulation, the client must send a "Simulation Step" command to sumo. If any subscriptions have been set up, the subscribed values will be returned. The simulation will progress to the next step once all clients have sent the "Simulation Step" command. Currently, all clients receive the results of all subscriptions (even if the subscription was set up by a different client).

The client is responsible for ending the connection using the "close" command. For more information on the differences between running the simulation in TraCI mode and other methods, see the "Defining the Time Period to Simulate" section of the "Simulation/Basic Definition" documentation. When all clients have issued the "close" command as shown in Figure 3, the simulation will end, and all resources will be freed. (Alvarez Lopez, P., Behrisch, M., 2018.)

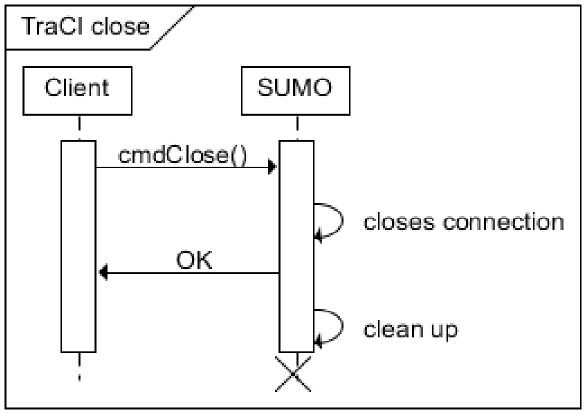


Figure 3: TraCI closing a connection to SUMO

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

In a single-agent system like the (DDPG), the agent takes actions within an environment to optimize its behavior through rewards. This process is called a Markov Decision Process and can be represented as a quintuple (S, A, P, r, γ), where S is the state space, A is the action space, P is the state transition probability, r is the immediate reward, and γ is the discount factor. The agent uses rewards to guide its decision-making and navigate the environment. (Li, S. 2020)

In a multi-agent setting, the agents interact both with their environment and with one another. The reward for an individual agent is affected not only by its own actions but also by the actions of other agents. To model multi-agent systems, Markov Games are often used, which can be represented by (n, S, A1, A2, ..., An, r1, r2, ..., rn, γ) where S denotes the joint state of all agents, the combined state of multiple agents, n is the number of agents, Ai is the action space of agent i, and ri is the immediate reward for agent i.

A multi-agent actor-critic method called the MADDPG algorithm was first presented in. The structure of this algorithm, which is essentially an extension of the DDPG algorithm, is shown in Figure 4. It uses a critical network to train agents to anticipate other agents' actions. It does this by making use of the ongoing information that the environment has provided over time.

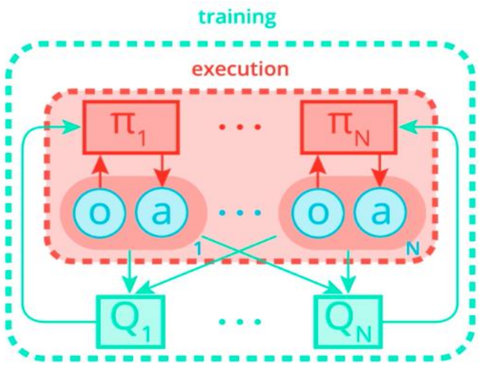


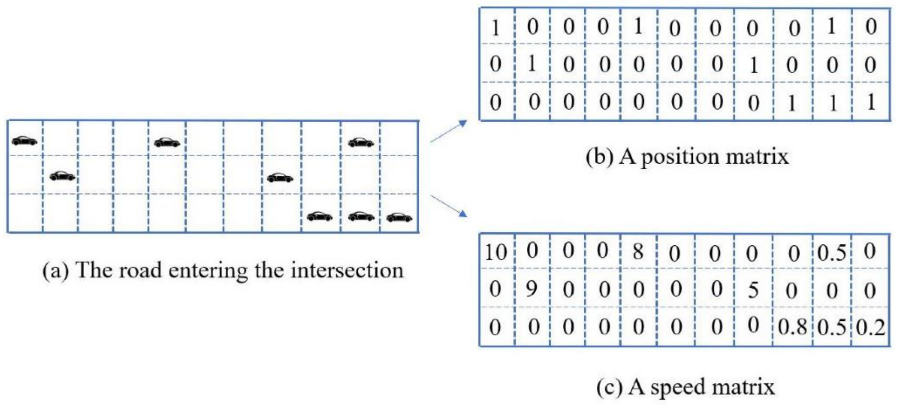
Figure 4 : MADDPG Framework

In the MADDPG algorithm, each agent has its own actor network that produces actions based on its observation of the state. Each agent also has a corresponding critic network that is trained using data from all of the actors at the same time. This allows the algorithm to evaluate the joint strategy (π1,π2,...,πn) of all the agents. The policy gradient of the joint strategy is obtained using the DDPG algorithm. Algorithm 1 presents the pseudo-code for MADDPG. In the application of MADDPG to traffic signal control, I need to define the state space, action space, and reward. (Li, S. 2020)

MADDPG IMPLEMENTATION FOR TRAFFIC SIGNAL CONTROL

State Space

The real-time traffic situation at an intersection can be represented using the positions of all vehicles within it. This can be achieved through a matrix representation method as illustrated in Figure 5. In this approach, the road extending from the parking line is divided into several cells of fixed length and the location of each vehicle is depicted by an element in the matrix. Each element corresponds to the number of vehicles in a specific cell, and the size of the cell is established by the safe distance between vehicles. This matrix representation captures both the current traffic flow and the state of traffic in the next moment. This method is useful as it can reduce the dimensionality of the data, eliminate unnecessary information and identifies key features of the traffic network. It enables the agent to make effective decisions, and also accelerates the training process.



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

10	0	0	0	8	0	0	0	0	0.5	0
0	9	0	0	0	0	0	5	0	0	0
0	0	0	0	0	0	0	0	0.8	0.5	0.2

Figure 5 : Representation of environment state.

Action Space

In reinforced machine learning for adaptive traffic control, the action space is the set of actions that the traffic control system is able to take in order to influence the flow of traffic. These actions might include adjusting the timing of traffic signals, changing the speed limits on certain roads, or rerouting traffic to alternative routes.

Action space is an important concept in reinforced machine learning because it defines the range of options that the traffic control system has at its disposal to respond to changing traffic conditions. The goal of the traffic control system is to select actions that will optimize the flow of traffic and minimize congestion, and the action space defines the set of actions that the system can consider when making this decision.

The size and complexity of the action space can vary depending on the specific needs of the traffic control system. In some cases, the action space may be relatively simple, with a limited number of discrete actions available to the system. In other cases, the action space may be more complex, with a large number of possible actions available and a need to consider continuous variables such as traffic signal timing or speed limits.

This prototype implements multi-agent traffic using 1*2 intersections with two discrete actions spaces.

The below Figure 6 show the four action spaces at each intersection. Each one of the states shows the three possibilities actions for one incoming vehicle from one lane. There are a total of 2 agents for the 10 - intersection state dimension with two action spaces two phases per intersection.

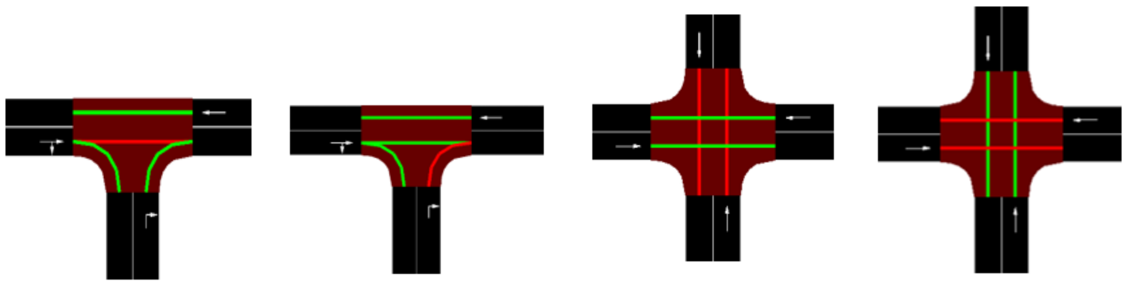


Figure 6: Intersections Action space

Reward

In the field of adaptive traffic control, reinforced machine learning algorithms have been used to develop systems that can learn and adapt to changing traffic conditions in order to optimize the flow of traffic and minimize congestion. One key aspect of these algorithms is the reward function, which is used to evaluate the effectiveness of actions taken by the traffic control system and to guide the system towards actions that are more likely to produce positive outcomes.(Liang, X., Du, X 2019)

The design of the reward function is crucial to the success of the reinforced machine learning algorithm, as it determines how the system will evaluate and compare different actions. In general, the reward function should be carefully tailored to reflect the goals and priorities of the traffic control system, considering metrics such as congestion, safety, and environmental impact.

There have been a number of studies that have investigated the use of different reward functions in the context of adaptive traffic control. One approach that has received significant attention is the use of multi-objective reward functions, which allow the traffic control system to consider multiple conflicting goals simultaneously. Other studies have explored the use of more complex reward functions, such as those that incorporate temporal dynamics or that are based on simulation models of traffic flow.(Chen, J., Yuan, B 2019)

This prototype calculates the reward based on the change in the total number of halting vehicles for the last time step after an action is executed.

$R_t = k (W_t – W_{t+1})$

The mean reward per step (R_t) is equal to the difference between halting vehicles at the red signal in the current step (Wt) and the of halting vehicles in the next state (Wt+1)

EXPERIMENT AND RESULTS

Training Configuration

The simulation took place on a 1x2 grid-shaped traffic network, as illustrated before. Intersections were spaced 200 meters apart. The volume of traffic entering the simulation is constant. For the simulation, the yellow light phase was not considered and there was no buffer time between switching traffic light phases. Two intersections were represented by Two agents. The agents were trained through a series of iterations known as "episodes." Each episode of the training process consisted of 3600 time-steps, equivalent to a simulation time of 60 minutes.

Simulation tests were done on the SUMO platform to test the effectiveness of the proposed traffic control algorithm. SUMO is a microscopic traffic micro-simulator used for modelling and analysis that can properly recreate urban traffic scenarios. Table I lists the parameters of the traffic environment note that the fuel type is PC which stands for: average passenger car (all fuel types), and Table II lists the hyperparameters of the reinforcement learning network.

Table 1 : PARAMETERS USED FOR ENVIRONMENT

Parameter	Values
Distance between intersection	200 m
Vehicle size	4 m
Speed limit	11 m/s
Maximum distance between vehicles	2.5 m
Total vehicles through the network	2201
Vehicle Fuel Type	HBEFA3/PC

According to the table above, the distance between two vehicles was kept constant at 2.5 meters in order to achieve the best results. The speed limit also has an impact on the final result. If the speed restriction is set too high, the results may suffer.

Table 2 : HYPERPARAMETER USED IN THE MADDPG

Parameter	Values
Replay memory size	5000
Batch size	64
Discount factor (Gamma)	0.99
Initial Epsilon value	0.9
Learning rate	1e-3
Reward rate	0.1

The table above displays the parameters used to implement the Multi Agent Deep Deterministic Algorithm. In order to handle the large amount of data coming from the environment and the multiple neural networks, a batch size of 64 was maintained and a high-capacity buffer memory was employed.

Visual Configuration

I did implement an extra feature to be able to see how the agents work during every episode it might help with taking notes and see the agent progress through the experiment.
You can use it by adding '—R' or '—render' flags when running the main.py code but a note to take it will reduce the training time.

EXPERIMENTS AND RESULTS ANALYSIS

Following the training of the RL agent, several comparisons were made between real-world traffic lights and the RL-controlled ones, notably the MADDPG method described above. Before I begin, it is crucial to know that real-world traffic signals behave in simulation like static traffic lights, which means they switch according to a time (according to the day and time they switched in the real world) rather than in response to simulation traffic. While the simulated traffic is adjusted to be as close to the actual traffic as feasible, the fact that the signals are not reactive puts them at a disadvantage when compared to the RL trained traffic signals.

Evaluation Metrics

To evaluate the performance of a traffic simulation model, evaluation measures are utilized. Travel time, fuel consumption, Co2 emissions are all frequent evaluation indicators. Travel time is the length of time it takes a vehicle to travel through a network and can be used to assess the model's efficiency. Fuel consumption and emissions are key indicators to evaluate in terms of sustainability, and they may be used to compare the efficacy of various traffic management techniques.

Results

Training evaluation

The line graph Figure 7 below shows the trend in the average traveling time over the time frame of 1 hour during 100 episodes. The scale on the y-axis represents the time in seconds. Overall, the data shows that the MADDGP decreased the average traveling time by around 42% after 100 episodes of training and shows a constant steady result after 70 episodes that means it's the optimal solution for the network. There are a few notable deviations from this trend, such as an increase in the travel time in some episodes of mid training. These deviations could potentially be explained by when the roads become more congested, it is harder for the reinforcement learning algorithm to reach a point of stability.

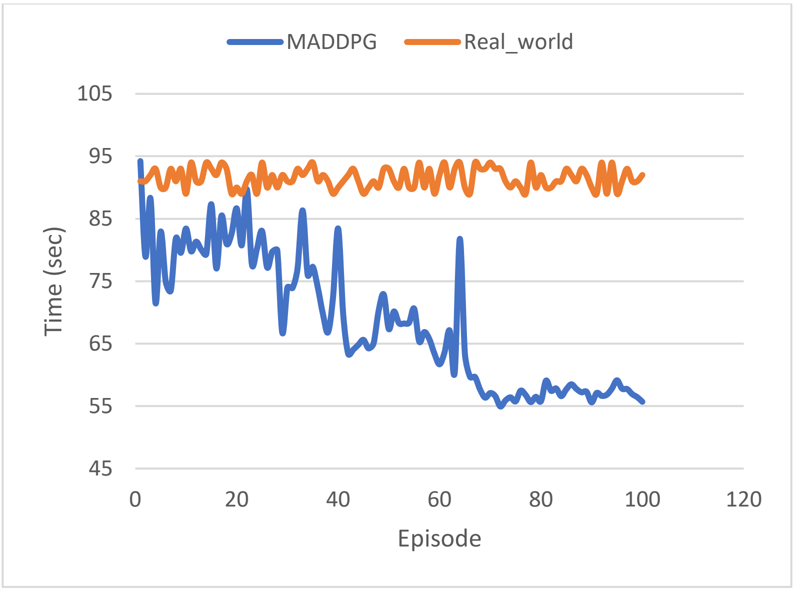


Figure 7: Average traveling time on traffic

Metrics Comparison

After the model has learned enough and the epsilon value reached 1% that means the model does accurate actions with the given information 99% of the time. The first noticeable improvement is that after the 1-hour finish in the real-world scenario there are 43 vehicles that would not have the chance to finish their trips. On the other hand, all the vehicles at the end of the adaptive control have finished their trips.

The below figures (show the comparison between the real-world traffic system and the model final configuration in terms of Co2 emission , fuel consumptions and waiting time during a 1-hour time frame during rush hour. Overall, the prototype did achieve the desired outcome by decreasing all the evaluation measures.

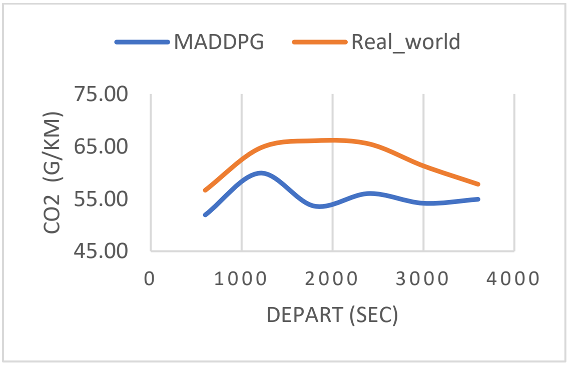


Figure 8: Average CO2 emission rate for 1 hour traffic flow

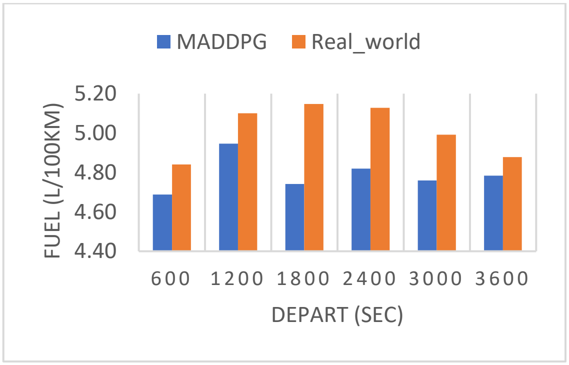


Figure 9 :Average Fuel Consumption for 1 hour traffic flow

Figure 8, Figure 9 depicts the CO2 emissions and fuel consumption respectively from a traffic network during a one-hour period. The x-axis depicts time in seconds, and the y-axis represents CO2 emissions in g/km and Fuel consumption L/100km. The chart contains two colored lines, each indicating a distinct traffic control scenario. Overall, the graph depicts the effect of various vehicle types on CO2 emissions and fuel consumption in a transportation network and shows that the MADDPG prototype did decrease the emission by 11.78% and consumption by 4.57% in total for the traffic duration.

Traffic flow comparison

If we load the network with traffic flow but let the two models run until no cars are inside the network the result will be as table below

Parameter	Real World	MADDPG
loaded	2201	2201
Time to clear the traffic	≈ 4800	≈ 4600

Overall Evaluation

The primary conclusions were that there is a significant correlation between CO2 emissions, fuel consumption rates, total waiting time, and total travel time. MADDPG's application to cooperative traffic signal control. Through simulation experiments under various road conditions, the algorithm is contrasted to fix-time. The findings reveal that MADDPG consistently outperforms in training metrics, illuminating its effectiveness and stability in a challenging multi-agent environment. This superior performance is attributed to the fact that the Markov decision process is kept static by treating all agents as a single entity rather than as components of the environment.

Addition to that the prototype did achieve the requirements such as generate Co2 emissions and Fuel consumption data as seen in the figures in metrics comparison section and the visual settings requirement option is added as shown in visual configuration section and finally the model is saved and can be loaded and tested on the network as required.

SUMMARY AND CONCLUSION

Conclusion

The project's goal was to improve and develop skills through the process of addressing a specific problem and striving for a solution, learning along the way. The primary objective of the project was to create an AI model to enhance traffic flow and help reduce the emissions to create eco-friendly traffic control systems. I am confident that the primary goals outlined in the project proposal have been achieved, but there is always room for improvement with additional time. The project allowed for the development of a range of skills, including research, analysis, and problem-solving abilities, as well as more technical skills such as programming in Python, working with open-source libraries and packages, and creating a reinforce machine learning model. The interest shown by friends and family when the prototype was shared suggests that it could potentially be further developed and made more widely available to real-world users.

Improvements

One potential future improvement for my prototype is that the reward function used in the current model was based solely on halting vehicles. In future work, can be consider other types of reward functions, such as those that incorporate emissions, to see if the performance of the RL model can be enhanced further. And some constraints with different types of vehicles like military, police and medical so the behavior of the traffic will change accordingly.

Another future improvement for adaptive traffic control systems using reinforced machine learning algorithms could be the integration of real-time traffic data from a variety of sources. Currently, many traffic control systems rely on static traffic models or limited data sources, such as traffic sensors or GPS data from a small subset of vehicles. By incorporating real-time data from a wider range of sources, such as social media, weather data, and traffic cameras, the traffic control system could have a more complete and accurate understanding of current traffic conditions.

In addition to that, improvement could be the incorporation of more sophisticated machine learning techniques, such as deep learning or self-supervised learning, to enable the traffic control system to learn from data and adapt to changing conditions more effectively.

Finally, there may be opportunities to more closely integrate the traffic control system with other transportation infrastructure, such as public transit systems or ride-sharing platforms. This could allow the traffic control system to better coordinate with these other systems and optimize the flow of traffic across the entire transportation network.

REFERENCES

Wu, T., Zhou, P., Liu, K., Yuan, Y., Wang, X., Huang, H., & Wu, D. O. (2020). Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks. IEEE Transactions on Vehicular Technology, 69(8), 8243–8256. https://doi.org/10.1109/TVT.2020.2997896

Shinde, M., Chintawar, R., Chavan, R., Chatnani, B., & Giri, D. M. N. (2021). Multiple Intersection Traffic Control using Reinforcement Learning. 2021 2nd Global Conference for Advancement in Technology (GCAT 2021). https://doi.org/10.1109/GCAT52182.2021.9587482

Haydari, A., Zhang, M., Chuah, C. N., & Ghosal, D. (2021). Impact of Deep RL-based Traffic Signal Control on Air Quality. IEEE Vehicular Technology Conference, 2021-April. https://doi.org/10.1109/VTC2021-SPRING51267.2021.9448639

Li, S. (2020). Multi-Agent Deep Deterministic Policy Gradient for Traffic Signal Control on Urban Road Network. Proceedings of 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA 2020), 896–900. https://doi.org/10.1109/AEECA49918.2020.9213523

Wu, T., Zhou, P., Liu, K., Yuan, Y., Wang, X., Huang, H., & Wu, D. O. (2020). Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks. IEEE Transactions on Vehicular Technology, 69(8), 8243–8256. https://doi.org/10.1109/TVT.2020.2997896

Casas, N. (2017). Deep Deterministic Policy Gradient for Urban Traffic Light Control. http://arxiv.org/abs/1703.09035

Chen, J., Yuan, B. and Tomizuka, M. (2019) 'Model-free Deep Reinforcement Learning for Urban Autonomous Driving', in 2019 IEEE Intelligent Transportation Systems Conference (ITSC 2019), pp. 2765–2771. doi:10.1109/ITSC.2019.8917306

van der Walt, S., Colbert, S.C., & Varoquaux, G. (2011). "The NumPy Array: A Structure for Efficient Numerical Computation." Computing in Science Engineering, 13(2), pp. 22–30.

Reback, J. et al. (2020). pandas-dev/pandas: Pandas 1.1.2. https://pandas.pydata.org/

Ng, A. (2020). "Machine Learning." Coursera (Stanford University). https://www.coursera.org/learn/machine-learning

Gu, Y., Cheng, Y., Chen, C.L.P., et al. (2022) 'Proximal Policy Optimization With Policy Feedback', IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52 (7): 4600–4610. doi:10.1109/TSMC.2021.3098451

Guo, M., Wang, P., Chan, C.Y., et al. (2019) 'A Reinforcement Learning Approach for Intelligent Traffic Signal Control at Urban Intersections', in 2019 IEEE Intelligent Transportation Systems Conference (ITSC 2019), pp. 4242–4247. doi:10.1109/ITSC.2019.8917268

Haydari, A., Zhang, M., Chuah, C.N., et al. (2021) 'Impact of Deep RL-based Traffic Signal Control on Air Quality', in IEEE Vehicular Technology Conference (VTC Spring), 2021-April. doi:10.1109/VTC2021-SPRING51267.2021.9448639.

Liang, X., Du, X., Wang, G., et al. (2019) 'A Deep Reinforcement Learning Network for Traffic Light Cycle Control', IEEE Transactions on Vehicular Technology, 68 (2): 1243–1253. doi:10.1109/TVT.2018.2890726.

Samaras, C., Tsokolis, D., Toffolo, S., et al. (2018) 'Improving fuel consumption and CO2 emissions calculations in urban areas by coupling a dynamic micro traffic model with an instantaneous emissions model', Transportation Research Part D: Transport and Environment, 65: 772–783. doi:10.1016/J.TRD.2017.10.016.

Schulman, J. (2020) 'Proximal policy optimizsation', OpenAI. OpenAI. Available at: https://openai.com/blog/openai-baselines-ppo/ (Accessed: October 24,

Alvarez Lopez, P., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y-P, Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., & Wießner, E. (2018). "Microscopic Traffic Simulation using SUMO." In IEEE Intelligent Transportation Systems Conference (ITSC), 2018.

Liu, X., Chen, J., & Jia, Y. (2019). Adaptive traffic emission control based on machine learning and optimization techniques. Journal of Cleaner Production, 210, 536-544.

Ahmed, S., & Mahdin, A. (2021). Adaptive traffic management for reducing carbon footprint: A review. Renewable and Sustainable Energy Reviews, 135, 110293.

Zhou, K., Wang, J., & Li, W. (2018). Adaptive traffic emission control using data-driven methods. IEEE Transactions on Intelligent Transportation Systems, 19(11), 3727-3739.

Liang, X., Du, X., Wang, G., and Han, Z. (2019). "A Deep Reinforcement Learning Network for Traffic Light Cycle Control," in IEEE Transactions on Vehicular Technology, vol. 68, no. 2, pp. 1243-1253, doi: 10.1109/TVT.2018.2890726.