Faculty of Computer and Data Science,

Alexandria University

The graduation project was submitted in partial fulfillment of the B.SC. Degree

Traffic Manager (2024-2025)

A system that aims for less congestion, and better transportation

Authors:

Ahmed Ibrahim Hassan

Bichay Adel Ramses

Emad Mohammed Fawzy

Kerolss Bolis Shehata

Marwa Shebl Mohammed

Mayan Islam Mohamed

Mostafa Hussian Mostafa

Remon Ehab Ramses

Taha Fawzy Elshreif

Youssef Ibrahim Hanfy

Supervisors

Dr. Amira A. Alshazly

ACKNOWLEDGMENT

We express our heartfelt gratitude to God for his unwavering support and blessings throughout this journey.

We are deeply thankful to our supervisor, Dr. Amira A. Alshazly, for her invaluable guidance, encouragement, and steadfast assistance, which have been instrumental to the success of this project.

We would also like to extend our sincere appreciation to Dr. Shaimaa Lazem, Dr. Samia Hafez, Eng. Nouran Hisham and Eng. Mohamed Hatem Abdulkader for their support, insightful feedback, and encouragement throughout the course of our work.

Our sincere thanks also go to the authors and contributors of the reference report from project [1], whose work provided valuable insights and inspiration for this project.

Finally, we extend our deepest appreciation to our families, whose unwavering support and encouragement have been the cornerstone of our achievements.

LIST OF ABBREVIATIONS

A2C	Advantage Actor-Critic	24
A3C	Asynchronous Advantage Actor-Critic	25
ACC	Accident	57
ACK	Acknowledgement	57
AI	Artificial Intelligence	15
ANOVA	Analysis Of VAriance	105
API	Application Programming Interface	4
Benchmark	Benechmark experiments of our environment	73
CAM	Correlational Attention Mechanism	27
CHK	Check (for internet)	57
CNN	Convolutional Neural Network	30
CV	Computer Vision	29
D2QN /DDQN	Double Deep Q-Network	18
D3QN /Dueling D2QN	Dueling Double Deep Q-Network	19
Data 1/Area 1	Tomart data	63
Data 2/Area 2	Mosheer Ismail	63
Data 3/Area 3	San Stefano	63
DITLCS	Dynamic and Intelligent Traffic Light Control System DITLCS	21
DNN	Deep Neural Network	10
DQN	Deep Q-Network	17
DRIVE MATE	“Name” of mobile app	57
DRL /RL	Reinforcement Learning/Deep	15
EM	Emergency Mode	22
EMR	Emergency	57
Environment Experiments	Experiments of our enviroment in different scenarios	74

EpsDQN	Epsilon-Greedy Deep Q-Network	73
FM	Fair Mode	22
FR	Functional Requirement	116
GAT	Graph Attention Networks	26
GCD	Greatest Common Divisor	66
GPS	Global Positioning System	59
Grouped	Environment with action space groubed by direction	66
GSCTSA-A3C	Graph-Structured Correlation Time-Spatial Attention – Async Actor-Critic	27
GYM	OpenAI gym	4
HighGrouped	Environment of one signal per lanes	66
HMAS	3 Hierarchical Multi-Agent Systems	20
IoT	Internet of Things	15
IOU	Intersection Over Union	30
IP	Internet Protocol	60
KL	Kullback–Leibler (Divergence)	29
LAs	Local agent	20
Literature	Literature reward that focus on waiting time	68
LR	Linear Regression	33
LSTM	Long Short-Term Memory	21
MAE	Mean Absolute Error	9
MFDQL-DTCMFDQL-DTC	Mean Field Double Q-Learning with Dynamic Timing Control	23
ML	Machine Learning	5
MPE	Mean Percentage Error	10
MQTT	Message Queuing Telemetry Transport	114
NFR	Non-Functional Requirement	117

NLP	Natural Language Processing	27
NMS	Non-Maximum Suppression	30
NN / DNN	Neural Network /Deep	10
OSM	OpenStreetMap	63
PATM	Process Activity Task Matrix	43
PM	Priority Mode	22
PPO	Proximal Policy Optimization	28
Proposed	Our proposed reward that optimizes for state space	67
QL	Queue length	57
RACS-A2C	Region-Aware Cooperative Strategy A2C	25
RAs	Regional Agents	21
Reference	Reward for reference project	67
RF	Random forest	8
RMSE	Root Mean Square Error	9
SPI	Software Process Improvement	2
SRS	Software Requirements Specification	116
SUMO	Simulation of Urban MObility	3
SumoEnv /Full	Environment with all combination in action space	66
TCP/IP	Transmission Control Protocol / Internet Protocol	57
Temporal Difference /TD	Temporal Difference	11
TraCI	Traffic Control Interface	5
TRPO	Trust Region Policy Optimization	29
XGBoost	Extreme Gradient Boosting	8
Yolo	You Only Look Once	29

DECLARATION

We hereby declare that this graduation project report is the result of our efforts, with guidance from our supervisors. Any external sources used in the work have been appropriately cited, and this document has not been submitted to any other institution for any degree or qualification.

Name	ID	Signature
Ahmed Ibrahim Hassan	20221373950	
Bichay Adel Ramses	20221440637	
Emad Mohammed Fawzy	20221461122	
Kerolss Bolis Shehata	20221374504	
Marwa Shebl Mohammed	20221448840	
Mayan Islam Mohamed	20201323258	
Mostafa Hussian Mostafa	20221449690	
Remon Ehab Ramses	20221459758	
Taha Fawzy Elshreif	20221466557	
Youssef Ibrahim Hanfy	20221454778	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ABSTRACT

Urban traffic is a significant problem that disrupts daily life, delays work and operations, and negatively impacts the environment, public health, and overall productivity.

In Egypt, traffic congestion in urban cities is a critical issue affecting various aspects of life, including commuting to work and school. Despite the government investing heavily yearly in infrastructure improvements such as roads and bridges to ease traffic, the problem persists.

Our project aims to develop an optimized solution for managing traffic by dynamically controlling traffic signals. This approach seeks to reduce vehicle waiting times, enhance traffic flow, and ensure fairness among all road users.

Unlike traditional systems that often lack responsiveness and efficiency, this approach integrates AI with IoT to create a robust and intelligent traffic management framework. The project aims to deliver a scalable, energy-efficient solution that minimizes vehicle delays, reduces energy consumption, and mitigates environmental impact. This work aspires to contribute significantly to advancing smarter, greener cities, fostering sustainable urban development, and an improved quality of life for residents.

This project presents the design and implementation of an advanced traffic flow control system tailored for smart cities. It leverages deep reinforcement learning (DRL) to address critical urban challenges such as congestion and energy inefficiency. By utilizing state-of-the-art algorithms, the system processes spatial traffic data to optimize traffic light coordination, thereby improving the efficiency of transportation networks.

The proposed solution is designed for integration within an IoT-enabled ecosystem, facilitating seamless communication between sensors, traffic controllers, and mobile devices. To enhance functionality, the system incorporates mobile application support and a streaming server infrastructure, enabling real-time traffic monitoring and adaptive management. This ensures rapid responses to dynamic traffic patterns and external factors such as weather changes or accidents.

CHAPTER 1 : INTRODUCTION

In this chapter, the urban traffic congestion problem is described, with a focus on its specific impact in Egypt. The limitations of traditional traffic systems are highlighted, and the for dynamic, AI-driven control is explained. The key contributions of the project are introduced, including a reinforcement learning-based signal optimization approach and a mechanism need for prioritizing emergency vehicles.

1.1 Project Background

The rapid increase in the number of vehicles on the road has resulted in a significant rise in traffic congestion, leading to numerous challenges for urban areas in Egypt. The smooth functioning of daily life and work routines for millions of people has been disrupted due to this congestion. As cities continue to grow and vehicle ownership increases, greater pressure is placed on the existing traffic infrastructure. Moreover, external factors such as unpredictable weather conditions add another layer of complexity to the management of traffic flow. Delays are exacerbated, road safety is reduced, and maintaining efficient transportation systems in urban environments is made even more difficult [1],[2].

1.2 Importance of the Topic

Considerable challenges are posed by traffic congestion, including negative impacts on the environment, public health, and productivity. The management of traffic effectively requires dynamic, real-time responses to address these issues. Traditional models have been found to be limited in their ability to optimize energy consumption and traffic flow, highlighting the need for innovative solutions [2].

1.3 Project Contribution

Advanced AI techniques, specifically Reinforcement Learning (RL), are utilized by our system to train a model that optimizes traffic flow by simulating traffic conditions in Alexandria's urban streets. Traffic signals are dynamically adjusted based on these simulations. Additionally, an innovative solution is introduced for registered emergency vehicles, such as ambulances and fire trucks, allowing them to log in, select a route, and send this information to the system. The traffic signals along the selected route are identified by the system and adjusted by extending green light durations and shortening red light durations to facilitate smoother passage. To validate the overall structure and functionality of the system, a small-scale prototype simulating a single intersection has been developed. The project's concept, deployment feasibility, and integration with a mobile application are tested through this prototype.

1.4 Report Organization

The structure of the next chapters is organized as follows:

Chapter 2: Scientific Background – The scientific foundations of traffic optimization are reviewed, including both traditional methods and modern advancements.

Chapter 3: Literature review– The existing literature is reviewed, various papers and studies are compared, and the findings are analyzed to identify gaps or areas requiring further exploration, thereby defining the scope of the work and necessary extensions.

Chapter 4: System Requirements and Design – In this chapter, the design of the project will be discussed in detail. The overall system flow will be explained, along with the stages of the SPI model and the software engineering methodologies that were applied.

Chapter 5: Experimental Design, Evaluation, and System Prototyping – In this chapter, the design of the experiments is presented, and their results are analyzed.

Chapter 6: Conclusion and Future Work – A summary of the project is provided, and potential directions for future work are discussed.

1.5 Conclusion

In this chapter, the background of the problem and its impact on Egypt have been discussed. Its significance has been elaborated, followed by an explanation of how the issue is addressed by our project. Finally, the structure of the subsequent chapters has been outlined. The next chapter will dive into the scientific background of algorithms used in the project.

CHAPTER 2 : SCIENTIFIC BACKGROUND

In last chapter, the issue of traffic congestion in urban areas has been introduced, emphasizing the challenges brought about by urbanization and the increasing number of vehicles.

In this Chapter, the key algorithms, solutions, and foundational concepts relevant to related work, as well as those suitable for this project, are discussed.

- Section 1: Simulation – The importance of simulation in traffic management is explained, and SUMO, a commonly used framework for traffic simulation, is introduced.
- Section 2: Machine Learning – The background of machine learning algorithms proposed in related work or considered for this project is covered. The discussion progresses from basic heuristic algorithms through unsupervised and supervised learning, concluding with reinforcement learning, including approaches such as Deep Q-Networks (DQN) and GSCTSA-A3C.
- Section 3: Computer Vision – A background on computer vision, which is utilized in the proposed solution, is provided.

2.1 Simulation

[3]

A simulation is a representation of how an existing or proposed system operates, designed to assist in decision-making by testing various scenarios or process changes. Simulations play a crucial role in performance enhancement, process optimization, safety assurance, theoretical testing, personnel training, and even entertainment through video games. By scientifically modeling systems, simulations allow users to analyze how different conditions and actions impact system performance.

Simulations are particularly valuable when the actual system is inaccessible, too dangerous to study directly, or still in its conceptual stages. A safe and cost-effective means to explore and experiment with different scenarios is provided.

The accuracy of a simulation is heavily dependent on the quality and reliability of the data used in its model construction. As a result, ongoing research focuses on improving methods for verifying and validating simulation models, particularly in computer-based environments.

Modern simulation tools leverage specialized software to create detailed visual models of processes, incorporating factors such as timing, rules, resource allocation, and constraints. For instance, a supermarket simulation might predict customer behavior during peak hours, offering insights to optimize staffing, store layouts, and inventory management.

Several simulation tools are available for addressing traffic congestion:

- **SUMO:** An open-source framework for simulating traffic networks, widely used in research. Due to its free availability, it has been selected for this project.

- **PTV Vissim:** A paid framework offering advanced features such as 3D visualization, pedestrian simulation, and integration with other PTV software.
- **Gymnasium (Gym):** An OpenAI framework that provides various basic environments for training reinforcement learning (RL) agents, with integration capabilities for SUMO, making it useful for RL-based traffic optimization experiments.

2.1.1 GYM

[4]

OpenAI Gym is an open-source Python toolkit designed for developing and comparing reinforcement learning (RL) algorithms. It provides a standardized API for communication between learning algorithms and environments, facilitating development.

Key Features:

- **Standardized API:** Gym offers a consistent interface for various environments, enabling seamless integration with RL algorithms.
- **Diverse Environments:** A wide range of environments is included, allowing for comprehensive testing and benchmarking of RL models.
- **Community Support:** As an open-source project, Gym has an active community that provides extensive resources, tutorials, and support.

Gym can be integrated with traffic simulation tools like SUMO to create environments for training RL agents in traffic management scenarios. This integration allows researchers to develop and test algorithms for optimizing traffic flow, signal control, and other transportation-related tasks within a simulated environment.

2.1.2 Simulation of Urban Mobility (SUMO)

[5]

Simulation of Urban Mobility (SUMO) is an open-source traffic simulation platform designed to model, analyze, and optimize urban mobility within complex road networks. Its microscopic simulation approach enables a highly detailed representation of individual vehicle behaviors, including acceleration, deceleration, lane changes, and route selection. The platform supports various vehicle types—cars, buses, bicycles, and pedestrians—and their interactions with infrastructure components such as traffic signals, pedestrian crossings, and road closures.

One of SUMO's most notable features is its flexibility in integrating with external tools and algorithms, particularly in the domains of artificial intelligence (AI) and machine learning. For instance, SUMO can be combined with reinforcement learning algorithms to design and evaluate adaptive traffic control strategies. This capability transforms SUMO into a dynamic training environment for AI models aimed at optimizing real-time traffic flows, reducing congestion, and minimizing travel times by simulating realistic traffic dynamics.

Key Features of SUMO include.

- Extensive Data Output Capabilities: SUMO facilitates detailed analysis and visualization by generating metrics such as vehicle travel times, queue lengths, and fuel consumption. These outputs provide valuable insights for identifying traffic patterns, inefficiencies, and potential solutions.

- Real-Time Interaction via TraCI: SUMO provides the Traffic Control Interface (TraCI), allowing users to interact with the simulation in real-time using programming languages like Python. Traci enables external applications to:
    - Retrieve live simulation data, such as vehicle positions, speeds, and waiting times.
    - Dynamically modify simulation parameters, including traffic signal phases, vehicle routing, and lane speed limits.

- TraCI's API includes commands for real-time simulation control and customization, offering flexibility to test innovative traffic management strategies tailored to specific urban challenges.

- By integrating advanced simulation features with AI-driven tools, SUMO provides a versatile platform for researchers and practitioners to develop smarter, more efficient traffic management systems.

2.2 Machine Learning

[6]

Machine learning (ML) is recognized as a subset of artificial intelligence (AI), focusing on the creation of systems capable of learning from data and making decisions or predictions. Rather than being explicitly programmed for every task, ML models are trained to learn patterns and rules from data. It is defined as the field of study that enables computers to learn without explicit programming. As implied by its name, it allows computers to acquire learning capabilities, making them more human-like.

2.2.1 Heuristic algorithms

Heuristic algorithms are designed to efficiently address complex optimization problems by providing approximate solutions when exact methods are impractical.

This approach is particularly useful in scenarios where traditional algorithms are computationally prohibitive. The handling of uncertainty, incomplete information, and large-scale optimization tasks is facilitated by heuristics, making them widely utilized.

Heuristic algorithms are categorized into two approaches based on the complexity of the problem and solution: Heuristics and Metaheuristics.

Heuristics

Heuristic problem-solving techniques are employed to find satisfactory or "good enough" solutions within a reasonable time frame when computing exact solutions is too costly or time-consuming. Simple rules of thumb, educated guesses, or intuitive judgments are typically utilized by these algorithms, aiding in efficient navigation through complex problem spaces.

[6]

Metaheuristics

Metaheuristics are defined as higher-level procedures or strategies that guide heuristic algorithms in solving complex optimization problems. These methods are flexible and general-purpose, allowing adaptation to various fields such as engineering, AI, and economics.

[7]

There exist many algorithms of heuristics algorithms, genetic algorithms, Simulated Annealing Algorithms, and Local Search Algorithms (Hill-Climbing), the most common one algorithm is genetic algorithms.

Genetic algorithms

These algorithms are inspired by the process of natural evolution. Genetic algorithms use principles such as selection, crossover (recombination), mutation, and survival of the fittest are utilized to iteratively search for optimal solutions in a problem space.

[7]

2.2.2 Supervised Learning

Supervised learning is recognized as a foundational approach in machine learning, in which models are trained on labeled datasets. This method is applied to predict outcomes based on input features and is widely used in various domains, such as spam detection, image classification, and medical diagnosis.[3]

2.2.2.1 Decision Tree

[8].[9]

A decision tree is widely utilized as a machine learning model that employs a tree-like structure to represent decisions and their possible consequences, including outcomes, resource costs, and utility. Each internal node is used to test an attribute, each branch represents the outcome of the test, and each leaf node contains a class label. This structure facilitates classification and regression tasks and is easily interpretable.

Entropy:

Entropy is regarded as a key concept in information theory, introduced by Claude Shannon. It measures the amount of randomness or impurity present in a dataset.

$$H(S) = -\sum_{i=1}^{n} P_i(P_i)$$

Equation 2.1 Entropy

Where:

Pi is the probability of target class i given a feature in the data

During decision tree splitting, the maximization of purity is aimed at ensuring effective classification. Entropy is utilized to quantify the uncertainty of a random variable and to determine how well a feature splits the data.

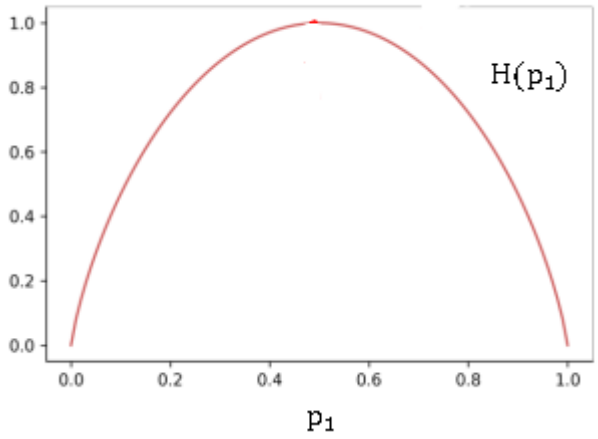


Figure 2.1 Visualizing Entropy Function

And so, the decision to choose a feature with maximum information gain

$$H(P_1^{root}) - (w^{left} H(P_1^{left}) + w^{right} H(P_1^{right}))$$

Equation 2.2 Information gain equation

Where H is the entropy function

w is a fraction of instances with a specific feature value

The regression tree A regression tree is recognized as a type of decision tree that is applied to continuous target variables. Instead of splitting based on entropy, variance reduction is utilized as the criterion, and the mean value of the target variable for each group is predicted.

However, decision trees are prone to overfitting, necessitating the use of more robust methods. Common techniques employed to address this include tree ensembles, sampling techniques, and cumulative loss, as applied in XGBoost.

However, decision trees tend to overfit, so more robust methods are required. Common approaches to address this include tree ensembles, sampling techniques, and cumulative loss (as used in XGBoost).

Tree Ensembles

Tree ensembles are utilized to enhance robustness and reduce error by combining multiple decision trees. Rather than relying on a single tree, predictions are aggregated across many trees, thereby improving stability and accuracy.

Sampling in Tree Ensembles

Sampling techniques are incorporated into tree ensembles to enhance computational efficiency and minimize error. Instead of working with the entire dataset, subsets of data are used. The algorithms that implement these techniques include Random Forest and XGBoost, which are discussed in the subsequent sections.

2.2.2.1.1 Random Forest

[8],[9]

Random Forest (RF) is identified as a variation of the decision tree algorithm that employs an ensemble of trees. The method involves:

- **Sampling:** Each tree is trained on a random subset of the data.
- **Bagging:** The final decision is determined by aggregating the outputs of individual trees, utilizing majority voting for classification or averaging for regression.

Bagging: Bagging is recognized as a technique used in tree ensembles. The predictions of individual trees are combined by:

- Taking the majority vote for classification tasks.
- Calculating the mean for regression tasks.

2.2.2.1.2 XGBoost

[8],[9]

XGBoost (Extreme Gradient Boosting) is acknowledged as an advanced variation of the decision tree algorithm. The method involves:

- The utilization of tree ensembles with sampling.
- **Boosting:** Instead of merely aggregating outputs, errors from previous trees are optimized to enhance performance.

XGBoost improves the gradient boosting algorithm, providing enhanced speed and efficiency. Due to its robustness, it is extensively applied in competitions and real-world implementations.

Boosting

Boosting is employed to improve ensemble learning through the sequential training of trees. Unlike bagging, predictions are not simply aggregated (e.g., by majority vote or mean). Instead, adjustments are made for errors encountered by previous trees, optimizing the overall loss function.

Regularization is incorporated within XGBoost to mitigate overfitting, ensuring better generalization to unseen data and increasing reliability in predicting traffic delays under varying conditions.

Parallel processing is supported by XGBoost, expediting training and making it well-suited for large-scale simulations and extensive datasets.

Tree pruning is employed in XGBoost to reduce model complexity and enhance interpretability.

To assess the performance of regression trees, evaluation metrics commonly utilized include:

Mean absolute error (MAE):

$$MAE = \frac{1}{n}\sum_{1\cdot}^{n} \quad |y_i - \hat{y}_i|$$

Equation 2.3 Mean absolute error

R squared:

$$R^2 = 1 - \frac{\sum_1^n \quad (y_i - \hat{y}_i)^2}{\sum_1 \quad (y_i - \underline{y})^2}$$

Equation 2.4 R-squared

Root Mean Square error (RMSE):

$$RMSE = \sqrt{\frac{1}{n}\sum_{1\cdot}^{n} (y - \hat{y}_i)^2}$$

Equation 2.5 Root Mean Square Error

Mean Percentage Error (MPE):

$$MPE = \frac{1}{n}\sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i} \times 100$$

Equation 2.6 Mean Percentage Error

2.2.2.2 Deep Neural Network (DNN)

[8]

A Deep Neural Network (DNN) is recognized as an AI algorithm designed to mimic the human brain. It is widely applied in fields such as speech recognition and image processing.

DNNs consist of neurons, where each neuron functions similarly to a regression/logistic regression model with an activation function.

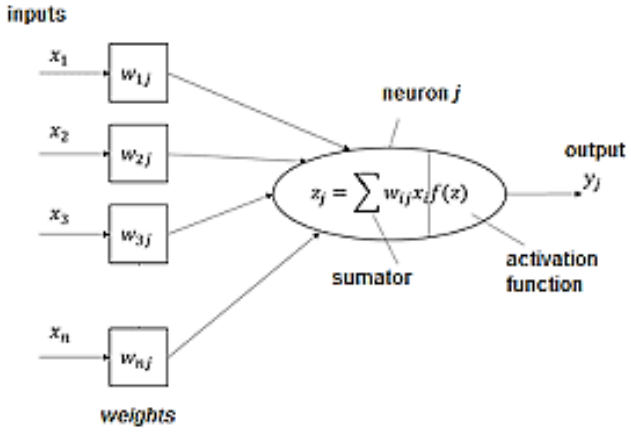


Figure 2.2 Single Neuron

[11]

Using many of these enables the NN to be able to learn more about the data making it a powerful algorithm.

Layers:

- **Input Layer:** Receives raw data.

- **Hidden Layers:** Intermediate layers where computations occur; multiple hidden layers contribute to the "deep" aspect of DNNs.
- **Output Layer:** Produces the final prediction or classification.

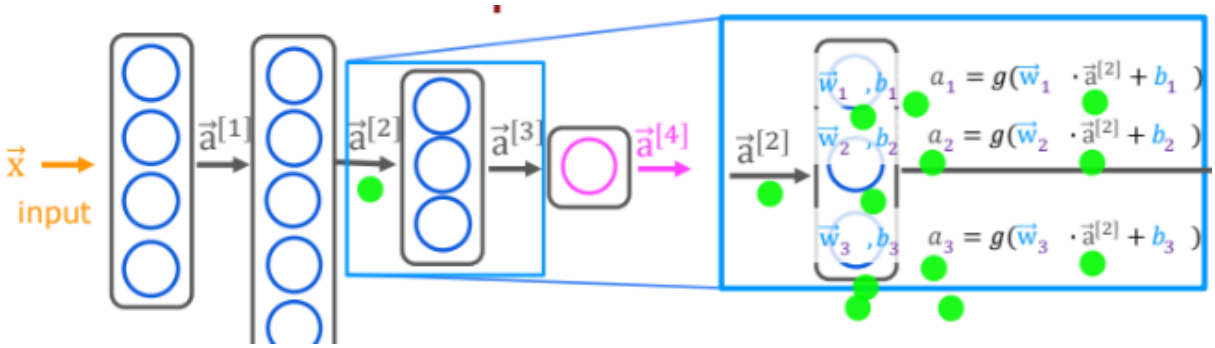


Figure 2.3 NN with 4 layers

2.2.3 Reinforcement Learning

Reinforcement learning (RL) is a machine learning methodology in which decisions are learned by an agent through interaction with its environment via trial and error. Feedback is provided in the form of rewards or penalties, guiding the agent to optimize its behavior. This approach is modeled after how humans learn from experiences to improve actions and achieve desired outcomes. The primary goal of a reinforcement learning agent is to maximize its expected cumulative reward, often referred to as the expected return. Although both supervised learning and reinforcement learning involve mapping inputs to outputs, their feedback mechanisms differ significantly. In supervised learning, feedback is provided as correct actions or labels, whereas in reinforcement learning, feedback comes in the form of rewards and punishments, encouraging positive behavior and discouraging undesirable actions.

This introduction explains the main concepts of reinforcement learning (RL), beginning with the key components of any RL model. It is followed by the discount rate, which is used to calculate rewards more effectively, and the ways in which data is collected by the agent (online and offline). Next, the types of algorithms, including policy-based and value-based methods, are discussed. The section also explores soft updates, and the amount of data collected, with a focus on Temporal Difference (TD) and Monte Carlo methods. Finally, the Bellman function is introduced.

Key Components of Reinforcement Learning [11]:

- Agent: The decision-maker who learns by interacting with the environment, taking action, and receiving feedback.

- Environment: The external system in which the agent operates, defined by specific rules and conditions.
- State: The representation of the environment at a given moment. For instance, in a video game, the state could be a single frame, while in a trading system, it could be the current value of a stock.
- Action: A decision made by the agent in each state. The set of all possible actions is referred to as the action space.
- Reward: The feedback signals the agent receives from the environment after acting. Rewards quantify the desirability of the agent's actions, with the goal being to maximize cumulative rewards over time.

  The reward is fundamental in RL because it's the only feedback for the agent. Thanks to it, our agent knows if the action taken was good or not.

$$R(\tau) = \sum_{k=0}^{\infty} (r_{t+k+1})$$

Equation 2.7 Basic cumulative reward in RL

Cumulative rewards are the total rewards accumulated by the agent over time.

However, in practice, we can't just add them directly, we add a factor that determines how much agents consider long-term rewards.

The discount rate(γ) determines the importance of future rewards:

Rewards received earlier (at the start of the game) are more likely to occur as they are more predictable than long-term future rewards.

- A higher γ means the agent considers long-term rewards more heavily.
- A lower γ means the agent focuses on short-term rewards [11].

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots$$

=

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Equation 2.8 Cumulative reward with discount factor

Where:

τ is trajectory (sequence of state/action)

γ is the discount factor

The Overview of the RL process:

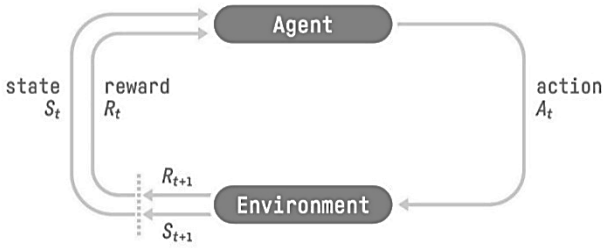


Figure 2.4 Overview of the RL process

Reinforcement learning algorithms can collect data from the environment using two approaches: online or offline.

In online reinforcement learning, the agent gathers data directly: it collects a batch of experiences by interacting with the environment. Then, it uses this experience immediately (or via some replay buffer) to learn from it (update its policy).

On the other hand, in **offline reinforcement learning**, the agent only uses data collected from other agents or human demonstrations. It does not interact with the environment [13],[14].

The approaches can also be viewed from another perspective: in offline training, the model is trained without real-time interaction with the environment, while online training involves collecting data after training and using it to directly improve the model from the environment.

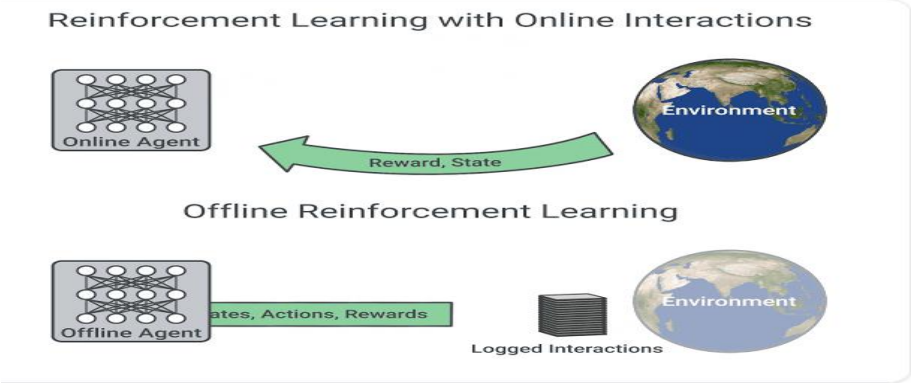


Figure 2.5 Online and Offline RL interaction

Reinforcement learning algorithms can be trained using two approaches: one involves training to predict cumulative rewards and selecting actions with the highest reward (value-based), while the other predicts actions directly.

Policy-Based Methods [11]

Policy-based methods focus on learning a **policy function** that directly maps states to optimal actions.

- **Deterministic Policies**: The agent consistently chooses the same action for a given state.
- **Stochastic Policies**: The policy generates a probability distribution over possible actions for each state, allowing for action variability.

Value-Based Methods

Value-based methods aim to learn a **value function** that estimates the expected return for being in a state or performing an action in that state.

- **State Value Function (V(s))**: Represents the expected cumulative reward starting from state s and following policy π:

$$V^{\pi}(s) = E_{\Pi}[\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} \,|s_t = s]$$

Equation 2.9 State Value Function

- **Action Value Function (Q(s, a))**: Represents the expected cumulative reward starting from states, taking action a, and then following policy π:

$$Q^{\pi}(s,a) = E_{\Pi}[\sum_{t=0}^{\infty} \gamma^{t} r_{t+1} \,|s_t = s, a_t = a]$$

Equation 2.10 Action Value Function

The frequency with which an RL algorithm collects data from the environment can be either over the entire episode or just a temporal part of it. First, the term "episode" is explained, followed by soft update rule, which is used in exploring laws, then explanation of Monte Carlo and Temporal Difference methods.

An **episode** refers to a complete cycle of interaction, starting from an initial state, progressing through a series of actions and rewards, and ending in a final state. For example, an episode could represent one game or an attempt to achieve a specific goal.

Soft Update [11]

In reinforcement learning, **soft updates** gradually adjust the parameters of a target network, commonly used in algorithms like Deep Q-learning.

Formula:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha X$$

Alternatively:

$$Q(s,a) = Q(s,a) + \alpha(X - Q(s,a))$$

Equation 2.11 Soft update rule

- This gradual adjustment improves stability in training compared to hard updates, which copy parameters directly.

Temporal Difference (TD) and Monte Carlo Methods [11]

- **Monte Carlo Methods**:
    - Learn from entire episodes of experience before updating.
    - Use the total return (Gt) at the end of an episode to update the value function.
    - Formula:

$$V(s_t) = V(s_t) + \alpha[G_t - V(s_t)]$$

Equation 2.12 Monte Carlo update rule

Where:
$V(s_t)$: value of state
α: learning rate
G_t: return from episode

- **Temporal Difference (TD) Methods**:
    - Learn incrementally after each step.
    - Update the value function based on the TD target.
    - Formula:

$$V(s_t) = V(s_t) + \alpha[R_{t+1} + \gamma V(s_{t+1}) - (V(s_t)]$$

Equation 2.13 Temporal Difference update rule

- Where:
  $[R_{t+1} + \gamma V(s_{t+1})]$ is called TD target

The full update rule in Q-learning algorithms is represented by the Bellman equation, which first selects an action and then follows the optimal behavior.

Bellman Equation:

[11]

The Bellman equation is a fundamental equation in reinforcement learning that helps an agent figure out the best way to maximize its total rewards over time.

It's not a new thing, it's a simplified way of saying if an agent takes an action, it is rewarded immediately (based on R(s, a)), and then it expects to behave optimally later to maximize the total reward, which is why the future rewards are considered through the MAX a′ V(s′) term.

In other words, the Bellman equation in RL does suggest that an agent should take an action in the current state that will maximize its future expected rewards, weighted by γ. The agent considers both the immediate reward and the potential future rewards (looking ahead and behaving optimally)

$$Q(s,a) = R(s) + \gamma\ max\ (Q(s',\ a')$$

Equation 2.14 Bellman equation

[16] , [9]

2.2.3.2 QN Algorithms

This section explains the first RL algorithm, Q-learning (QNetwork), along with its variations, most of which are used in the related work.

2.2.3.2.1 Q Network (QN)

[16].[11]

Q-learning (Q-Network /QN) is a fundamental form of learning that serves as the foundation for more advanced techniques. It has been widely used in both research and industry applications. Q-learning is a reinforcement learning method that enables agents to learn optimal actions through temporal difference (TD) learning. In this process, the agent takes an action in a given state, receives a reward or penalty, and updates its understanding of that state based on the feedback received.

Algorithm Steps:

- **Initialize the Q-table**: Set up a table with Q-values for each state-action pair. Initially, these values can be set to zero or small random values.
- **Choose an action using the ε-greedy strategy**: With probability ε, choose a random action to explore different possibilities (exploration). Otherwise, choose the action with the highest Q-value for the current state (exploitation).
- **Act** at **get reward** Rt+ **and observe next state** St+1: The agent performs the chosen action in the environment, receives a reward, and transitions to a new state.

- **Update** Q(St, At) Use the Q-learning update rule to adjust the Q-value for the current state-action pair based on the received reward and the maximum expected future rewards.

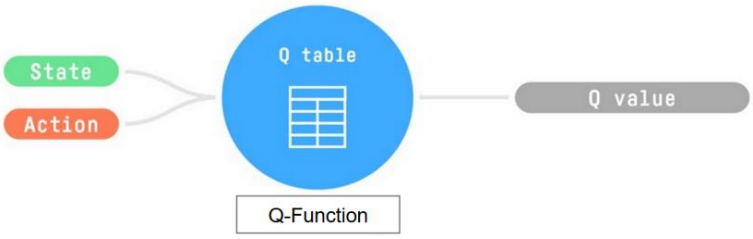


Figure 2.6 Q function

The limitation Q-Learning is a **tabular method**, meaning it stores Q-values for each state-action pair in a table. This becomes a significant limitation when the state or action spaces are **large**, requiring an enormous table size, or when the spaces are **continuous**, making it infeasible to represent them using arrays or tables.

To overcome Q-Learning's scalability issues, **Deep Q-Learning** replaces the table with a deep neural network to approximate Q-values, explained in the next section.

2.2.3.2.2 Deep Q Network (DQN)

[17]

To address Q-Learning's scalability issues, Deep Q-Learning (DQN) replaces the table with a deep neural network to approximate Q-values.

Before explaining this, it's important to introduce a key concept: **Experience Replay**. Experience Replay improves the efficiency of learning by creating a Replay Buffer, which stores experience tuples collected during interactions with the environment. Later, small random batches of these tuples are sampled from the buffer for training. This approach prevents the network from focusing solely on recent actions and outcomes, ensuring a more stable and comprehensive learning process. It enables the efficient utilization of experiences, helps prevent forgetting, and reduces correlation (mitigating catastrophic forgetting).

The training algorithm for DQN involves two main phases:

- **Sampling**:
- The agent interacts with the environment by taking actions and stores the observed experiences as tuples in the replay memory. Each tuple typically includes:
    - s: Current state
    - a: Action taken
    - r: Reward received
    - s′: Next state
- **Training**:
    - A small, random batch of experience tuples is selected from replay memory.

- The agent learns from this batch by performing a gradient descent update step to minimize the Q-value prediction error.

In some cases, the input may proceed, for example when the input is Images or Video frames.

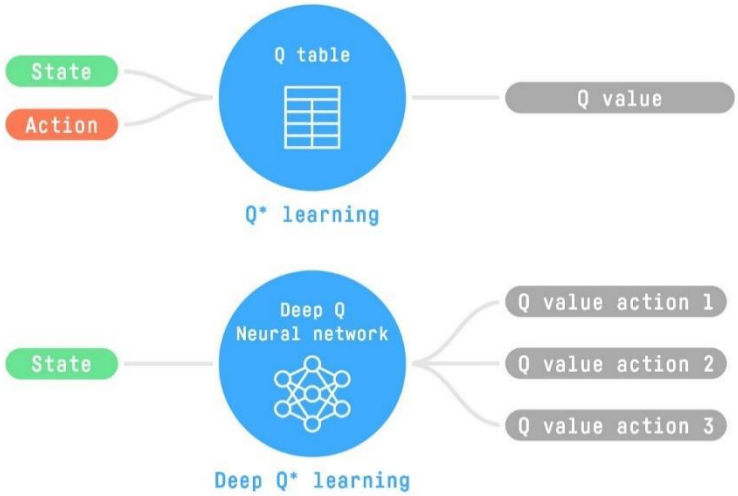


Figure 2.7 QN and DQN architecture overview

2.2.3.2.3 Double Deep Q Network (D2QN)

[11]

When calculating the TD error (also known as the loss), we compute the difference between the TD target (Q-Target) and the current Q-value (the estimation of Q).

However, the challenge arises because the same parameters (weights) are used for estimating both the TD target and the Q-value. As a result, there is a significant correlation between the TD target and the parameters we are adjusting. This means that, at each training step, both the Q-values and the target values shift. While we are getting closer to the target, the target itself is also moving. It's like chasing a moving target! This dynamic can lead to significant oscillations during training.

To illustrate this with an analogy: imagine you are a cowboy (representing the Q-value estimation) trying to catch a cow (representing the Q-target). Your goal is to get closer to the cow (reduce the error). However, at every time step, as you try to approach the cow, it keeps moving (because you are using the same parameters for both the Q-value and the TD target). This constant movement of the cow makes it difficult to catch, causing instability in the training process.

The solution to this issue is to decouple the action selection from the target Q-value generation by using two separate networks.

Specifically, we:

- Use our **DQN network** to select the best action to take for the next state (the action with the highest Q-value).

- Use our **Target network** to calculate the target Q-value for taking that action at the next state.

By using two networks, we prevent the target from constantly shifting with the Q-value estimates, thereby reducing the oscillation and instability in training. This helps stabilize the learning process.

Advantages of the Algorithm:

- Reduced Bias: Minimizes Q-value estimation errors for better decision-making.

- Stability: Target networks and experience replay ensure stable training.

- Performance: Excels in complex environments due to its robust architecture.

Limitations of the Algorithm:

- Implementation Complexity: Requires significant computational resources and expertise.

- Dependence on Reward Structure: Effectiveness heavily relies on the quality of the reward design.

- Hyperparameter Sensitivity: The DDQN algorithm's performance and stability are highly sensitive to changes in key hyperparameters,

2.2.3.2.4 Dual DDQN (D3QN)

[18],[19]

This algorithm is like DDQN, but with the addition of the Advantage function to improve stability.

Advantage function:

We can stabilize learning further by using the Advantage function

The idea is that the Advantage function calculates the relative advantage of an action compared to the others possible at a state: how taking that action at a state is better compared to the average value of the state.

- **If A(s, a) > 0**: our gradient is pushed in that direction.
- **If A(s, a) < 0**: (our action does worse than the average value of that state), our gradient is pushed in the opposite direction.

$$A(s,a) = Q(s,a) - V(s)$$

Equation 2.15 Advantage function

Where:

V: is the average value of the state

The idea of the D3QN algorithm:

Dueling Architecture: In dueling DQN, instead of directly calculating the difference with the advantage function, we **separate** the two components:

1. **State value (V(s))**: This tells you how good a state is.
2. **Advantage function (A(s, a))**: This tells you how much better a specific action is compared to others in that state.

Advantages of the algorithm:

- **Better Policy Stability**: Reduces overestimation bias, resulting in more stable and reliable policies.
- **Improved Learning Efficiency**: Achieves faster convergence and better sample efficiency, reducing the need for extensive training.
- **Enhanced Decision-Making**: Excels in environments with many similar states, where certain actions are more critical to optimal performance many similar states where certain actions are more critical.

Disadvantages of the algorithm:

- **Higher Computational Requirements**: The added complexity of the architecture leads to increased computational demands and requires significantly more resources due to the additional streams, approximately 2.5 to 3 times that of DQN.
- **Complexity in Training**: The dual-stream architecture and combination of Double Q-Learning and Dueling Network require more advanced techniques and longer training times.
- **Not Ideal for High-Dimensional State Spaces**: The algorithm may struggle with environments that have extremely high-dimensional state spaces.
- **Requires Extensive Tuning**: Finding the optimal hyperparameters (e.g., learning rate, exploration rate) is time-consuming but crucial for achieving good performance.

2.2.3.3 Hierarchical Multi-Agent Systems (HMAS)

[2], [20]

This algorithm exemplifies how hierarchical and predictive systems can improve urban traffic management; the algorithm consists of **two levels of control**:

- **Local Agents (LAs)**:
    - Each intersection has its local agent managing traffic lights.

- Uses **Reinforcement Learning** to learn how to reduce traffic delays based on past experiences.

- **Regional Agents (RAs)**:
  - Each region (a group of intersections) has a regional agent overseeing traffic.
  - Uses **LSTM** to predict future traffic conditions based on data from all the local agents.

Communication Between Levels:

- Local agents send real-time traffic data (e.g., queue lengths, delays) to the regional agent.
- Regional agents decide when coordination is needed and guide local agents in managing traffic.

Reinforcement Learning (RL): Helps local agents adapt to traffic conditions at their intersections.

LSTM Neural Networks: Allow regional agents to predict traffic jams before they happen.

Two Modes of Operation:

- Independent Mode: Manages the intersection autonomously.
- Cooperative Mode: Coordinates with the Regional Controller when there's heavy traffic.

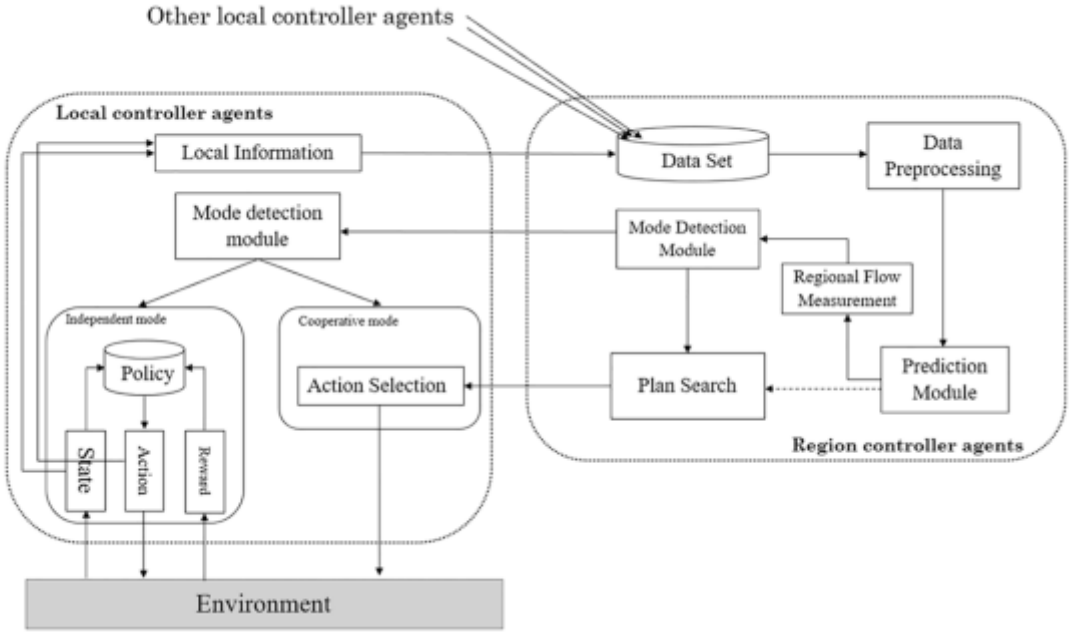


Figure 2.9 HMAS architecture

2.2.3.4 Dynamic and Intelligent Traffic Light Control System DITLCS

[21]

The Dynamic and Intelligent Traffic Light Control System (DITLCS) is a proposed algorithm designed to enhance urban traffic management through a combination of deep reinforcement learning and fuzzy logic.

Modes of Operation:

- **Fair Mode (FM)**: Treats all vehicles equally, aiming to minimize the overall average waiting time.
- **Priority Mode (PM)**: Assigns different priorities to vehicles, reducing waiting times for higher-priority vehicles (e.g., buses).
- **Emergency Mode (EM)**: Gives the highest priority to emergency vehicles like ambulances and fire trucks.

Algorithm Workflow:

Mode Selection:

The fuzzy inference system (if condition-fuzzy logic) selects the most appropriate mode based on traffic conditions.

For example, if emergency vehicles are detected, EM is activated.

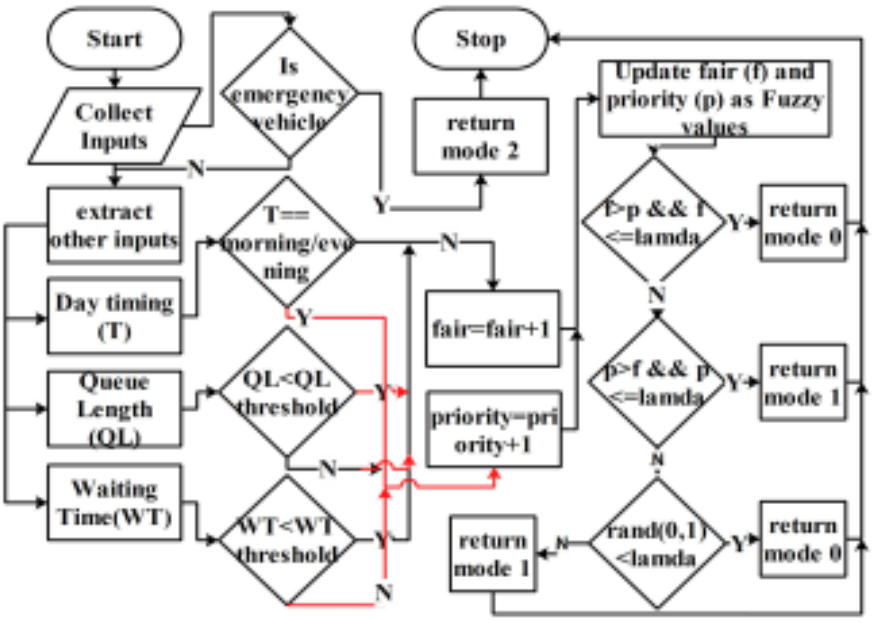


Figure 2.10 DITLCS fuzzy logic

Action Selection: The DRL agent analyzes traffic states to decide whether to change the traffic light phase or extend the current phase duration.

Phase Adjustment: Based on the DRL output, signal durations are dynamically adjusted for each traffic direction to optimize flow.

Advantages of the algorithm:

- Efficiency: Minimizes average waiting times, queue lengths, and delays.
- Scalability: Adapts to heterogeneous traffic, including priority and emergency vehicles.
- Environmental Benefits: Reduces fuel consumption and carbon emissions by decreasing idling times.

2.2.3.5 Mean Field Double Q-Learning with Dynamic Timing Control (MFDQL-DTC)

[22]

The MFDQL-DTC algorithm (Mean Field Double Q-Learning with Dynamic Timing Control) is an algorithm for dynamic traffic signal control that integrates reinforcement learning and other AI for traffic optimization.

The algorithm steps:

For each episode:

1. Start with an initial observation of the local intersection.
2. Predict the traffic flow βt for the current timestep using the RNN.
3. Compute the optimal phase duration Dt dynamically using a revised Webster's formula.

   The **Webster's Formula** calculates how long the green light should stay on.

   Example:

   - If lots of cars are coming: Green light lasts longer.
   - If fewer cars are coming: The green light lasts shorter.

4. Each agent (uses the DQN algorithm) selects an action (a green phase from a predefined set) based on the "Boltzmann" SoftMax policy.
5. Execute the selected actions for the computed duration Dt.
6. Observe the rewards and next state observations.

7. Calculate the mean action of neighboring agents **(Mean Field)**, The **mean field** is the **average effect** that a group of many agents (or things) has on a single agent.
8. Instead of calculating the influence of each agent individually, we take their **average behavior** use that as a simplified representation, and then store transitions in replay buffers.

Then update values of DQN.

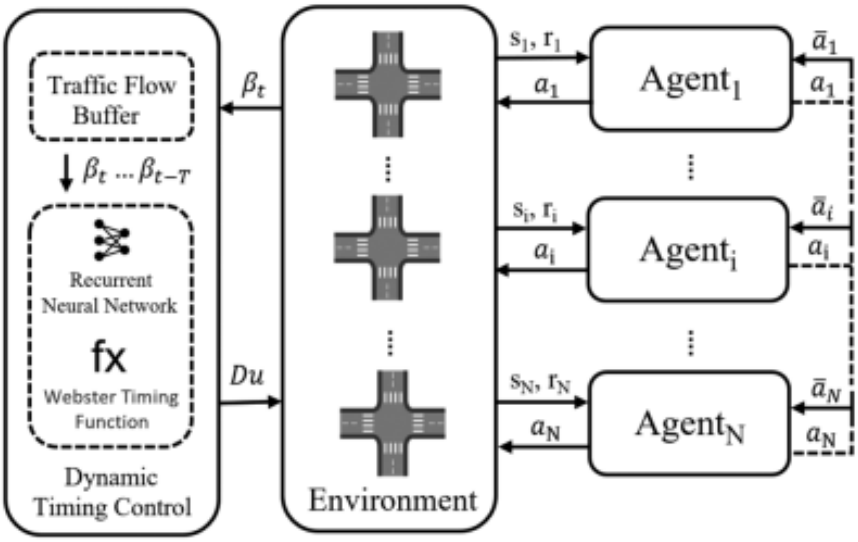


Figure 2.11 MFDQL-DTC architecture

2.2.3.6 Actor-Critic Algorithm and Variation

2.2.3.6.1 Advantage Actor-Critic (A2C)

In Policy-Based methods, we aim to optimize the policy directly without using a value function. More precisely, Reinforce is part of a subclass of *Policy-Based Methods* called *Policy-Gradient methods*. This subclass optimizes the policy directly by estimating the weights of the optimal policy using Gradient Ascent [11].

Given the stochasticity of the environment (random events during an episode) and stochasticity of the policy, trajectories can lead to different returns, which can lead to high variance. Consequently, the same starting state can lead to very different returns. Because of this, the return starting at the same state can vary significantly across episodes [23].

The solution is to mitigate the variance by using a large number of trajectories, hoping that the variance introduced in any one trajectory will be reduced in aggregate and provide a “true” estimation of the return [23].

Actor-critic methods, a hybrid architecture combining value-based and Policy-Based methods that helps to stabilize the training by reducing the variance using:

- *An Actor* that controls how our agent behaves (Policy-Based method)

- *A Critic* that measures how good the taken action is (Value-Based method)

The details of algorithms:

At each timestep, t, we get the current state St from the environment and pass it as input through our Actor and Critic, The Policy takes the state and outputs an action At, Then the critic takes that action also as input and, using St and At, computes the value of taking that action at that state: the Q-value**.**

the model then executes the action then updates the parameters of the actor and critic [18].

2.2.4.6.2 Advantage Actor-Critic (A2C)

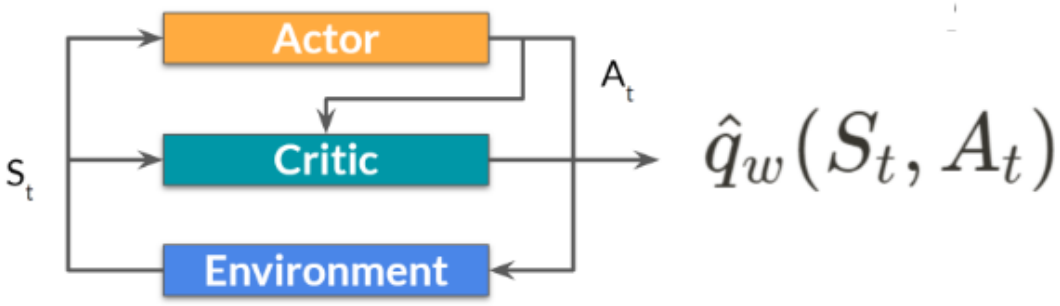


Figure 2.12 Actor-Critic Architecture

It's based on A2C with Advantage function as discussed in section 2.2.4.2.3

Combining the Actor critic with the advantage function results in better algorithm called Advantage Actor critic (A2C) [18].

2.2.3.6.2 Async Advantage Actor critic (A3C)

[24]

The A3C algorithm is an asynchronous version of the advantage actor-critic (A2C) algorithm.

A key innovation of A3C is the use of multiple worker agents running in parallel. Each worker interacts with its own environment and performs updates to the global model asynchronously. This reduces the correlation between training data and helps prevent overfitting.

A3C includes an entropy term in the objective function to encourage exploration and prevent the policy from becoming too deterministic too early in training.

2.2.3.7 Region-Aware Cooperative Strategy A2C (RACS-A2C)

[24]

The proposed Region-Aware Cooperative Strategy (RACS) based on the Advantage Actor-Critic (A2C) algorithm is a decentralized approach designed to improve adaptive traffic signal control in multi-intersection scenarios.

The paper uses an A2C algorithm with an extension Region-Aware Cooperative Strategy (RACS).

Region-Aware Cooperative Strategy (RACS):

To operate within a decentralized framework, where each intersection is controlled by a local Reinforcement Learning (RL) agent and **captures** spatial correlations between different intersections by assigning weights to nearby nodes (intersections): The paper introduced **Region-Aware Cooperative Strategy (RACS)**, this is implemented by using **Graph Attention Networks (GAT).**

The algorithm calculates the weighted neighbor features between neighbors of the intersection

$$f_{ui} = LeakyReLU(b^T.[g_u;g_i'])$$

Equation 2.16 correlation score between nodes in RACS

Where:

f_{ui}: correlation score between node u and I

b: Learnable weight vector

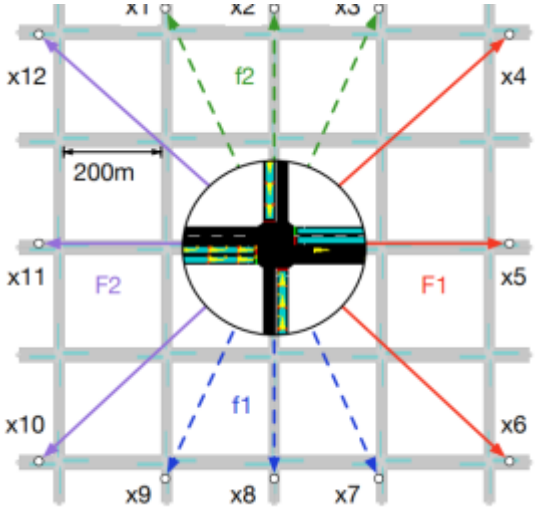
$$b_{u_i} = \frac{exp\ exp\ (f_{u_i})}{\sum_{L=1} \quad exp\ (f_{uL})}$$

Equation 2.17 SoftMax normalize of weight

$$g'_u = \frac{1}{u-1} \sum_{g' \in cw} \quad b_{ji} \times : g'_j$$

Equation 2.18 Average Weight of Intersection in RACS

Where:
gu: Final updated feature for intersection u
b_{ui} : Weight showing how many neighbors i influence u
g'_i : Neighbor i's feature



Equation 2.19 Correlation with a neighbor in RACS

2.2.3.8 Graph-Structured Correlation Time-Spatial Attention – Async Actor-Critic (GSCTSA-A3C)

The algorithm is a combination of RL with other algorithms aimed at measuring the traffic flow in more than one view trying to capture features.

The algorithm is a combination of:

1- Graph Attention Networks (GAT): The paper used GAT, allowing it to measure **Spatial** info, inspired by paper [24].

2- Correlational Attention Mechanism (CAM): Dynamically prioritizes features (e.g., traffic volume, weather) based on their relevance at different times [2].

This is inspired by the attention mechanism widely used in deep learning, especially in natural language processing (NLP)

1- Long Short-Term Memory (LSTM): Models the temporal dependencies in traffic patterns

Output = LSTM(current traffic ,previous memory)

Equation 2.20 LSTM for temporal pattern analysis

2- Asynchronous Advantage Actor-Critic (A3C) is a reinforcement learning algorithm, a variation of A2C developed by OpenMind to learn policies in a decentralized architecture.

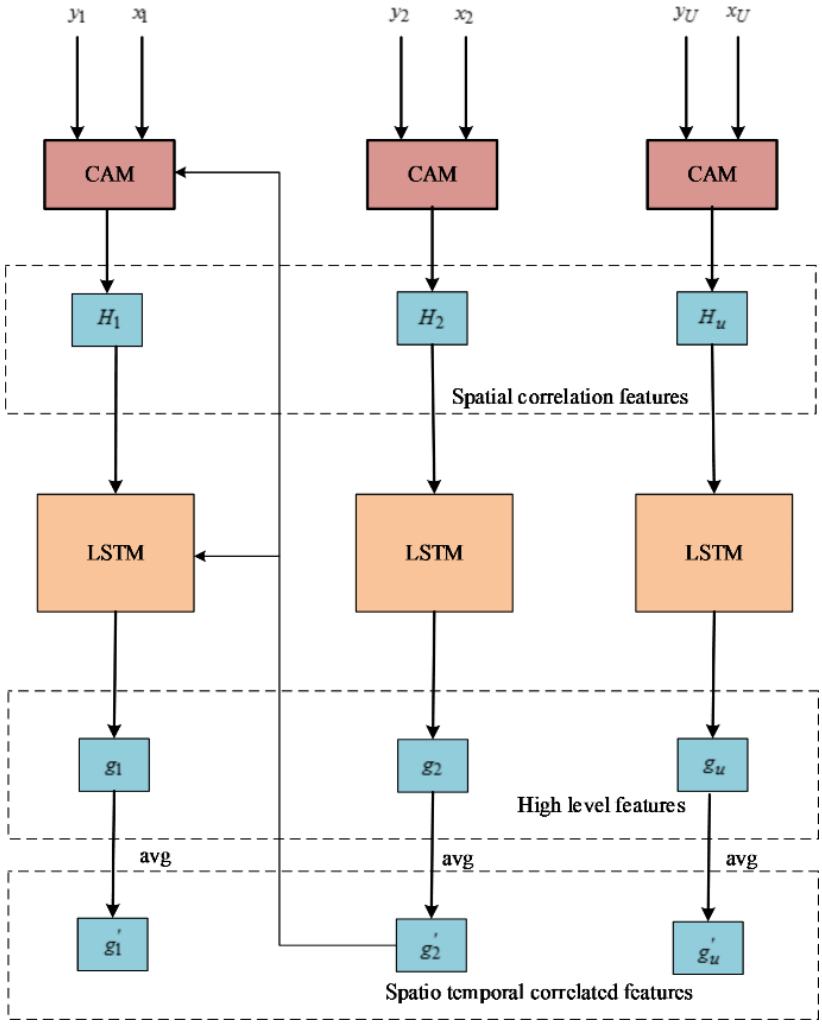


Figure 2.13 GSCTSA-A3C architecture

The weight b in this algorithm is usually normalized (by softmax to make all weights sure all sum to 1)

Limitations of this Algorithm [2]:

- **Scalability and Complexity**: Due to the decentralized nature of A3C, scaling the model to large, real-world environments may introduce significant complexity.

2.2.3.9 Proximal Policy Optimization (PPO)

[25]

The idea with Proximal Policy Optimization (PPO) is that we want to improve the training stability of the policy by limiting the change you make to the policy at each training epoch: we want to avoid having too large of a policy update.

. To do so, we need to measure how much the current policy changed compared to the former one using a ratio calculation between the current and former policy. We clip this ratio in a range$[1-\epsilon,1+\epsilon]$ meaning that we remove the incentive for the current policy to go too far from the old one (hence the proximal policy term).

At first, the algorithm calculates the ratio between the old policy and the updated policy (This ratio can replace the log probability in the policy objective function)

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$$

Equation 2.21 ratio between old policy and updated policy in PPO

- If $r_t(\theta)>1$, the action *at*, at state *st* is more likely in the current policy than the old policy.
- If $r_t(\theta)$ is between 0 and 1, the action is less likely for the current policy than for the old one.

So, this probability ratio is an easy way to estimate the divergence between old and current policy

However, without a constraint, if the action taken is much more probable in our current policy than in our former, this would lead to a significant policy gradient step and, therefore, an excessive policy update.

Consequently, we need to constrain this objective function by penalizing changes that lead to a ratio far away from 1 (in the paper, the ratio can only vary from 0.8 to 1.2).

By clipping the ratio, we ensure that we do not have a too large policy update because the current policy can't be too different from the older one.

To do that, we have two solutions:

- *TRPO (Trust Region Policy Optimization)* uses KL divergence constraints outside the objective function to constrain the policy update. However, this method is complicated to implement and takes more computation time.
- *PPO* clip probability ratio directly in the objective function with its Clipped surrogate objective function**.**

$$L^{CLIP}(\theta) = E_t[min\ (r_t(\theta)\widehat{A_t},\ clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A_t}]$$

Equation 2.22 CLIP probability loss in PPO

2.3 You Only look Once (Yolo)

[26]

Computer vision (CV) is the science and technology of enabling machines to interpret and understand visual data. It involves developing theoretical and algorithmic methods to acquire, process, analyze, and understand images or videos and use this information to create meaningful representations and interpretations of the world.

YOLO (You Only Look Once) is a fast and accurate object detection model introduced by Joseph Redmon and his team in 2016.

How YOLO Works.

YOLO divides an input image into an S × S grid. If the center of an object falls within a grid cell, that cell becomes responsible for detecting that object. Each grid cell predicts B bounding boxes, each with a confidence score. This score tells us how sure the model is that a box contains an object and how accurate its position is

.

The Main concepts of YOLO are listed below.

Bounding Box Prediction

YOLO allows each grid cell to predict multiple bounding boxes. During training, we want only one box predictor to handle each object. So, YOLO chooses the predictor with the highest IOU (Intersection over Union) with the true object location. This approach helps each predictor specialize in certain object types or shapes, leading to better performance overall.

Non-Maximum Suppression (NMS)

When YOLO detects objects, it often creates multiple overlapping boxes for the same object. To fix this, it uses a technique called Non-Maximum Suppression (NMS). NMS removes extra or less accurate boxes and keeps the one with the highest confidence, so each object is detected only once.

Network Architecture

YOLO processes the input image using a deep convolutional neural network (CNN). This CNN directly detects objects by predicting class probabilities and bounding box coordinates in one go.

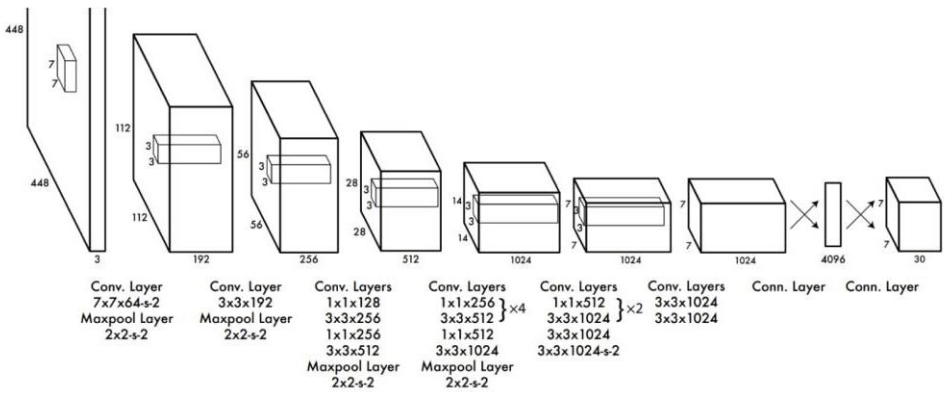


Figure 2.14 Yolo Typical Neural Network

Object Detection

Object detection is a task in computer vision where a model identifies and locates objects within images or videos. It is used in many fields, including security, autonomous vehicles, and robotics.

[27]

Speed Estimation

Speed estimation is performed as part of the object tracking process. Objects are detected and tracked across video frames, and their movement is measured in pixels. When the video frame rate and real-world scale are provided, the object's speed is calculated based on the distance traveled over time. The estimated speed is then displayed for each tracked object.

2.4 Conclusion

In this chapter, the scientific background necessary for understanding the algorithms and techniques used in the project was presented. The next chapter will explore related work, providing comparisons between existing approaches and identifying the scope of the current study.

CHAPTER 3 : LITERATURE REVIEW

In last chapter , we covered the scientific background relevant to the problem, including simulation, algorithms, and computer vision.

In this chapter, the results of the algorithms presented in each paper will be compared, followed by a comparison between the papers and related work. Finally, the need for extending current work will be highlighted, and the scope of the project will be defined.

3.1 Main results of related papers

Based on the results of a study of the algorithms DQN, DDQN, D3QN in [19], D3QN is chosen

Table 3.1 algorithms result according to [19]

	DQN	DDQN	D3QN	Fixed time
Average reward	-222.76	-213.38	-211.22	-
Avg delay	25.06	24.03	23.3	40.17

Based on the results of a study of the algorithms DQN, DDQN in [28], DDQN is chosen

Table 3.2 algorithms result according to [28]

Measure	DQN (low traffic)	DDQN (low traffic)	DQN (high traffic)	DDQN (high traffic)
total negative reward	-3367.6	-4440.8	-113320.2	-77260
total accumulated wait time	14025.8	20139	6222282.6	2822351
expected wait time per car	14.228	20.6837	180.8533	136.3796
average queue length	0.53202	0.7551	22.8937	18.0516

Based on the results of a study of the RF, DNN, XGBoost, Linear Regression (LR) algorithms in [29], XGBoost is chosen

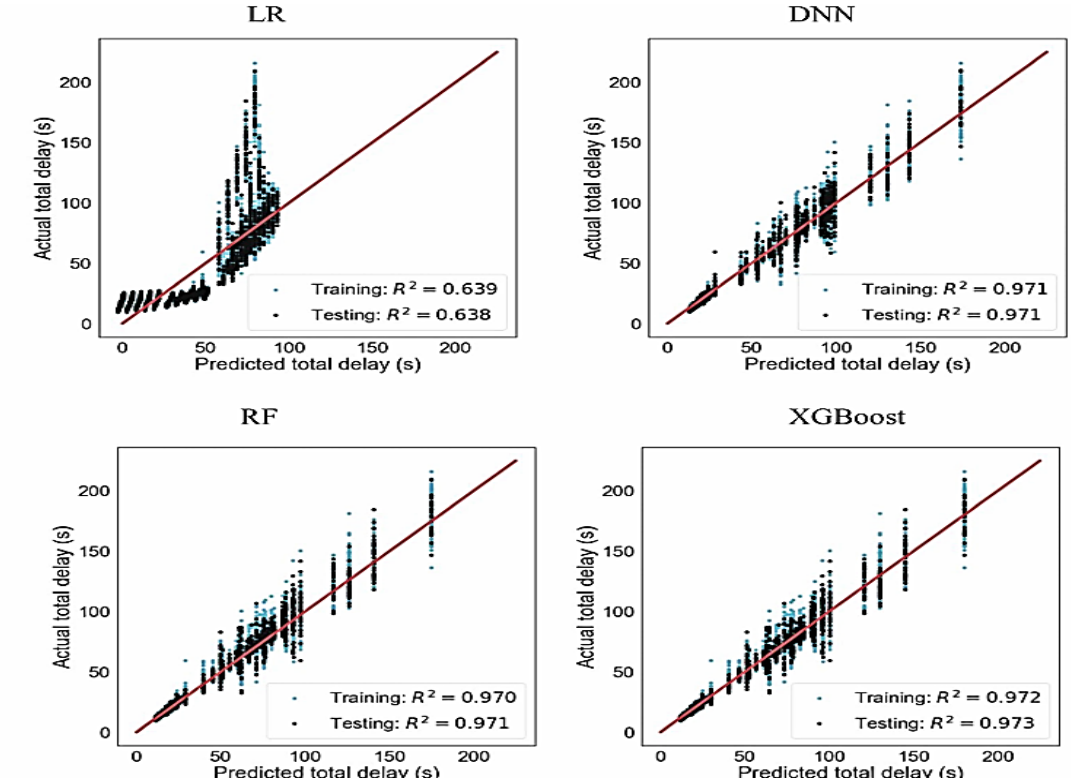


Figure 3.1 Plot of performance of algorithms according to [29]

Table 3.3 Results of algorithms according to [29]

Model	Mean Absolute Error (MAE)	Root Mean Square Error (RMSE)	Mean Percentage Error (MPE)	R^2	Inference time (s per instance) Processing time
DNN	3.609	6.409	0.080	0.971	3.119E-05
RF	3.035	5.991	0.061	0.971	1.111E-05
XGBoost	2.919	5.614	0.060	0.973	4.604E-06

Based on the results of a study of the HMAS, MFDQL-DTC, RACS-A2C, DITLCS, GSCTSA-A3C algorithms in [3], GSCTSA-A3C is chosen

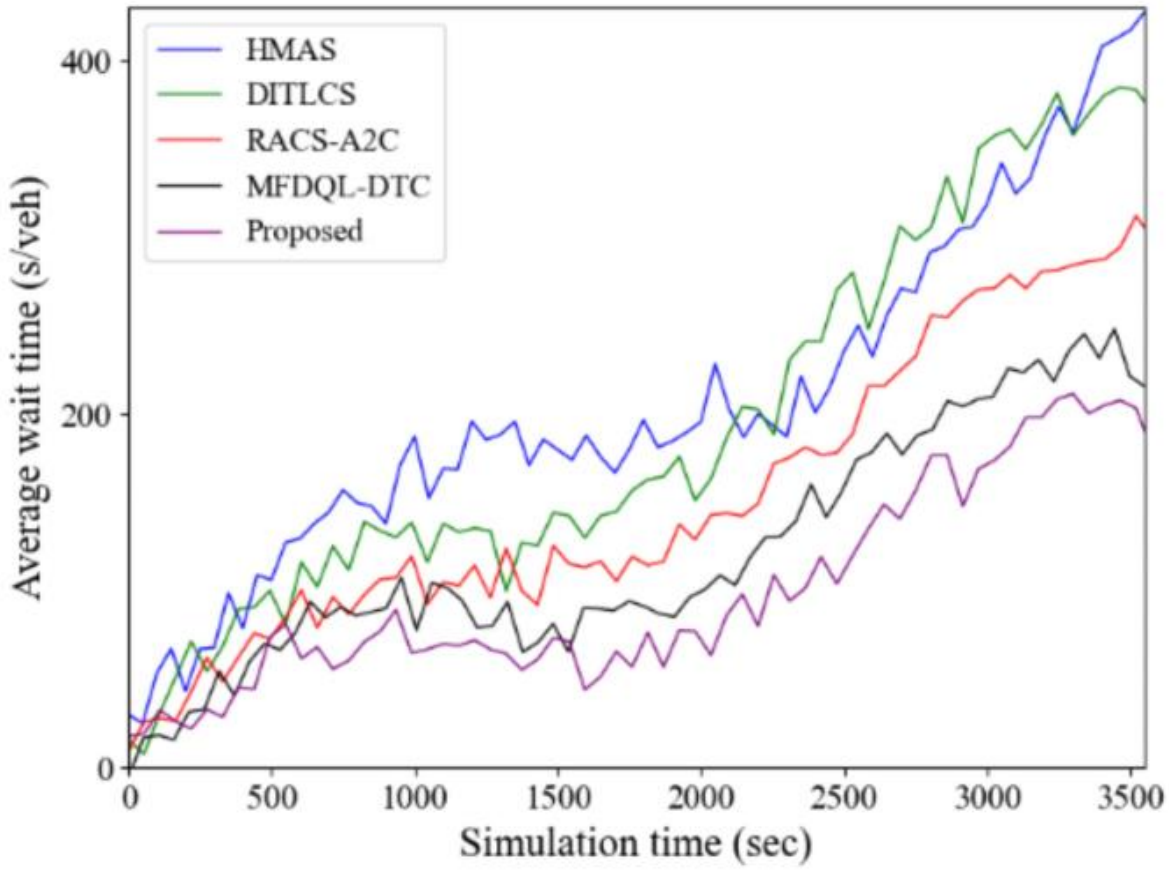


Figure 3.2 Relation between waiting time and simulation time in GSCTSA-A3C according to [1]

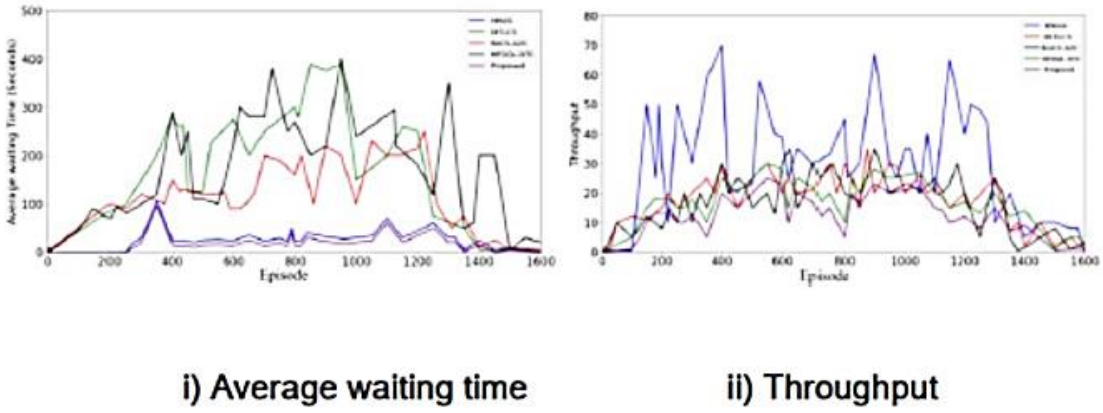
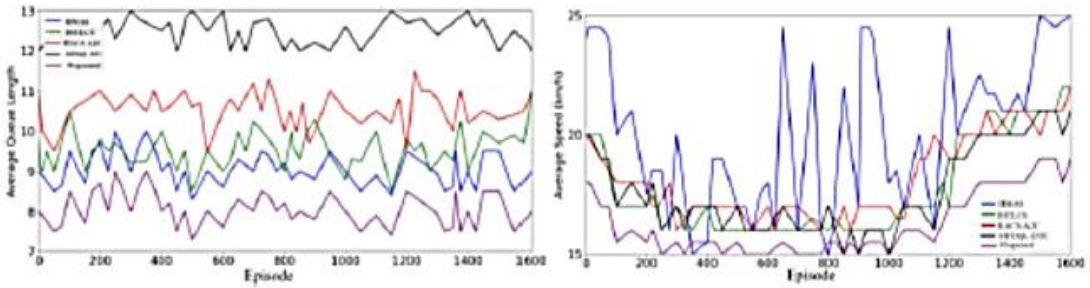


Figure 3.3 Plotting average wait time, throughput in GSCTSA-A3C according to [3]



iii) Average queue length iv) Average speed

Figure 3.4 Plotting average queue time, and speed in GSCTSA-A3C according to [3]

3.2 Comparison between related papers

1) Environment and Characteristics:

The first table compares papers in terms of Year, journal rank, availability of source, environment used in training (as described in Section 2.1), characteristics of the dataset, objective, and learning paradigm (which refers to the type of learning used).

Table 3.4 Comparison between related papers

		[2]	[32]	[19]	[29]	[28]	[30]	[1]	Ours
Year		2024	2022	2024	2024	2024	2019	2023	2025
Journal rank		Q1	Q1	conference	Q1	Q2	package	Graduation Report	Graduation Report
Survey		✓	✓	✓	✓	✓	✓	✓	✓
Open source		No	No	Yes	No	No	Yes	Yes	Yes
Environment	**GYM**	✗	✗	✗	✗	✗	✓	✗	✓
	SUMO	✓	✓	✗	✗	✓	✓	✓	✓
	VISSIM	✗	✗	✓	✓	✗	✗	✗	✗
Characteristics of dataset	**Length of line**	✓	✗	✓	✗	✓	✗	✓	✓
	Flow of traffic	✓	✓	✓	✗	✓	✓	✓	✓
	Weather	✓	✗	✗	✗	✗	✗	✗	✗
	Pedestrians	✓	✓	✗	✗	✗	✗	✗	✗
	Time of day	✓	✓	✗	✗	✗	✗	✗	✗
Objective	**Decrease delay**	✓	✓	✓	✓	✓	✓	✓	✓
	Fairness	✓	✓	✗	✗	✓	✓	✓	✓
	Energy	✓	✗	✗	✗	✗	✗	✗	✗

Learning Paradigms	**Reinforcement learning**	✓	✓	✓	✗	✓	✓	✓	✓
	supervised learning	✗	✗	✗	✓	✗	✗	✗	✗

Characteristics of dataset:

- Length of Line: The distance of vehicles waiting in a queue.
- Flow of Traffic: The movement rate of vehicles on a road.
- Weather: Conditions like rain, sun, or fog that affect driving.
- Pedestrians: People walking near or crossing the road.
- Time of Day: The specific hour, like morning, afternoon, or night.

Objective:

- Decrease Delay: Reducing waiting time for vehicles.
- Fairness: Ensuring all road users get equal priority.
- Energy: The fuel or power used by vehicles.

2) Models:

The second table compare between algorithms used in each paper and the one that got best result of each paper.

			[2]	[31]	[19]	[29]	[28]	[30]	[1]	Ours
Model	**ML**	**Heuristic**	✗	✗	✗	✓	✗	✗	✗	✗
		Random forest	✗	✗	✗	✓	✗	✗	✗	✗
		Extreme Gradient Boosting	✗	✗	✗	✓	✗	✗	✗	✗
	DL	**DNN**	✗	✗	✗	✓	✗	✗	✗	✗
	RL	**A2C**	✗	✗	✓	✗	✗	✗	✗	✗
		A3C	✓	✗	✗	✗	✗	✗	✗	✗
		HMAS	✓	✗	✗	✗	✗	✗	✗	✗
		DITLCS	✓	✗	✗	✗	✗	✗	✗	✗
		RACS-A2C	✓	✗	✗	✗	✗	✗	✗	✗
		MFDQL-DTC	✓	✗	✗	✗	✗	✗	✗	✗
		GSCTSA-A3C	✓	✗	✗	✗	✗	✗	✗	✗
		DQN	✗	✗	✓	✗	✓	✓	✓	✓

		DDQN	✗	✗	✓	✗	✓	✓	X	✗
		D3QN	✗	✓	✓	✗	X	X	X	✓
		PPO	✗	✗	X	✗	X	✓	X	✓
Best Model			GSCTSA - **A3C**	D3QN	D3QN	XGBoost	DDQN	PPO	DQN	D3QN

3) Other Characteristics

The third table compares additional characteristics of the experiments conducted in each paper, such as features, applications (mobile, IoT), and sensors used.

Characteristics

- Edge: A small device or system that processes data near its source.
- Feedback: Information returned to improve a system.
- Handle Accidents: Managing road incidents to reduce their impact.
- Priority for Emergency Vehicle: Giving emergency vehicles, like ambulances, the right of way.
- Predict Traffic: Estimating future traffic conditions.
- Notify with Certain News: Sending alerts about specific updates or events.

Sensor used

- Camera: A device to capture images or video for monitoring.
- Inductive Loops: Sensors in roads that detect vehicles.
- Weather: Environmental conditions like rain, wind, or snow

		[2]	[31]	[19]	[29]	[28]	[30]	[1]	Ours
Characteristics	**Edge**	✓	✗	✗	✗	✗	✗	✗	✓
	Feedback	✗	✓	✗	✗	✗	✗	✗	✗
	Handle accidents	✓	✓	✗	✗	✗	✗	✗	✓
	priority for emergency vehicle	✗	✗	✗	✗	✗	✗	✗	✓
	predict traffic	✓	✗	✗	✗	✗	✗	✓	✗
	Notify with certain news	✗	✗	✗	✗	✗	✗	✗	✗
App	**Mobile**	✗	✗	✗	✗	✗	✗	✗	✓
	IoT	✓ (theoretical)	✗	✓	✗	✗	✗	✗	✓

Sensors	**Camera**	✓	✓	✓	✗	✗	✗	✗	✓
	Inductive loops	✓	✗	✓	✗	✓	✗	✗	✗
	Weather	✓ (Rain-time of day)	✗	✗	✗	✗	✗	✗	✗

3.3 Comparison between similar systems

The following table compares the features of real-world systems.

- Cybersecurity: Protecting systems and data from cyber threats.
- Edge: Local devices or systems that process data near the source.
- Road Safety (Accidents): Measures to prevent or reduce accidents on the road.
- Tunnels: Underground passageways for vehicles or pedestrians.
- Predict: To estimate or forecast future events or conditions.
- Waterproof: Resistant to water damage.
- Identify and Track Traffic Violators: Detecting and monitoring people who break traffic rules.
- Pedestrian Monitoring: Observing and tracking people walking near or across roads.

Table 3.5 Comparison between similar systems

	[32]	[33]	[34]	[35]	Ours
Cybersecurity	✓	✗	✓	✓	✗
edge	✓	✓	✓	✓	✓
Road safety (accidents)	✓	✓	✓	✓	✓
tunnels	✗	✓	✓	✗	✗
predict	✓	✗	✓	✓	✗
Waterproof	✗	✓	✓	✓	✗
Identify and track traffic violators	✗	✗	✗	✗	✗
Pedestrian monitoring	✗	✗	✗	✗	✗

3.4 The Need to Extend Related Work

After a thorough review of the tables, the following conclusions were drawn:

1. No implemented open-source system is adequately documented to explain the technologies or methods used.
2. Only one of the existing papers addresses the Egyptian context, and the latest project report [1] fails to document the entire case.

3. The traffic optimization model is intended to be trained using real-world data, although most related work has been based on simulations due to their cost-effectiveness and suitability for training purposes.
4. The system's simulation should include key factors such as line length and traffic flow, which are commonly considered in similar systems. Additionally, elements like pedestrian presence, weather conditions, and time of day should be incorporated to enhance the model's accuracy.
5. The primary focus of the system should be minimizing traffic delays and ensuring fairness among vehicles. In some cases, energy consumption can be directly considered as an optimization factor.
6. In the latest related work on traffic optimization, reinforcement learning (RL) is used, yielding better results compared to traditional machine learning (ML).
7. Recent research also attempts to consider state temporal and spatial information, consider nearby intersections (by using GAT, LSTM), and optimize the model as much as possible [2],[22],[24].
8. Baseline modeling should involve using an available online model as a baseline, followed by performance improvements through customizing rewards, enhancing the model, training multiple models, or applying techniques such as grid search or Optuna.
9. Deploying the system in real-world IoT environments is essential. Common traffic monitoring sensors, including cameras, inductive loops, and weather sensors, should be utilized when environmental conditions are considered.
10. The system should be designed to be interactive, allowing user feedback to be provided for continuous improvement. Additionally, relevant updates and alerts should be sent to users.
11. Continuous refinement should be conducted by leveraging real-world data collected after deployment, enabling the system to adapt to changes such as government road improvements or the deterioration of road conditions due to factors like rain.
12. Accidents must be handled effectively by ensuring road sections are halted when incidents occur to maintain safety [32],[34],[35].
13. Traffic prediction capabilities should be integrated so that congestion can be anticipated and traffic flow optimized accordingly [32],[34],[35].
14. Cybersecurity measures must be implemented to ensure system safety and compliance with cyber regulations [32],[34],[35].
15. Fast response times should be ensured for both users and traffic conditions, minimizing the impact of factors such as distance and internet latency [2]
16. Traffic violators should be identified and tracked to enforce regulations effectively [32],[34],[35].
17. Support for emergency vehicles must be provided, and awareness of accidents should be maintained to ensure quick and safe passage.

3.5 Scope of Work

According to section 3.4 the scope of work:

- Make a documented open-source project.
- Train on crowded area in Egypt.
- Simulate using SUMO.
- Include main road measurements such as traffic flow and speed in the simulation.
- Minimize delays and ensure fairness.
- Use reinforcement learning (PPO, D2QN).
- Baseline modeling using open-source data from Egypt.
- Develop a small prototype for a single intersection using video analysis with computer vision.
- Latency should be optimized with a near real-time processing architecture.
- A mobile application should be developed.

3.6 Conclusion

In this chapter, the key results from related papers were presented, followed by a comparison of each paper's algorithms, environment, and other characteristics. Existing systems were also compared, and potential extensions were discussed based on the analysis of the tables. Finally, the scope of the work was outlined. The next chapter will focus on system requirements and design, where the project's architecture, system flow, SPI development stages, and applied software engineering methodologies will be discussed in detail.

CHAPTER 4 : SYSTEM DEVELOPMENT

In last chapter the key results from related papers were presented, followed by a comparison of each paper's algorithms, environment, and other characteristics. Existing systems were also compared, and potential extensions were discussed based on the analysis of the tables. Finally, the scope of the work was outlined

In this chapter, we will delve deeper into the project by detailing its life cycle, presenting diagrams, and discussing the software model.

4.1 Software Process Improvement (SPI model)

[36]

The Software Process Improvement (SPI) model was selected due to its iterative structure, which allows evolving requirements to be addressed and complex challenges to be divided across suitable iterations. It will be covered in the areas of phases, team management, iteration, and results.

4.1.1 Phases

The SPI model was implemented through four main phases:

1. **Initiation**: Reference documents and research papers were reviewed during the first term.
2. **Planning**: The system environment was designed, and software requirement specifications (SRS) were prepared.
3. **Execution and Monitoring**: Code was written, reviewed, and technical problems were addressed.
4. **Closure**: Final documentation was completed, and the project was finalized.

4.1.2 Team Management

SPI roles were adapted for the project’s limited scope:

1. **Project Management**: Role assignments and progress monitoring were conducted.
2. **Product Development**: Code implementation was performed.
3. **Peer Reviews:** Code logic was internally reviewed.
4. **Quality Assurance**: Feedback was provided by supervisors.
5. **Configuration Management**: File organization was handled, and final outputs were published on GitHub.

- **Meeting Minutes**: Key discussion points, decisions, and responsibilities were documented to ensure alignment and clarity across the team [Appendix III: Minutes of meeting].
- **Weekly Timesheets**: Work hours and tasks were tracked weekly to monitor individual contributions and project progress [Appendix IV: Weekly Timesheet].
- **JIRA Implementation Progress** The following QR is link to our project JIRA plan, also mentioned in [APPENDIX V: JIRA PLAN].

Figure 4.1 QR code of JIRA plan

4.1.3 Iterations

The SPI model followed across four main iterations:

1 Research on ideas and review of reference papers (First Term).

2 Initial environment design and study of related projects.

3 Completion of environment design.

4 Data modification, experiment execution, and prototype development.

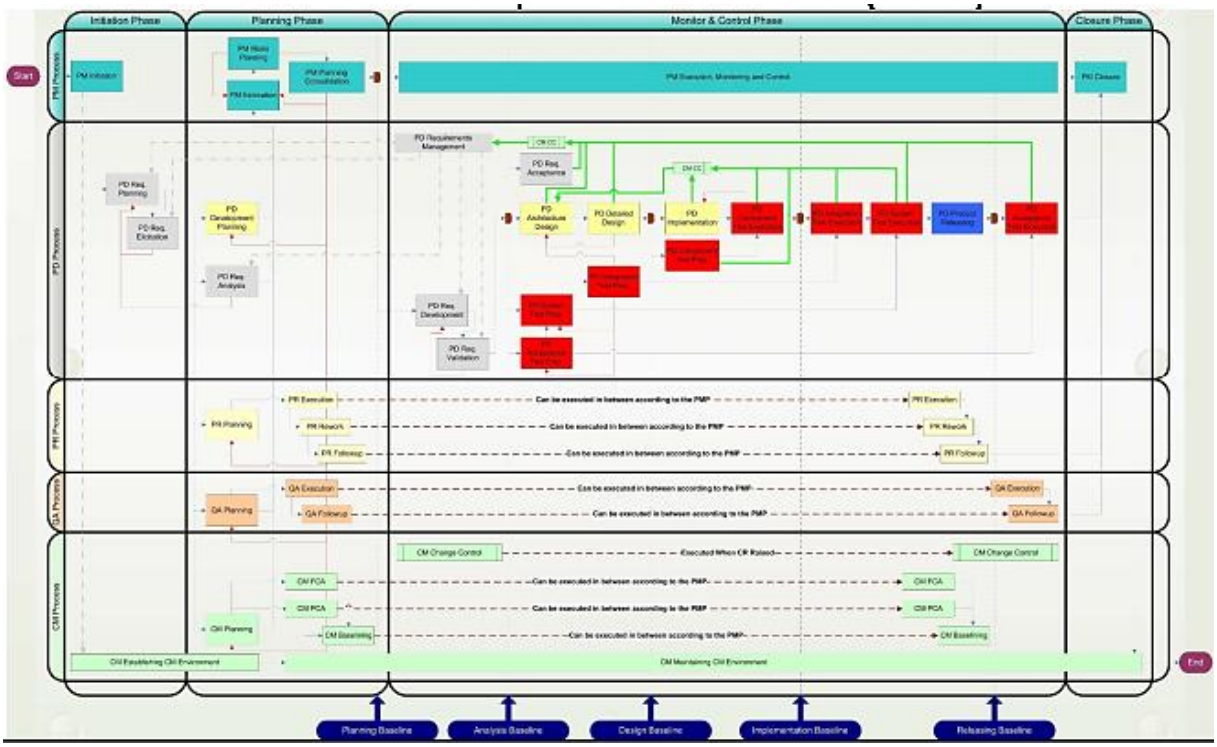


Figure 4.2 SPI Diagram

4.1.4 Results

- A realistic prototype was developed with the focus on training an effective model.
- Future iterations may involve accuracy improvements, use of real data, and optimization for deployment.
- The project was assessed to be at a "Managed" or nearly "Defined" maturity level.

4.2 Process Activity Task Matrix (PATM MATRIX)

In this section, the Process Activity Task Matrix (PATM) is presented, through which the system was analyzed to its main processes. Then, each process was analyzed to its activities. Finally, each activity was analyzed to its tasks

Table 4.1 PATM Matrix

Process	Activity	Task

Baseline	**Baseline with reference project**	Run a portion of the simulations.
		Define baseline parameters.
		Compare results.
	Implement The Model	Create a modified environment.
		Perform data preprocessing and analysis.
		Conduct experiments at different scales.
		Analyze results
	Create modified data	Modify the data.
	Create a prototype of model.	Choose the best model results and save them.
		Set up ESP32 as a single-lane traffic light.
		Analyze one video, estimating it as a single lane.
Emergency	**Maintaining emergency vehicle**	Create account
		Log in to the account.
		Select vehicle route
		Make a database for connecting to the server
		Manage Database using Backend Service
		Send place to server database.
Publication	**Documentation**	Documentation
	Open source	Publish public repo on GitHub.

4.3 Requirements

Functional Requirements

The proposed intelligent traffic management system will provide the following core functionalities:

- Adaptive Traffic Signal Control (FR-01):
  Uses Reinforcement Learning (RL) to dynamically adjust traffic light timings in real-time, aiming to reduce vehicle waiting time, delay, and time loss while improving traffic speed.
- IoT-Based Data Collection (FR-02):
  Integrates cameras and computer vision to detect vehicles and collect traffic data continuously for analysis and model training.
- Emergency Vehicle Priority Handling (FR-03):
  Authenticated emergency vehicles can log in, report incidents, and receive dynamic signal prioritization along their route without disrupting overall flow.
- Accident-Triggered Signal Adjustment (FR-04):
  Authenticated users can report accidents with location/severity, and the system will respond by adjusting nearby signals to restrict traffic and enhance safety.
- Simulation-Driven Optimization & Performance Monitoring (FR-05):
  Uses SUMO for simulation and offline experience data for model training, followed by real-time validation. The system also tracks its own performance using live metrics such as traffic efficiency and model accuracy.

For full Software Requirements Specification (SRS), see [appendix II, Software Requirements Specification (SRS)]

4.4 Design diagrams

In this section, various diagrams will be presented to clearly illustrate the project workflow. These will include the plan, design, experimental setup, prototype, mobile application, the steps to be followed, and the tools to be utilized.

4.4.1 Activity Diagram (Overall Architecture)

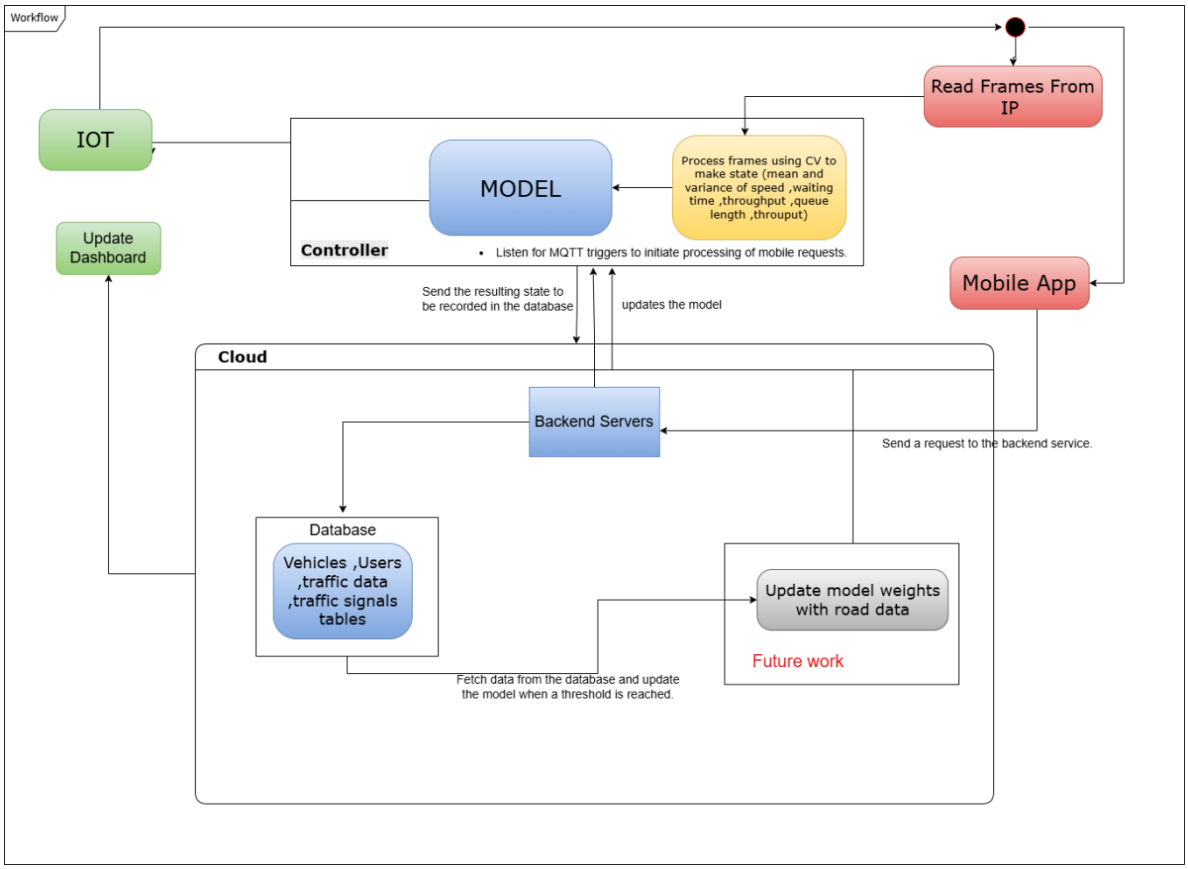


Figure 4.3 Activity Diagram

4.4.2 High-Level Workflow Flowchart (Experimental Architecture)

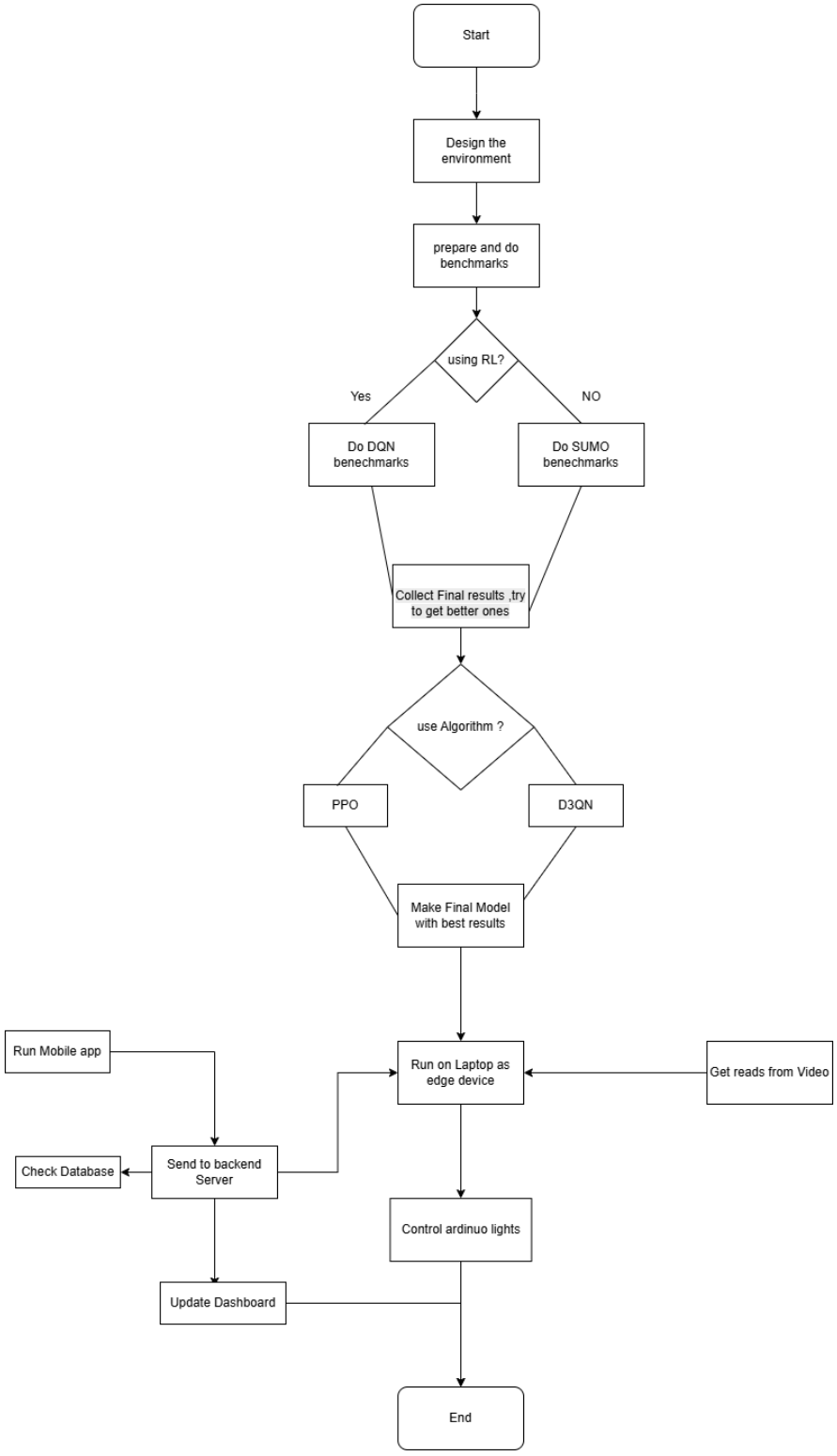


Figure 4.4 High-Level Workflow Flowchart

4.4.3 Tools and Environment

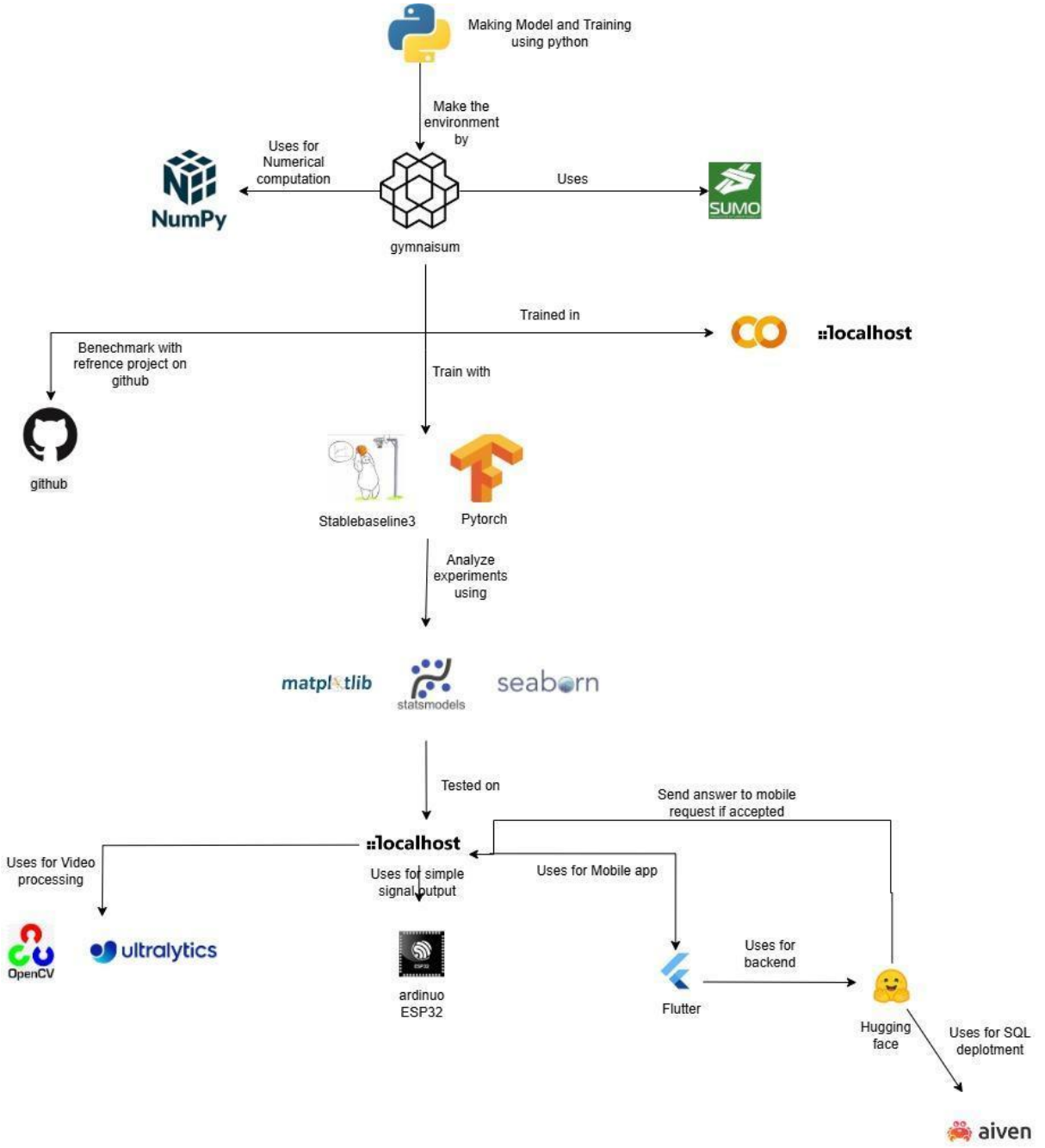


Figure 4.5 Tools and Environment

4.4.4 Use Case Diagram (Mobile App)

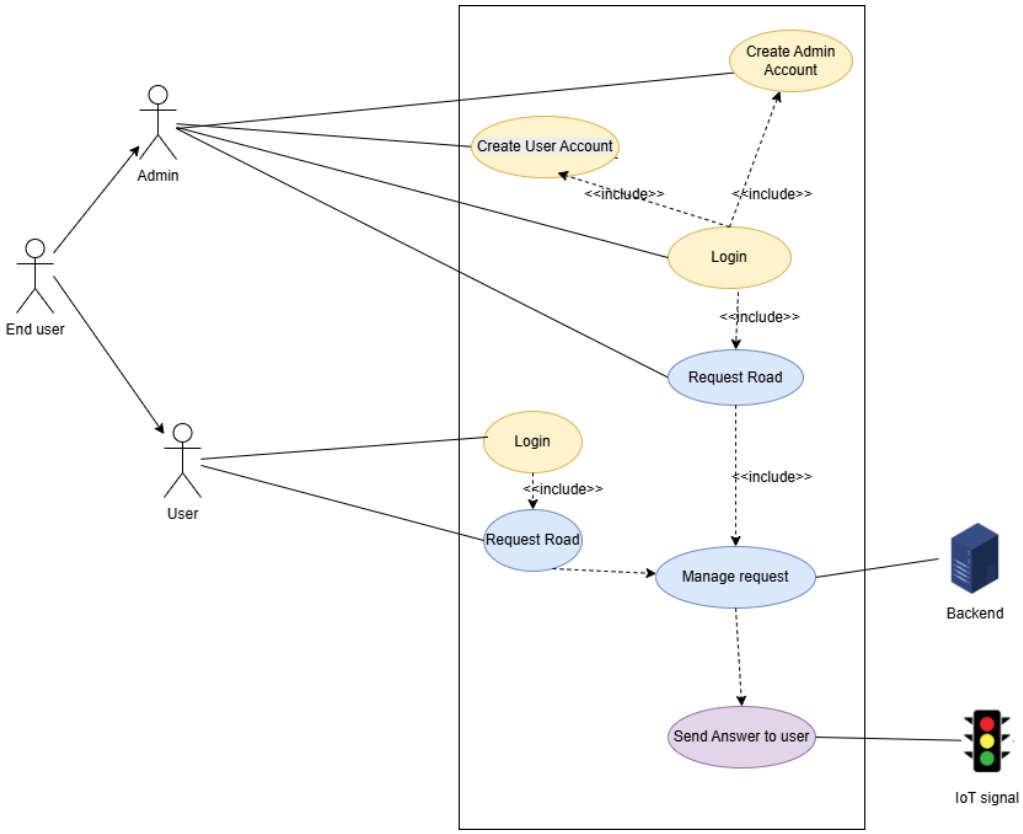


Figure 4.6 Use Case Diagram

4.4.5 Class Diagram

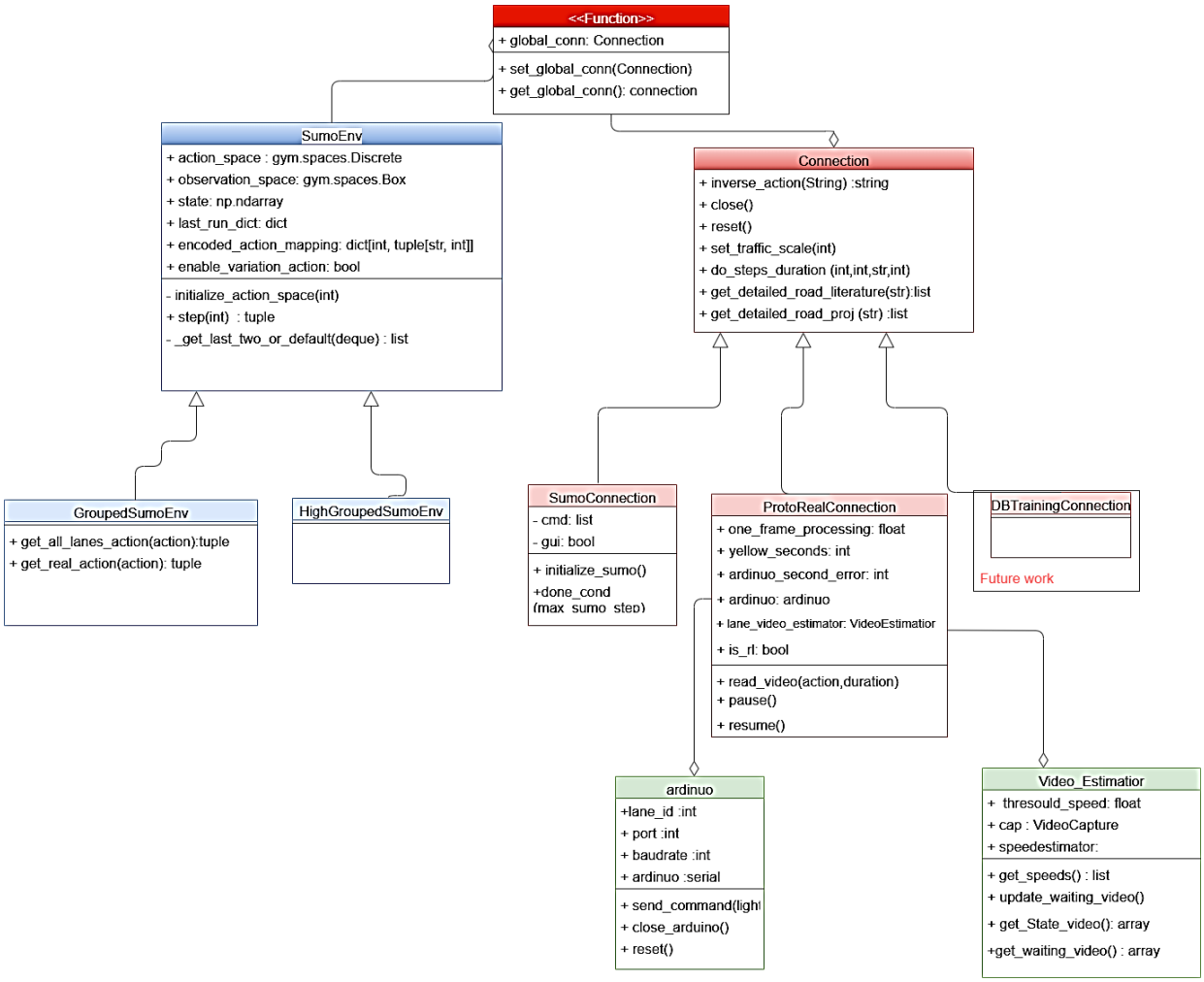


Figure 4.7 Class Diagram

4.4.6 Database Schema Entity-Relationship Diagram (ERD)

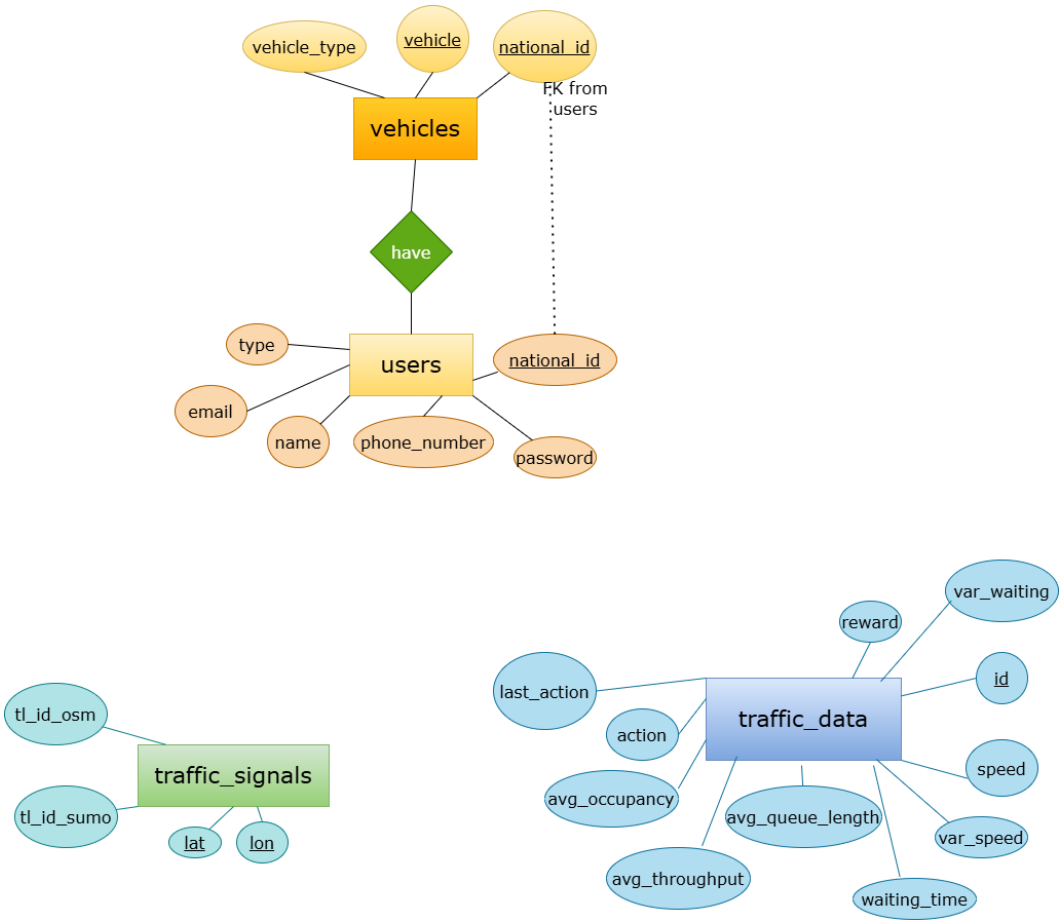


Figure 4.8 ERD diagram

4.4.7 Traffic Backend Component diagram

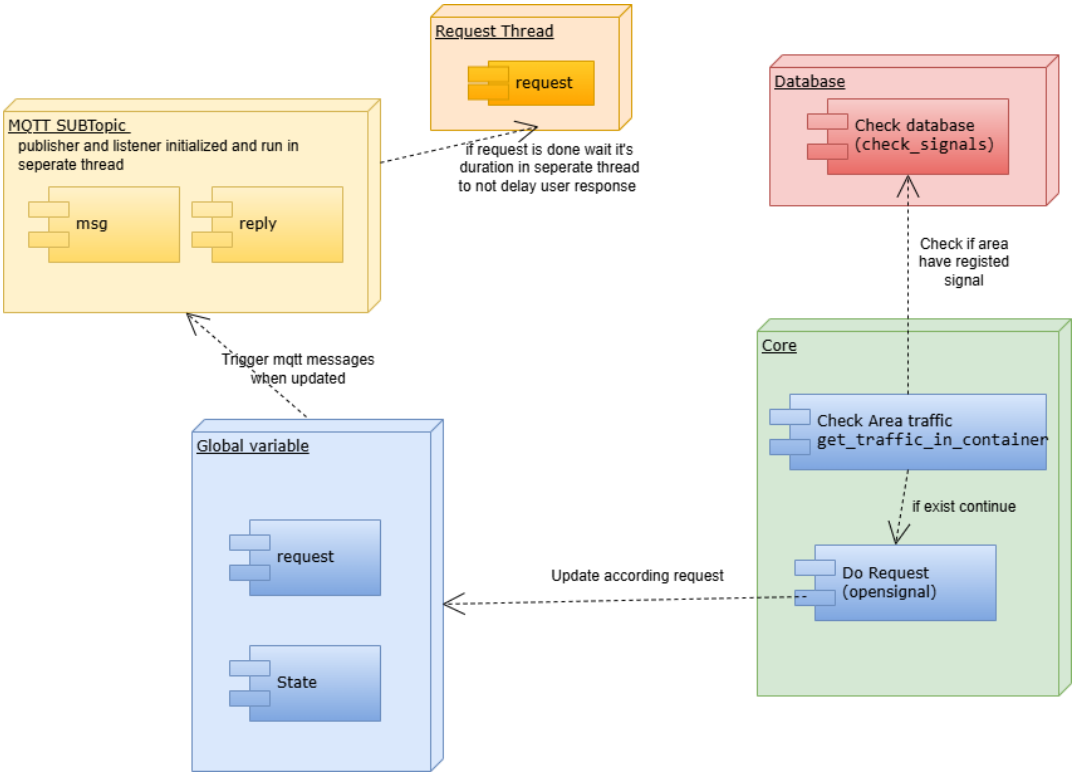


Figure 4.9 traffic backend diagram

4.4.8 Prototype ESP32 Schema

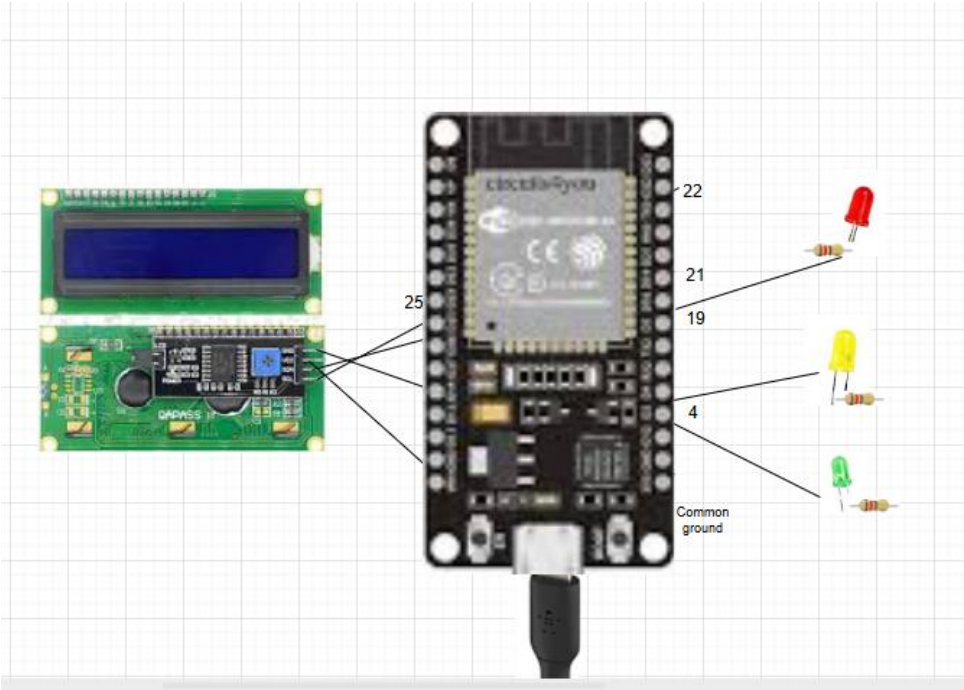


Figure 4.10 Prototype ESP32 Schema

4.4.9 Pipeline of traffic management processing model

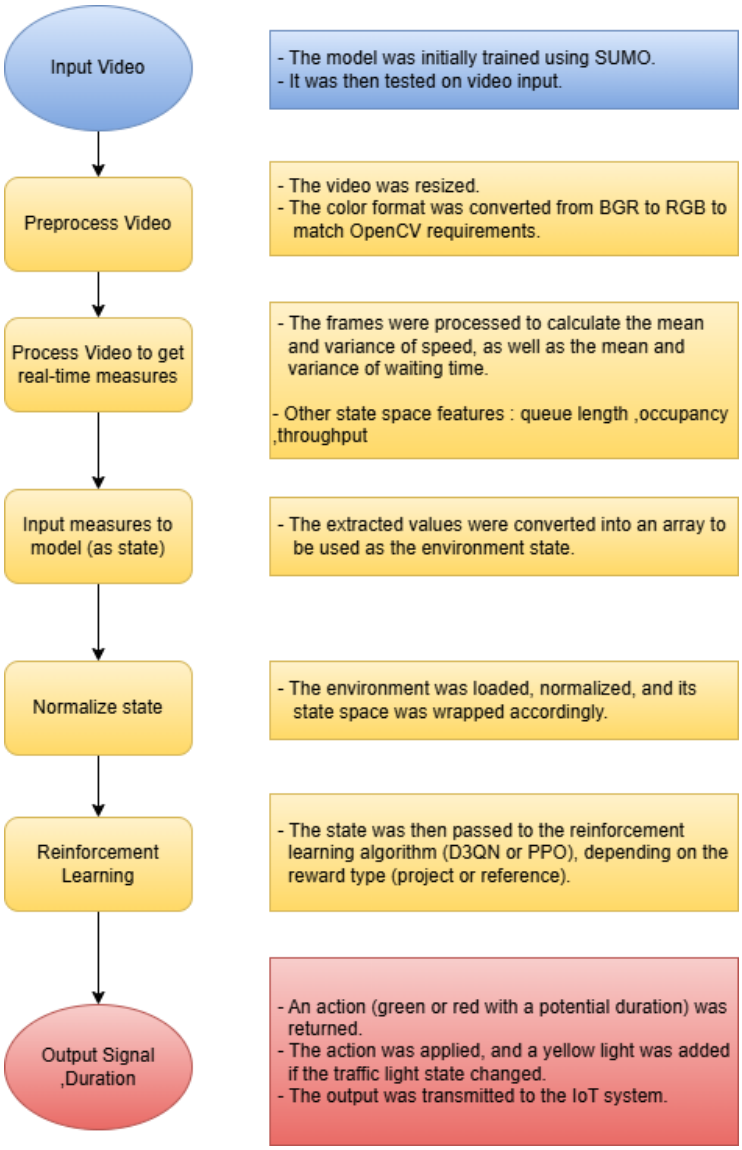


Figure 4.11 Pipeline of traffic management processing model

4.5 Prototype design

In the previous section, the environment design, experiment setup, and experimental results were discussed. In this section, the developed prototype is described, which utilizes the model that achieved the best performance during the analysis phase following training.

To implement the system, traffic measures are intended to be obtained from controlled lanes and averaged. However, in this prototype, the focus was limited to estimating one lane using individual measurements and controlling a single traffic light via a hardware-based setup.

4.5.1 Computer Vision Implementation

4.5.1.1 State Space estimation

As defined in the [section 5.1.3 State Space], the environment state includes 7 road measures:

- Mean speed
- Speed variance
- Mean waiting time
- Waiting time variance
- Occupancy
- Throughput
- Queue length

These features are used to inform the model's actions based on current road conditions.

In the prototype, due to time constraints, only **speed** and **waiting time** were directly measured. The remaining measures were estimated as follows:

- **Mean speed** and **speed variance** were computed based on detected vehicles.
- **Waiting time** was estimated by measuring the duration each vehicle's speed remained below a threshold (as in SUMO definitions).
  The logic used was:
  - For each frame:
    - If a vehicle was still in a "processing" state (not yet exited), the global waiting time was increased by 1/fps.
    - If the frame was not processed but the vehicle had not exited, the same increment was applied (this challenge is detailed in [Section 5.6.1.2 Error Estimation].
    - If the light was red/yellow, the waiting time was also incremented accordingly.
    - When a vehicle exited (i.e., newly processed), it was removed from the waiting list.
- **Mean and variance of waiting time** were computed from the accumulated individual waiting times.

The other features were **estimated** using the following formulas:

$$\text{occupancy} = \min\left(\frac{n_{car}}{10}, 1\right) * 100$$

Equation 4.1 Estimated occupancy

Assuming max 10 vehicles per lane

$$\text{queue\_length} = n_{waiting\_car}$$

Equation 4.2 Estimated queue length

$$\text{throughput} = n_{car} * fps$$

Equation 4.3 Estimated throughput

where fps is number of frames per second

This equivalent to Rate of passing vehicles

The ultralytics library was used for object detection (YOLOv11n).

4.5.1.2 Prototype Video Streaming and Processing Delays

For the prototype, a FastAPI app was developed to stream random videos from the internet, due to limited access to real-life data. The app was configured to run on a local IP (localhost), and the video feed was simulated to mimic a connected camera.

The prototype is runned on best results [section 5.7.3 Best Observed Results] got from **Data 3, Scale Normal** (as suitable for video we have)

Streaming API

The details of the API are outlined below. Its primary function is to generate frames from a given video and stream them.

- Small delays are introduced between frames to prevent processor overload.
- Pause and resume functionalities are included; however, they do not perform meaningful thing if real video but are implemented to simulate traffic signals given video (e.g., stopping or resuming flow after a stop).
- A locking mechanism is applied to ensure thread safety, particularly when the API is accessed simultaneously by a browser and the pause/resume commands are triggered by the application, preventing potential conflicts.

During testing, a frame was captured and held until the processing was completed. YOLOv11 was used for processing, and a maximum processing time per frame of approximately 0.56 seconds was observed. This value, referred to as processing_time_per_frame, was determined based on the performance of the test device, which included an Intel i7 7th generation processor and 16 GB of RAM.

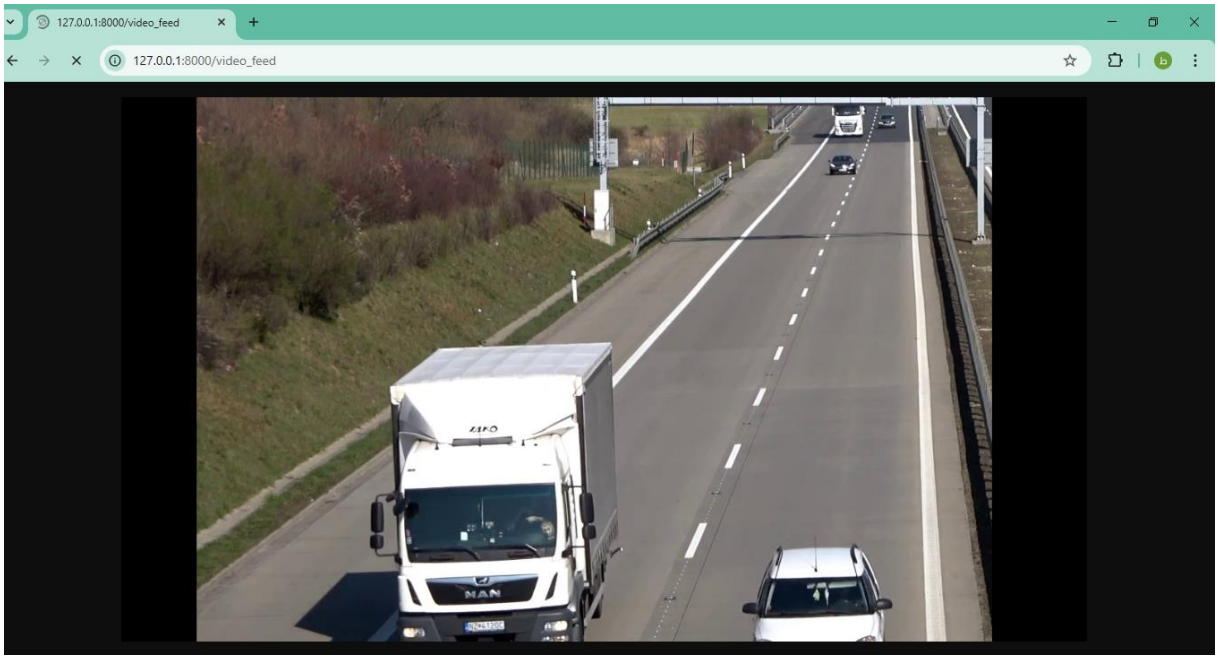


Figure 4.12 Testing prototype on video hosted on local IP address

4.5.2 Ardinuo Implementation

The Arduino was programmed using the C language and the LiquidCrystal_I2C library. Communication with the PC (LOCALHOST /Edge) was handled through pyserial[43]. The Python script handled model predictions and transmitted the corresponding traffic light actions via serial.

Several sources of error were identified in the Arduino component of the system. The primary source was the delay in sending signals to the Arduino. In this project, communication was handled via a serial connection, resulting in an average delay of approximately 0.5 seconds.

The process followed was as follows:

1. A command was sent to the Arduino specifying the full duration.

2. The timing discrepancy caused by the Arduino's delay was used to adjust the video reading offset.

duration_wanted_without_arduino = duration - arduino_second_error

Equation 4.4 duration without arduino error

This cycle ensured that the intended duration x was accurately achieved, compensating for any delays introduced by the Arduino.

An additional source of error was observed during the Arduino reset process. While the initial reset occurred without issue, subsequent resets triggered by application requests introduced minor timing inaccuracies. Due to our time constraints and real-time processing limitations, further reduction of this error was not feasible. However, the impact remained minimal and only occurred when handling specific requests.

4.5.3 Backend Implementation

The backend of the project is structured into three main components:

Traffic_Account

This component is hosted on Hugging Face and implemented using FastAPI. **Link:** [https://taha454-trafficmanager-account.hf.space]

It is responsible for managing login functionality for both users and administrators.

- As described in [**Section 4.4.4 Use Case Diagram (Mobile App)]**, account creation is restricted to administrators. This restriction ensures that only verified users are granted access to the mobile application.

Traffic_Manager

This component is also hosted on Hugging Face and built using FastAPI. **Link:** [https://taha454-trafficmanager.hf.space]

It is designed to handle communication between the mobile application and IoT devices, as detailed in **Section 4.4.7 (Traffic Backend Component Diagram)**.

A known issue discussed: the potential for unreliable internet access at signal locations. This has been addressed using a TCP/IP-like protocol with the following message sequence:

CHK → ACK → DONE QL? → QL.... → EMR/ACC ...duration….

After the availability check, a request for queue length is issued, as opening the signal would be ineffective in cases of severe congestion. However, in certain scenarios, the system may not behave as expected; addressing these cases is left for future work (**Section 6.2**).

4.5.4 Mobile implementation

The "DRIVE MATE" app is designed to facilitate emergency vehicle passage and manage road accidents. It achieves this by opening the traffic signal for emergency vehicles—provided the queue length is less than or equal to 15, as longer queues would render the intervention ineffective—or by closing the signal if an accident is requested that blocks the road.

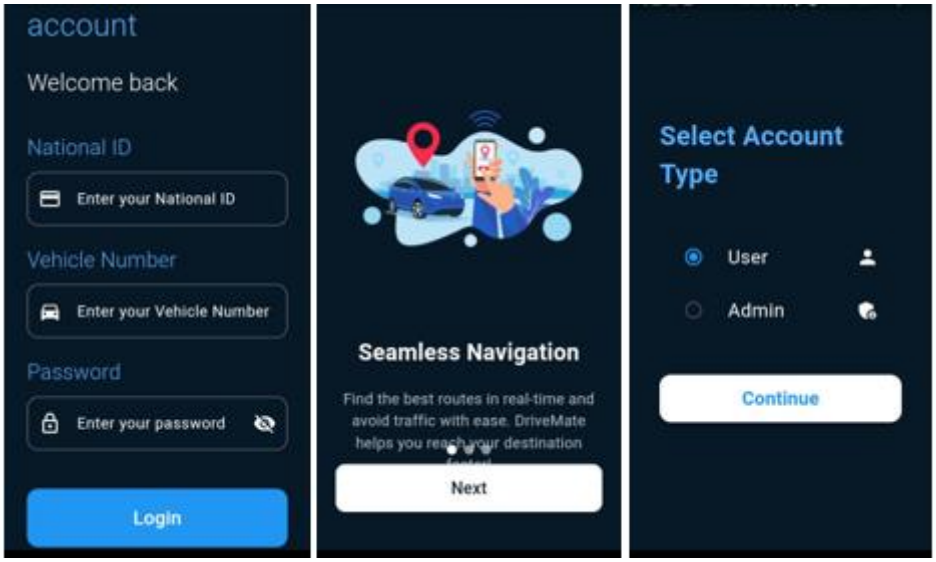


Figure 4.13 Screens of the UI of the prototype mobile app

The whole processes as follows:

1. The user first login, with verification described in [4.5.3 Backend Implementation]

2. A user selects a point, and an retrieved the shortest path.

3. The coordinates are checked against the system's traffic signal database. This is performed by identifying the minimum bounding square that includes the selected path and checking for any registered signals. However, the API may return two signals with opposite directions. Identifying and selecting the signal that matches the user's direction is considered future work (**Section 6.2**).

4. If a valid signal is found, it is activated. The duration and delay are provided by the user; however, future implementations aim to estimate these values automatically based on real-time positioning (**Section 6.2**).

4.5.5 Performance Monitoring

Streamlit [44] was used to develop a simple dashboard for monitoring the system's behavior. This tool enabled real-time observation of system interactions, which facilitated easier debugging and performance evaluation.

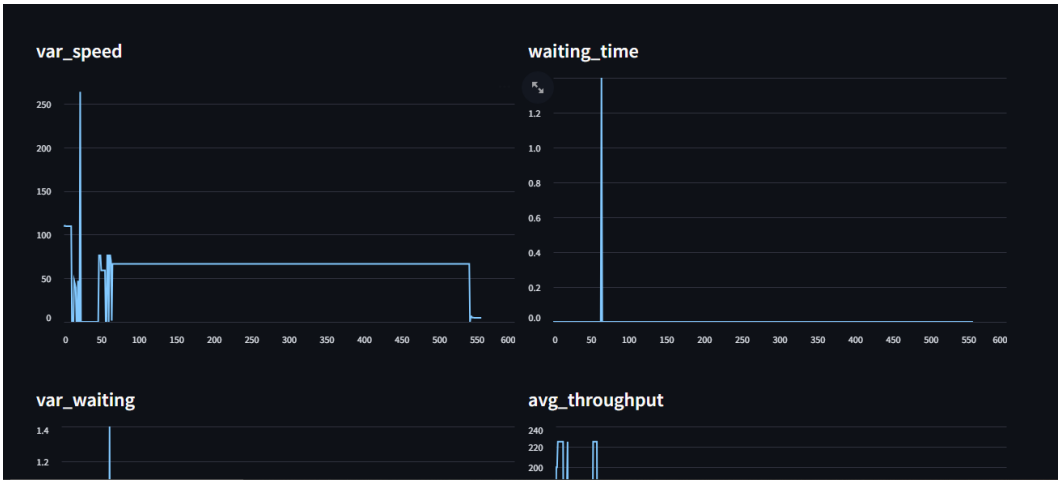


Figure 4.14 Screen of performance monitoring dashboard

4.5.6 Scaling the prototype

The prototype demonstrates a feasible pipeline:
Model training → Analysis → Best model selection → Deployment.

Future work includes retraining the model on real-time data before deployment, as reinforcement learning models are known to be sensitive to environment changes [45].

Further optimizations are proposed:

- Optimize YOLO models for inference time to take less time per frame than .56s for better accuracy or use alternative sensors (e.g., GPS, inductive loops) for faster data acquisition.
- Deploy on edge devices like Raspberry Pi, optimizing low-resource environments.

As suggested in the [section 6.2 Future Work] section, an **online learning setup** should eventually be integrated to allow the model to adapt dynamically to real-world road conditions.

4.6 Module decomposition

In addition to [section 4.4.6 Class Diagram], a modular design was adopted by organizing the project into separate files, each responsible for a distinct function. This structure provided a clear delineation of the system's components, making it easier to understand, maintain, and scale. By isolating functionalities into well-defined modules, development and debugging processes were streamlined, and the overall architecture of the project was made more transparent and extensible.

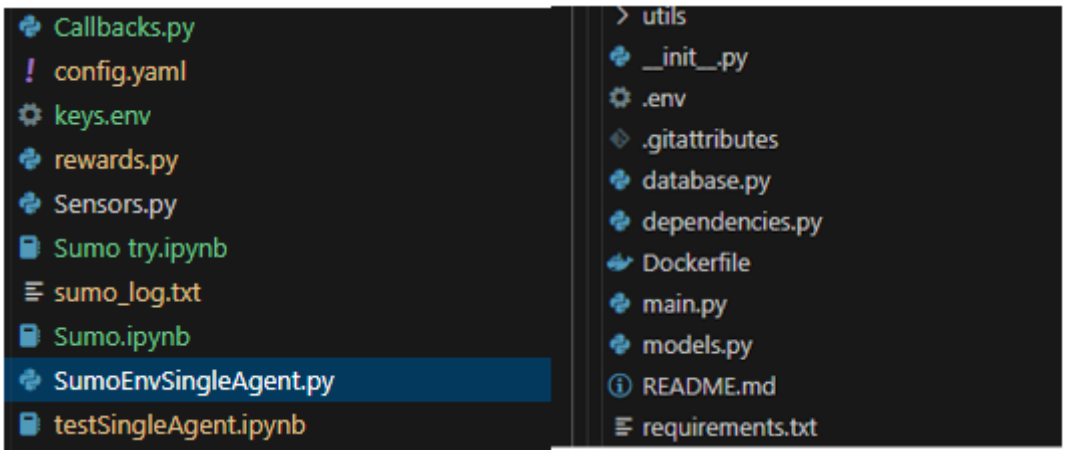


Figure 4.15 Organizing of files in modular design

4.7 Basic implemented security

1. Passwords were securely stored using hashing with bcrypt.
2. Access to the mobile application was restricted to verified users, as discussed in [section 4.5.3 (Backend Implementation)].
3. In accordance with modular design principles, secret keys were stored in separate configuration files.

Further security enhancements—such as restricting database access to a closed IP range and performing vulnerability assessments—are considered future work

4.8 Open-Source Availability

This project is open source and available for public access and collaboration. You can explore the full code and resources by scanning the following QR code:

Figure 4.16 QR code of the project main GitHub

While it is currently shared without a formal license, it is intended to support transparency, collaboration, and future contributions from the community. A specific open-source license may be added in the future to clarify usage and distribution rights [section 6.2 future work].

4.9 Conclusion

In this chapter, we explore the project in more detail, including its requirements and lifecycle. We present relevant diagrams, the JIRA plan, and the software model. These elements provide a structured overview of the project's development. The chapter aims to clarify the project's scope and planning. The next chapter will further elaborate on the experimental design and present the results of it.

CHAPTER 5 : EXPERIMENTAL DESIGN, EVALUATION, AND SYSTEM PROTOTYPING

In the previous chapter, the project was delved into in more depth, its life cycle was detailed, diagrams were presented, and the software model was discussed.

In this chapter, the environment design, data used, state space, action space, reward structure, and the chosen model will be presented. The selection of hyperparameters and the detailed environment setup will then be explained. This will be followed by the presentation of the environment's results and the implementation of the system prototype.

5.1 Environment Design

5.1.1 Problem and Environment Design

As explained in (Section 3.4, The Need to Extend Related Work), the goal was to implement an area traffic optimization system, designed to control traffic with optimal results—minimizing or eliminating waiting time and delays, and achieving the highest possible speeds as allowed by the road conditions. Additionally, the system was intended to be efficient for deployment, with no latency, and capable of adapting as road states change. However, this objective is challenging, as area-wide control constitutes a multi-agent problem. To simulate a realistic scenario, factors such as duration and hierarchical modeling were considered essential.

In this project, the focus was placed on training a single-agent model to achieve the best experimental outcomes. Traffic was modeled under default scenarios, and performance insights were drawn accordingly. A prototype was also developed and tested in a simple scenario using a mobile application.

As detailed in [Section 5.1.4 Action Space], the training focused on the single-agent problem. However, traffic lights typically control multiple lanes, leading to three options for action space design:

- The first option involves assigning independent green/red signals to each lane. While this approach is valid, it tends to introduce noise and may become computationally expensive for intersections with many lanes. Furthermore, it may exceed the capabilities supported by the Gymnasium (Gym) API used in this project.
- The second option groups lanes by direction, which is commonly used in research [28],[22].
- The third option involves controlling all lanes simultaneously. Although less optimal, this reflects the default traffic system design in Egypt, which is the context of this study.

To accommodate these cases, a flexible model was designed. The system supports three configurations: full control (which used in [1]), grouped lanes (used for applying advised strategies), and high-level control as the default scenario.

To prevent the system from getting stuck on a single action, a boolean flag enable_variation_action was introduced. This flag appends the current action to a queue, and if the same action is repeated three times, the action is reversed (e.g., green is changed to red) for each lane.

5.1.2 Data Used

As explained in (Section 3.4, The Need to Extend Related Work), the current work has been focused on the Egyptian region. However, the only found available project with data for Egypt is that used in [1]. In that referenced project, training was conducted on a sample of traffic lights that were controlled using a simple technique—based on looping logic rather than a gym environment—which may pose limitations in scalability for more advanced algorithms and restricts the algorithm's ability to control most areas effectively.

To address this, new Egyptian traffic data was prepared from two sources:

- Data 1: A traffic light was selected at random from the existing dataset, specifically one that controls a high number of lanes, to ensure thorough testing of the code (Tomart).
- Data 2: The traffic lights were estimated to cover approximately the same area as real ones, based on available geographical references, specifically in the Sidi Gaber and Mosher Ismail areas.
- Data 3: Another real dataset was obtained for the San Stefano area.

Using OpenStreetMap (OSM) [36] the system can be used with any other area containing more realistic traffic light configurations (or map is modified to include the signal if not).

Data for a 48-hour SUMO random trip was preprocessed.

Figure 5.1 Data 1 Tomart on google maps (left), SUMO (right)

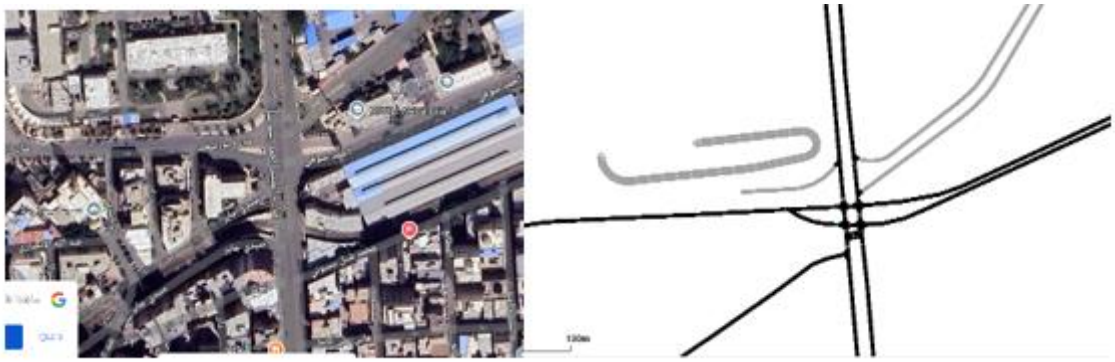


Figure 5.2 Data 2 Mosheer Ismail on google maps (left), SUMO (right)

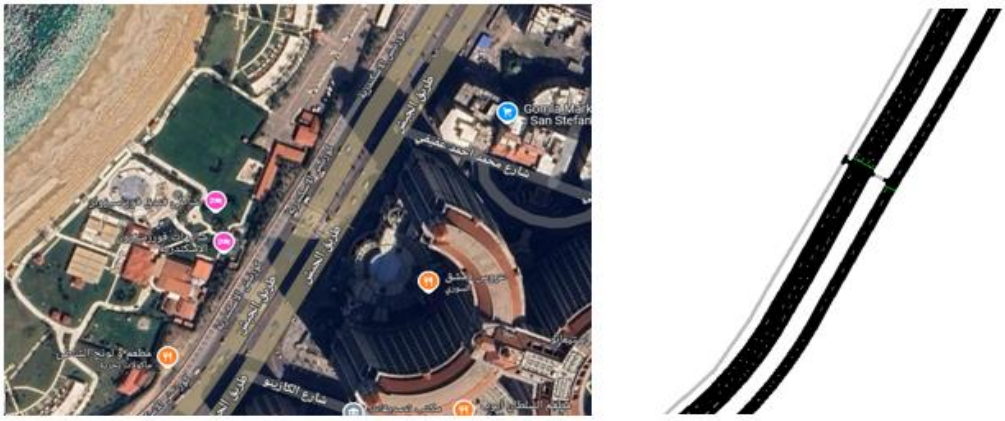


Figure 5.3 Data 3 San Stefano on google maps (left), SUMO (right)

5.1.3 State Space

The state space was designed to include key measures of the main road, along with their variances (reflecting deviation around the mean to ensure fairness in later evaluations). It was also intended to be easily measurable during deployment. The following features were proposed:

Table 5.1 State Space Defination

Measure	Defination
avg_speed (float)	The mean speed of all vehicles observed across controlled lanes.
var_speed (float)	The variance in vehicle speeds observed across controlled lanes.
avg_waiting_time (float)	The mean waiting time accumulated by vehicles across controlled lanes.
var_waiting_time (float)	The variance in the waiting time across vehicles in the controlled lanes.
avg_throughput (float)	The average number of vehicles passing through a lane per time step.
avg_queue_length (float)	The average number of vehicles halted or waiting in the queue across controlled lanes.
avg_occupancy (float)	The average percentage of lane occupancy, representing how much of the lane is occupied by vehicles.

In addition, as part of benchmarking, it was considered that other methods could be called from the connection interface to retrieve further necessary information. This approach is consistent with the logic used in [2], where the reward function differs from the state space.

The state space was designed to capture main road measures that affect the model and was also made easily integrable into the real world.

The measures were averaged from the camera for each controlled lane to maintain a static state space size.

5.1.4 Action Space

As described in the problem definition, each environment level was assigned its own action space.

For each lane, the action was defined as either green or red. To simulate a more realistic scenario, a duration parameter was also introduced into the environment. This parameter consisted of a list of possible durations, from which the algorithm was allowed to select during training.

These action-duration combinations were then encoded as a list and structured as a space to be used later for analysis. The Gym MultiDiscrete space was not used for actions, as some of the intended training libraries do not support it.

The final environment configurations included:

5.1.4.1 HighGroupedSumoEnv (HighGrouped)

$$A=\{(s,d)|s\in\{"r" * n,"g" * n\},d\in D\},\ D=\{d1,d2,...,dk\}$$

Equation 5.1 Action space for HighGrouped Sumoenv

- Where n is the number of lanes
- D is the set of possible durations
- Each action is a pair: signal string s and duration d
- g,r : traffic signal green, red

5.1.4.2 GroupedSumoEnv (Grouped)

$$A=\{(s,d)|s\in\{r,g\}^4,d\in D\},\ D=\{d1,d2,...,dk\}$$

Equation 5.2 Action space for Grouped Sumoenv

- D is the set of possible durations
- Each action is a pair: signal string s and duration d
- g,r : traffic signal green, red

5.1.4.3 Full (SumoEnv)

$$A=\{(s,d)|s\in\{r,g\}^n,d\in D\}\ ,\ D=\{d1,d2,...,dk\}$$

Equation 5.3 Action space for Full Sumoenv

- D is the set of possible durations
- Each action is a pair: signal string s and duration d
- g,r : traffic signal green, red

Each action was then encoded as an integer index within a Discrete Gym space of size |A|, and a mapping was maintained from each index to its corresponding (signal, duration) tuple

To accelerate training in SUMO, the greatest common divisor (GCD) of the given duration array was computed and used as the SUMO step interval. The total number of steps was then calculated by dividing the original durations by this step size.

5.1.5 Reward

As will be discussed in the [5.4 Experiment Design] section, the proposed reward will be used, along with literature-based rewards in the experimental design. A comparison will be made between our work and the reference project's reward. Below is the definition of the rewards.

5.1.5.1 Proposed Reward

$$R = Ss(log\ log\ \left(1 + \frac{\underline{v}}{\sigma_v^2 + \eta}\right)) - Sw\left(\frac{1}{1 + \underline{w} + \sigma_w^2}\right) + Se\left(w_1\underline{t} - w_2\underline{q}\right)$$

Equation 5.4 Proposed Reward

Where:

$\underline{v}$ is average speed, σ_v^2 is variance of speed, $\underline{w}$ is the average waiting time, σ_w^2 is variance of waiting time, $\underline{t}$ is average of throughput, $\underline{q}$ is average of queue length, η is small number to prevent division by zero(1e-6).

Ss, Sw, Se, w1, w2:is weight for terms, configured when testing the reward (w1 =.6, w2=.4, Ss=0.233, Sw=0.667, Se = 0.1), $\left(w_1\underline{t} + w_2\underline{q}\right)$ is efficiency term

The proposed reward is most suitable for a state space that considers all state factors. The main objective is to maximize speed, minimize waiting time, and maximize throughput while minimizing queue length, using a weighted sum approach. Variance is incorporated to ensure fairness. Additionally, a logarithmic function is used to prevent excessively large rewards (with a maximum reward of 5 for most experiments).

5.1.5.2 Reference Project Reward:

R(s,a,a') = (occupancy/haltingNumber) – hamming(a,a') - emergencyStops

Equation 5.5 Reference Reward

The reward used here achieved the best results in the reference project. The second reward function is designed to minimize sudden actions that cause rapid fluctuations by calculating the Hamming distance between consecutive actions. This approach aims to reduce unnecessary fluctuations. Additionally, the model is penalized for emergency stops. The reward function is defined as follows:

The first term in the equation primarily focuses on rewarding the model for high traffic flows. Specifically, the higher the occupancy and the lower the halting number, the greater the reward the model receives.

5.1.5.3 Literature Reward:

[2]

$$reward = \frac{1}{n_{wt}} \sum_{n=1}^{n_{wt}} \lambda \left[1 - \left(\frac{wt_n}{T_{wt}}\right)^{\alpha}\right]$$

Equation 5.6 Literature Reward

where nwt denotes the number of vehicles waiting at the intersection, wtn represents the waiting time of vehicle n, Twt is the acceptable waiting time, and λ and α are constants. Twt is set to 20, λ is set to 0.15, and α is set to 2.

The objective of the reward is to minimize the average vehicle delay while reducing energy consumption and enhancing junction traffic flow. The reward increases as the waiting period decreases. At the same time, fairness is ensured to prevent a small number of vehicles from experiencing excessively long waiting times.

The approach focuses on minimizing the average waiting time for all vehicles, divided by Twt to enable comparison with the acceptable waiting time (indicating that shorter waiting times for all vehicles are desirable). Additionally, the parameters λ and α are used to increase the significance of the reward, making it more impactful.

5.1.6 Preprocessing

Since reinforcement learning is heavily influenced by normalization [38], the following methods were used:

1. **Normalization was performed using:**
   - **VecNormalize** (for Stable-Baselines PPO): Automatically normalizes observations and returns to stabilize learning.
   - **NormalizeObservation** (for D3QN using Gym): Scales state observations to a standard range for more stable convergence.
2. **The environment was wrapped using DummyVecEnv**: Allows vectorized environments to be used by algorithms that expect batched inputs, even when using a single environment instance.

5.1.7 Model Used

- **DQN** was selected, as it was used in the benchmark project [1], to allow for result comparison see [Section 5.4.1.2 Benchmark].
- **D3QN** was chosen, as it achieved better results in [19], [31]. algorithm code is inspired by [8]
- **PPO** was employed as a modern and widely used reinforcement learning algorithm and was also utilized in [39].

5.1.8 Traffic scale

To measure traffic under different scenarios, experiments were conducted at two scales: Normal and Crowded [section 5.4.1 Parameters Defination of Experiments].

In SUMO, the scale is defined using an integer value; however, for this project, it needed to be estimated from real-world data. In the reference project [1], large scale values such as 25 or 50 were used, which are suitable for extensive areas. However, since this project focuses on a single intersection, such large values were deemed unnecessary [5].

As mentioned in [40], it can be estimated that the congestion level in Alexandria is approximately 14% under normal conditions and around 38% during crowded scenarios. Based on this, scale values of 1 (14% × 10) and 3 (38% × 10) were selected for the Normal and Crowded scenarios, respectively.

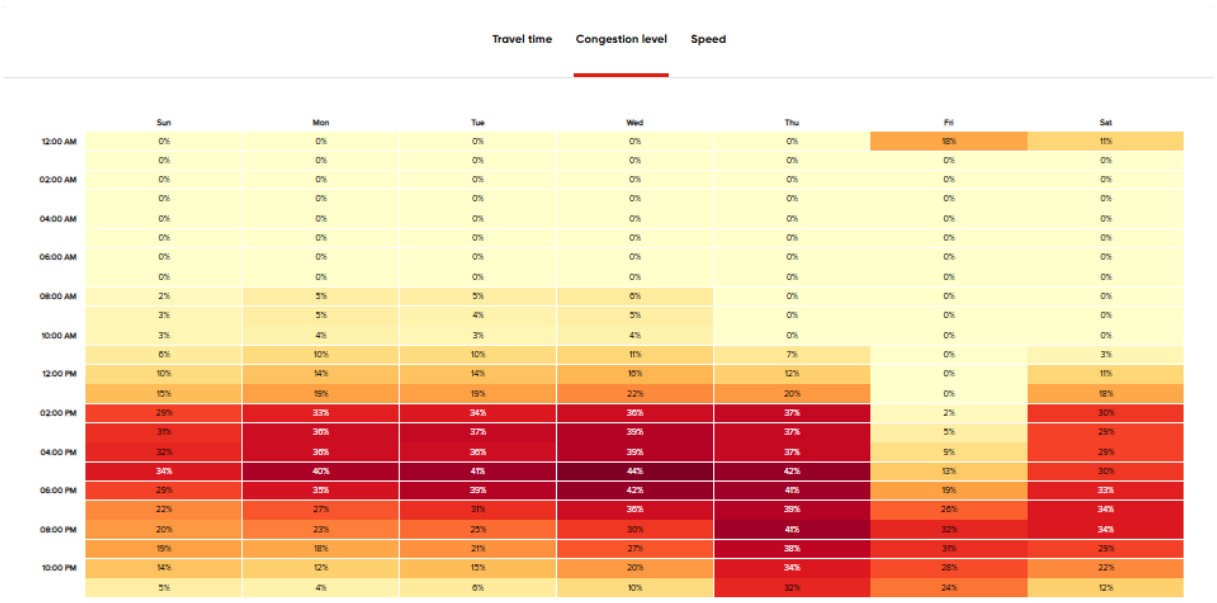


Figure 5.4 Congestion level in alexandria according to TomTom

5.2 Hyperparameter Selection and Tuning

For this project, the parameters were selected with the goal of achieving the highest possible performance. These parameters are categorized as follows:

- Algorithm Hyperparameters: These were determined through a hyperparameter optimization process.
- Benchmark Parameters: These were adopted as-is to ensure fair and consistent comparison with existing results [1].
- Experimental Parameters: These were used to compare the performance of different algorithms and to select the one that performed best.

Further details are provided in the following sections.

5.2.1 PPO and D3QN Hyperparameters

For tuning the PPO and D3QN algorithms, Optuna was employed to automatically search for optimal configurations.

- For D3QN, the following parameters were tuned:
  learning_rate, gamma, tau, l2_reg, epsilon_decay, and batch_size.

Table 5.2 D3QN hyperparameters

Parameter	Meaning	Range
learning_rate	Step size for updating network weights	1e-6 to 1e-3
gamma	Discount factor for future rewards	0.9 to 0.9999
tau	Soft update rate for target network	0.8 to 1.0
l2_reg	L2 regularization factor to prevent overfitting	0.001 to 0.01
epsilon_decay	Rate at which exploration decreases	0.0001 to 0.4
batch_size	Number of samples per training batch	[32, 64, 128, 256, 512]

- For PPO, the following parameters were tuned:
  learning_rate, gamma, gae_lambda, ent_coef, clip_range, batch_size, and net_arch.

Table 5.3 PPO hyperparameters

Parameter	Meaning	Range

learning_rate	Step size for updating network weights	1e-6 to 1e-3
gamma	Discount factor for future rewards	0.9 to 0.9999
gae_lambda	Smoothing factor for Generalized Advantage Estimation	0.8 to 1.0
ent_coef	Entropy coefficient for exploration	0.0 to 0.1
clip_range	Range for clipping policy updates to prevent large steps	0.1 to 0.4
batch_size	Number of samples per training batch	[32, 64, 128, 256, 512]
net_arch	Number of neurons in hidden layers of the neural network	[32, 64, 128, 256, 512]

- The parameter ranges were estimated, as no publicly available source explicitly defined where these values should be drawn from.

- The objective function was designed to maximize the derivative of cumulative reward (from last to first episode) to ensure efficient learning.

- Three different random seeds were tested for each configuration, and the average derivative was used as the optimization target in Optuna to ensure robustness.

- Each trial was run for 10 episodes, using 3 seeds and 20 trials, resulting in a total of 10 × 20 × 3 = 600 episodes.
The number of trials (20) was selected as a middle ground to balance between computational cost and result quality.
Durations tested: [15, 30, 60, 90] seconds, with a maximum of 500 SUMO steps per episode.

- Optuna tuning was repeated for each level (in the proposed experiments), for each of the two reward types, and for each algorithm, resulting in:
2 reward types × 2 levels × 2 algorithms = 8 hyperparameter sets.
These are considered critical, as the action space, reward function, and algorithm significantly influence the results.

- Training was conducted using Dataset 2, and the best parameters were validated on Dataset 3.

Final configurations were saved in a config file.

5.2.2 Parameters from the Benchmark

The same hyperparameters used in the benchmark project were adopted to ensure a fair comparison. Minor modifications were made to adapt them to the new environment, as detailed in [Section 5.3 Benechmark Configuration].

5.2.3 Experiment-Specific Parameters:

Additional parameters were varied during experiments to evaluate the system's performance under different conditions, see [section 5.4.3 Experiment design]. These were used to analyze environmental behavior, gain insights, and identify the best setup for deployment.

5.3 Benchmark Configuration

As described in [section 3.4 The Need to Extend Related Work], few projects are available for Egypt. The only available opensource project was [1], which has issues in fairly applying the model to the area. These issues include:

- Working on a subset of agents while obtaining results for the entire area.
- Using a for loop that may not capture dependencies between agents or facilitate effective learning from experiences [section 5.1.1 Problem and Environment Design].

However, this was the only available resource to compare Egypt's performance and using it for comparison with environments designed differently might not be ideal. Therefore, the following modifications were made to the code to ensure it aligns with the requirements of our project and allows for a fair comparison:

- The project was modified to work with a single agent (e.g., by limiting the loop to the target agent).
- The constructor was modified to accept a seed parameter.
- The code was updated to accept action spaces for SumoEnv, GroupedSumoEnv, and HighGroupedSumoEnv.
- The getSensor method implementation was slightly modified to ensure compatibility with our environment.

In addition to these modifications for our environment:

- The same algorithm hyperparameters from the previous project were used.
- This included defining (epsilon DQN) EpsDQN and (Root Mean Square Propagation) RMS_DQNPolicy, where:
    - EpsDQN inherited from Stable-Baselines3's DQN to enable epsilon-greedy with the same minimum, maximum, and decay rate as in the previous project.
    - RMS_DQNPolicy inherited from Stable-Baselines3's DQNPolicy to use RMSProp as the optimizer, as in the previous implementation.
- The model was run for one episode with a duration of 1 (i.e., 1 second per step).

5.4 Experiment Design

The model will be trained on different cases to study its behavior under varying reward structures, action spaces, and other conditions. The configuration that performs best across these scenarios will be selected. Details of the conducted experiments are provided in the following sections.

5.4.1 Parameters Defination of Experiments

5.4.1.1 SUMO static light

Table 5.4 SUMO experiment parameter defination

Parameter	Meaning	Real meaning	Range for experiment
Data	The dataset or environment used	Country-specific traffic data	Area 1 (Tomart)-Area2 (Mosheer Ismail)
Max sumo steps	Maximum number of simulation steps in SUMO	Represents the simulation time span—longer steps simulate longer real-world hours	(500-4000)
Traffic Scale	Traffic load intensity	Adjusts the number of vehicles generated in the simulation	Normal (.14) - Crowded (.38)

5.4.1.2 Benchmark

Table 5.5 Benchmark experiment parameter defination

Parameter	Meaning	Real meaning	Range for experiment
Data	The dataset or environment used	Country-specific traffic data	Area 1 (Tomart)-Area2 (Mosheer Ismail)
Action space	The scope of available agent actions	Reflects how detailed or flexible traffic signal decisions are in the real world	Full (SumoEnv)-GroupedSumoEnv HighGroupedSumoEnv
Reward	Type of reward function used	Determines how traffic control goals are	Reference Project(project)

		prioritized (e.g., efficiency, fairness)	
Max sumo steps	Maximum number of simulation steps in SUMO	Represents the simulation time span—longer steps simulate longer real-world hours	(500-4000)
Run	Experiment repetition count	The number of times the experiment is repeated to compare environments and ensure consistency	Reference Project (Project) -Proposed
Seeds	Random seed for reproducibility	Run with different seeds to ensure stability and robustness against randomness	0,1,2
Traffic Scale	Traffic load intensity	Adjusts the number of vehicles generated in the simulation	Normal (.14) -Crowded (.38)

5.4.1.3 Environment Experiments

Table 5.6 Environment experiment parameter defination.

Parameter	Meaning	Real meaning	Range for experiment
Area	The dataset or environment used	Country-specific traffic data	Area2 (Mosheer Ismail) – Area 3 (San Stefano)
Reward	Type of reward function used	Determines how traffic control goals are prioritized (e.g., efficiency, fairness)	Literature - Proposed
Environment type	Type of SUMO environment used	Level of traffic control detail and intersection grouping	GroupedSumoEnv HighGroupedSumoEnv
Algorithm	RL algorithm applied	Different learning strategies used to optimize traffic control	D3QN -PPO
Episodes	Number of training episodes	Number of training cycles (longer = more learning)	200,1000 ,10000
Max sumo steps		The number of times the experiment is repeated to compare	500-1500 -4000

	Maximum number of simulation steps in SUMO	environments and ensure consistency	
Traffic Scale	Traffic load intensity	Adjusts the number of vehicles generated in the simulation	Normal (.14) - Crowding (.38)
Seeds	Random seed for reproducibility	Run with different seeds to ensure stability and robustness against randomness	0,1,2,3

5.4.2 Metrices

The metrics aimed to be optimized include the reward obtained in the final training episode, the evaluation reward to ensure generalization beyond training, the reward derivative to confirm learning progress, and road-level performance metrics to assess the impact of model deployment. Further details are presented in the following table.

Table 5.7 Metrics defination

Metric	Defination	Direction
Reward of last epsiode	The total accumulated reward obtained by the agent during the final episode of training	Max
Average Reward for evaluated episodes	The mean of total rewards collected across multiple evaluation episodes.	Max/Equal Reward of last epsiode
Waiting Time	The average time spent standing (seconds)	Min
Speed	The average trip speed (m/s)	Max
Depart Delay	The average time vehicles had to wait before starting their journeys (seconds)	Min
Time loss	The average time lost due to driving slower than desired (including waiting time) (seconds)	Min
Waiting Car	Number of vehicles with delayed insertion that were still waiting for insertion at simulation end	Min
Means of Experiments	The average of a specific metric (all metrices) across multiple independent experiment runs.	Max
Variance of Experiments	A measure of how spread out the experiment results are. High variance indicates inconsistent results across runs.	Min

Total Time	The total time taken to complete the full training or evaluation process, from start to finish. (Seconds)	Min
Derivative of reward	Measures on how reward is changing over episodes — tells if the agent is improving.	Max

5.4.3 Experiment Design

After conducting the initial experiments, we will apply experimental design techniques to gain deeper insights. The most suitable approach is the **Full or Fractional Factorial Design**, as it allows us to systematically study multiple factors and their interactions—even when the number of experiments needs to be limited due to time constraints.

This design will help us answer key questions such as:

- What is the effect of the algorithm on average waiting time or other performance metrics?
- How does increasing the number of episodes impact the results?
- What are the mean differences across various experimental setups?

5.5 Results of Experiments

5.5.1 Results of Benechmark Experiments

5.5.1.1 Plotting for Area 1, Scale Normal

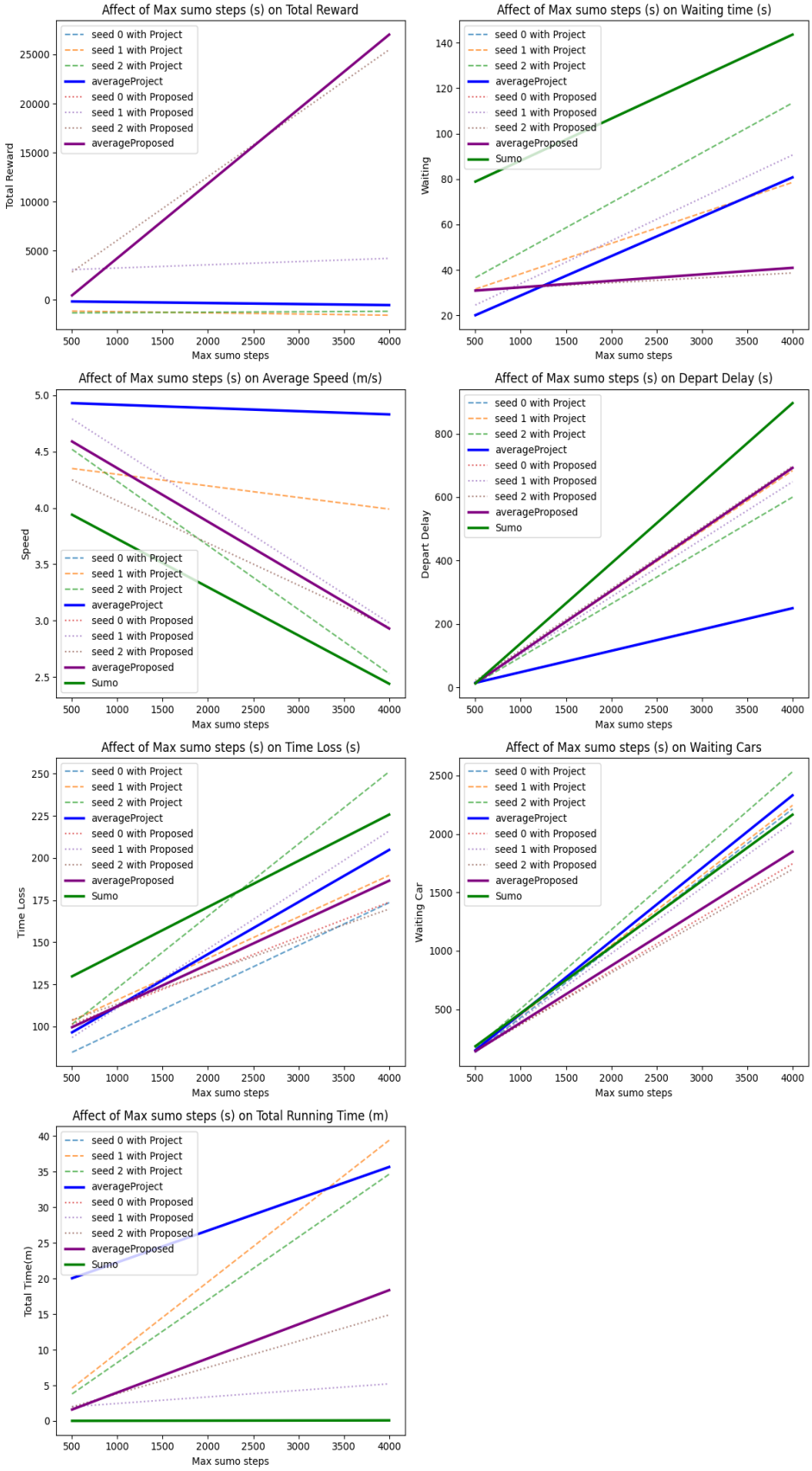


Figure 5.5 Benechmark Plotting for Area 1, Scale Normal

5.5.1.2 Plotting for Area 1, Scale Crowded

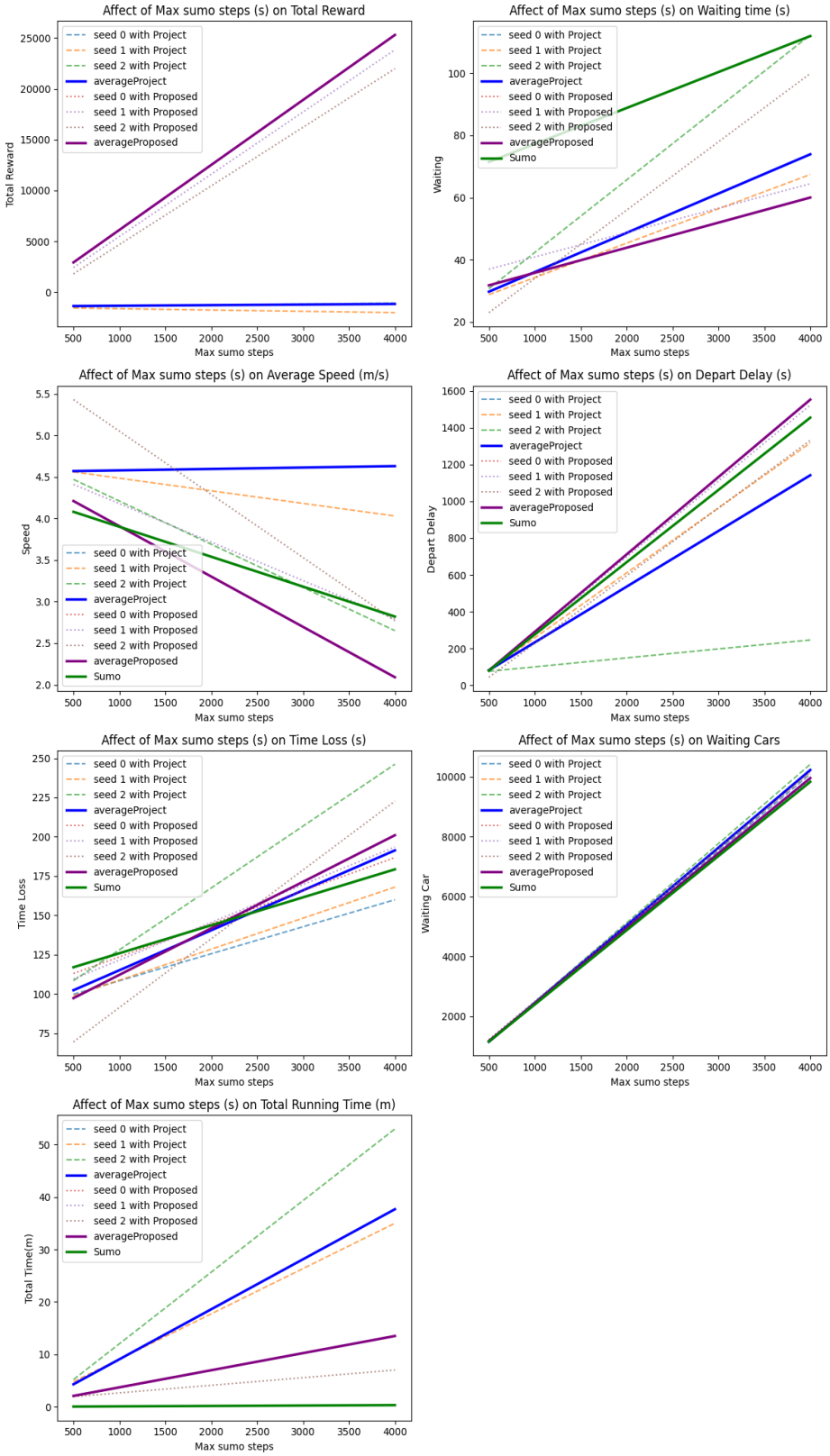


Figure 5.6 Benechmark Plotting for Area 1, Scale Crowded

5.5.1.3 Plotting for Area 2, Scale Normal

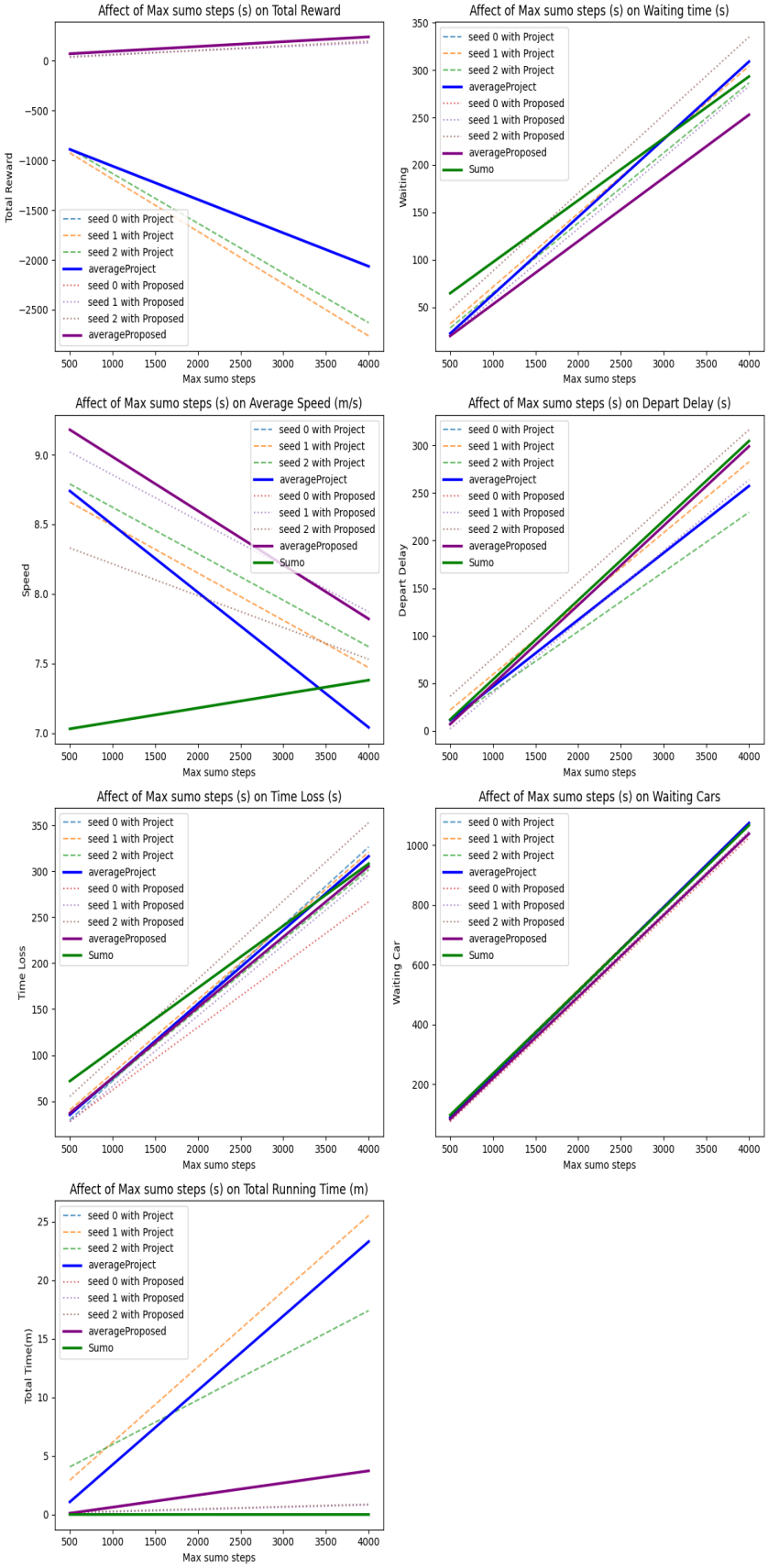


Figure 5.7 Benechmark Plotting for Area 2, Scale Normal

5.5.1.4 Plotting for Area 2, Scale Crowded

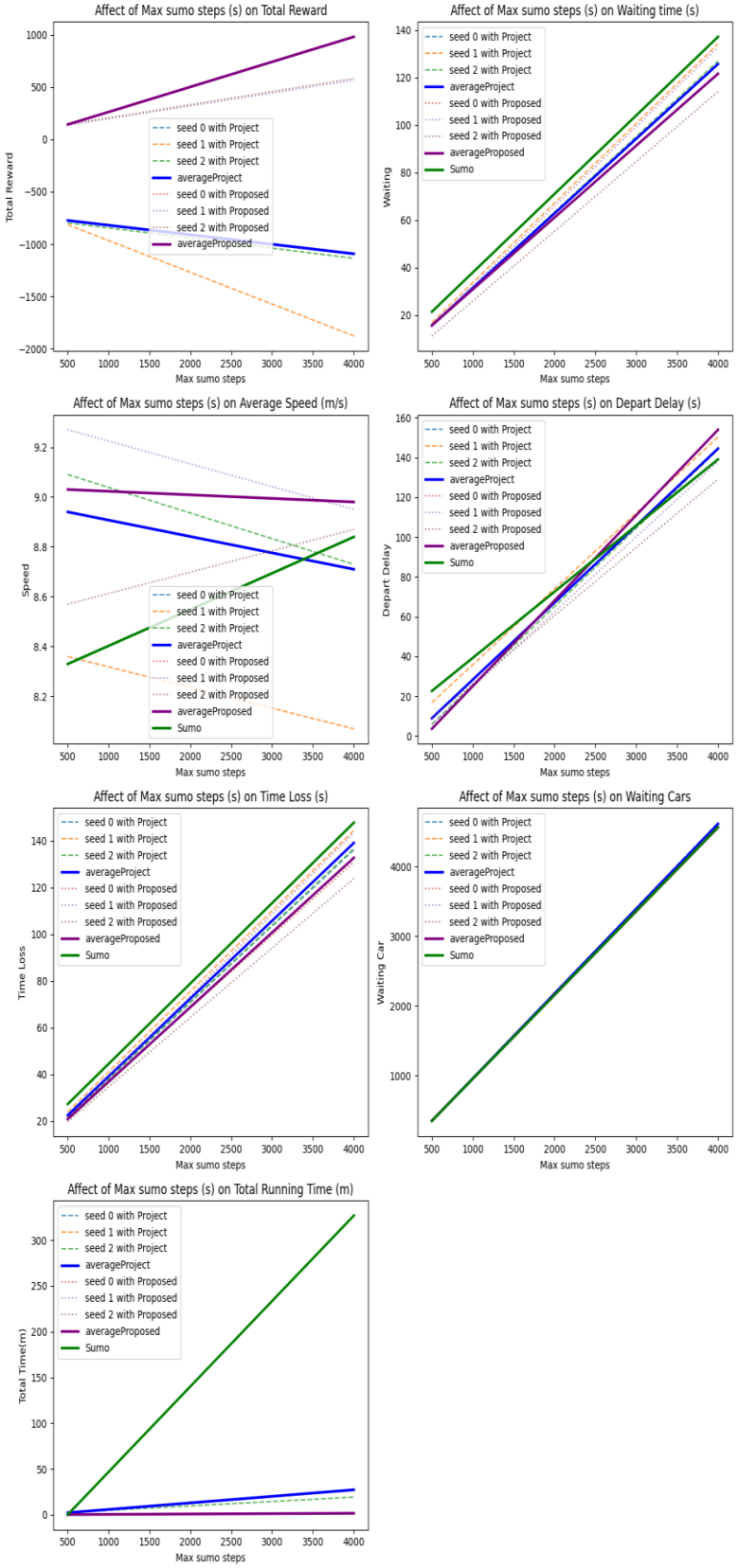


Figure 5.8 Benechmark Plotting for Area 2, Scale Crowded

5.5.2 Results of Environment Experiments

5.5.2.1 Effect on Episodes

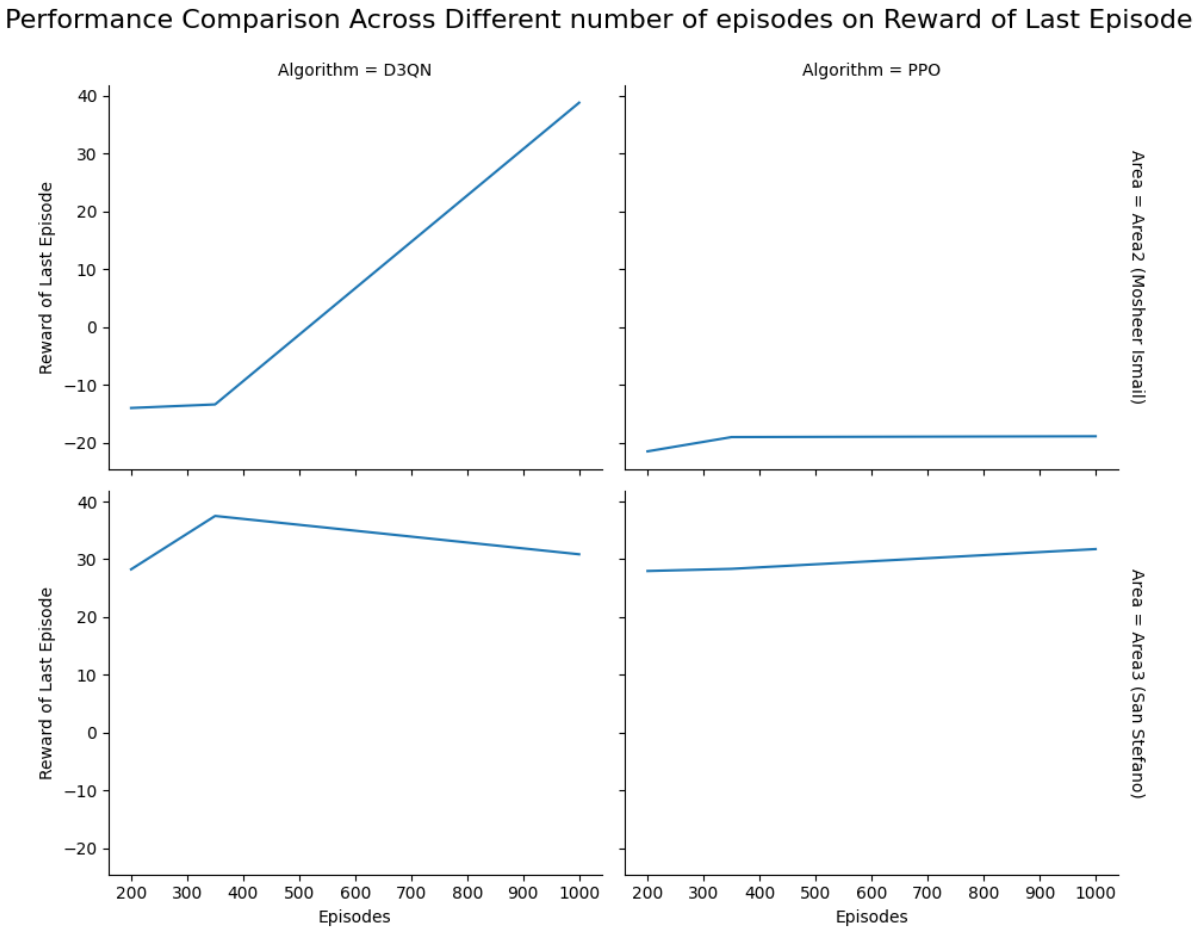


Figure 5.9 Effect of running many episodes on reward

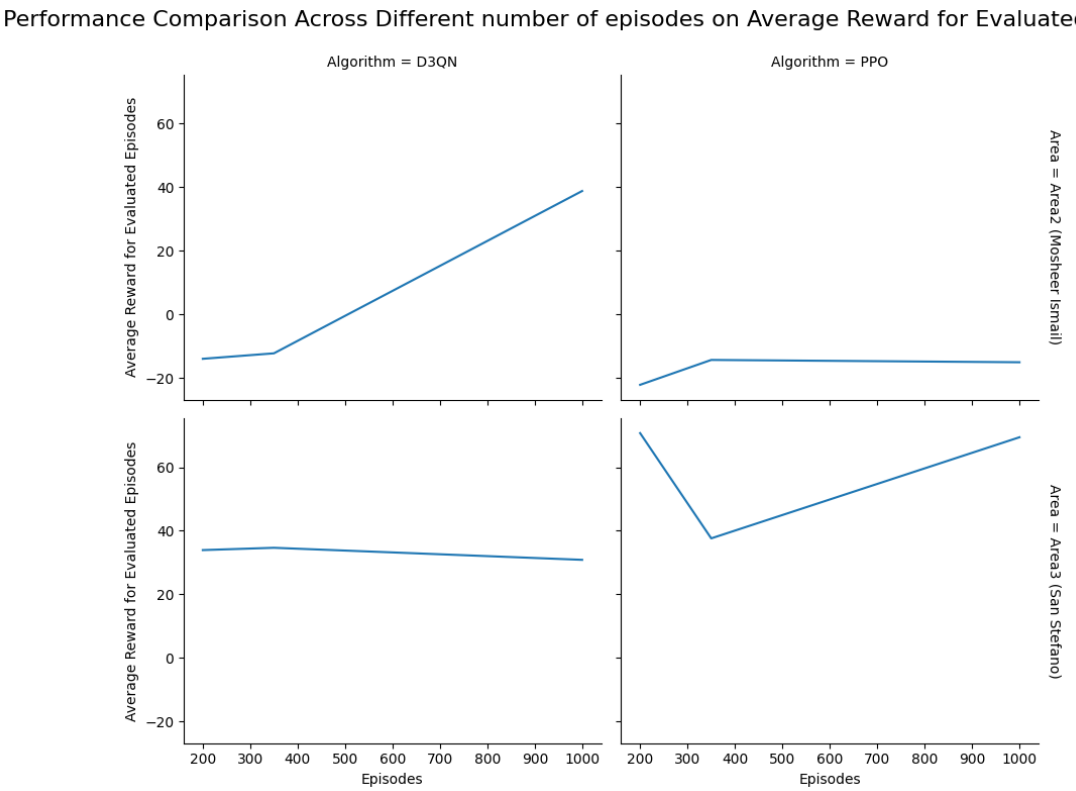


Figure 5.10 Effect of running many episodes on evaluated reward

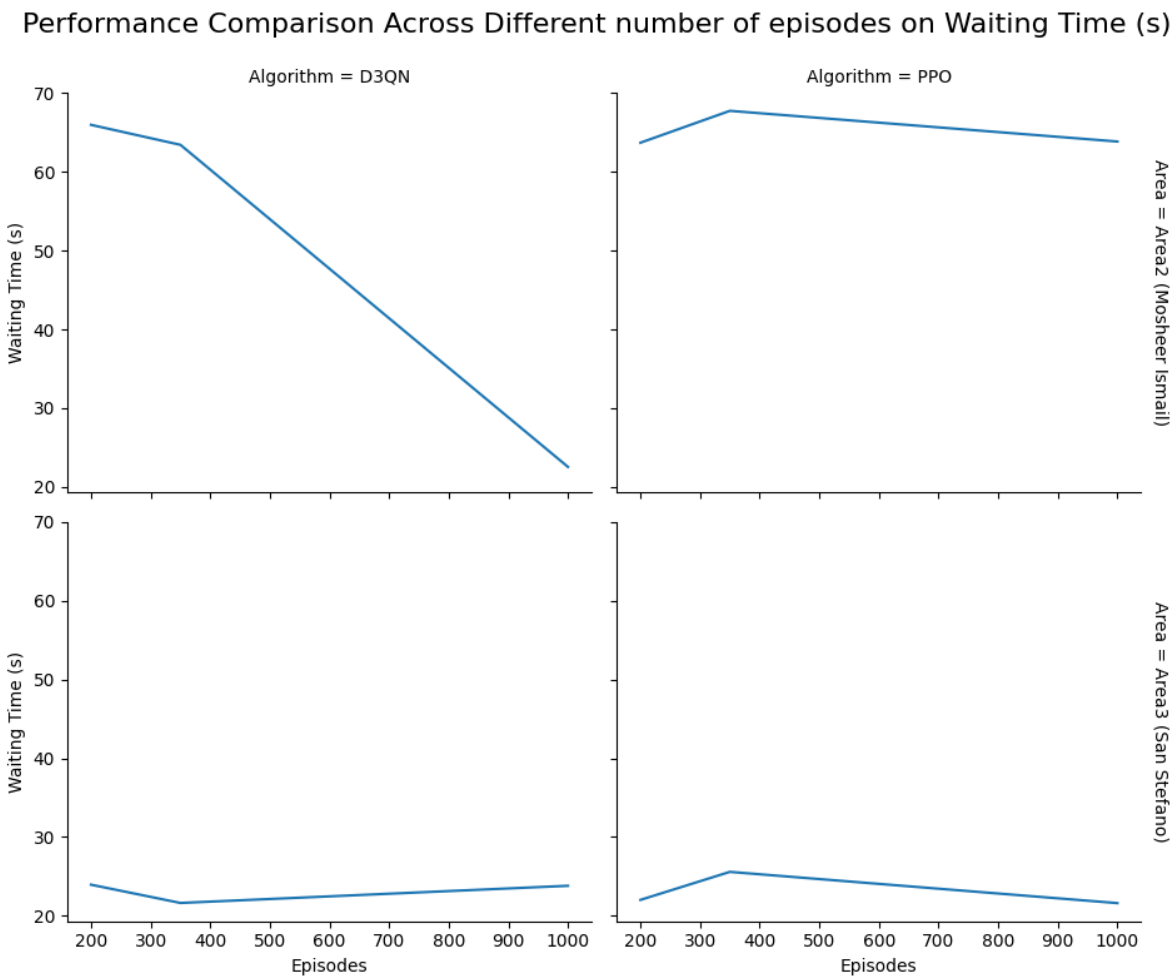


Figure 5.11 Effect of running many episodes on waiting time

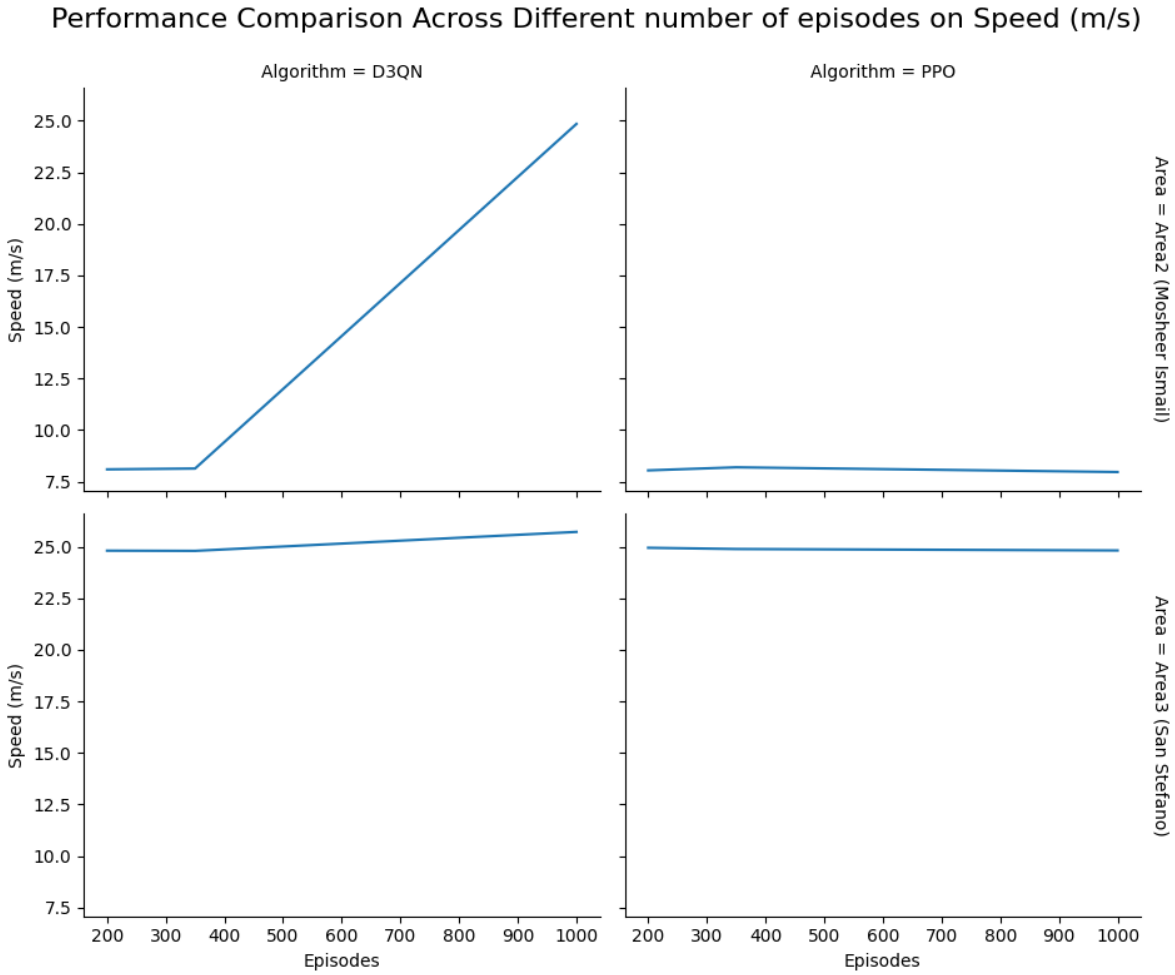


Figure 5.12 Effect of running many episodes on speed

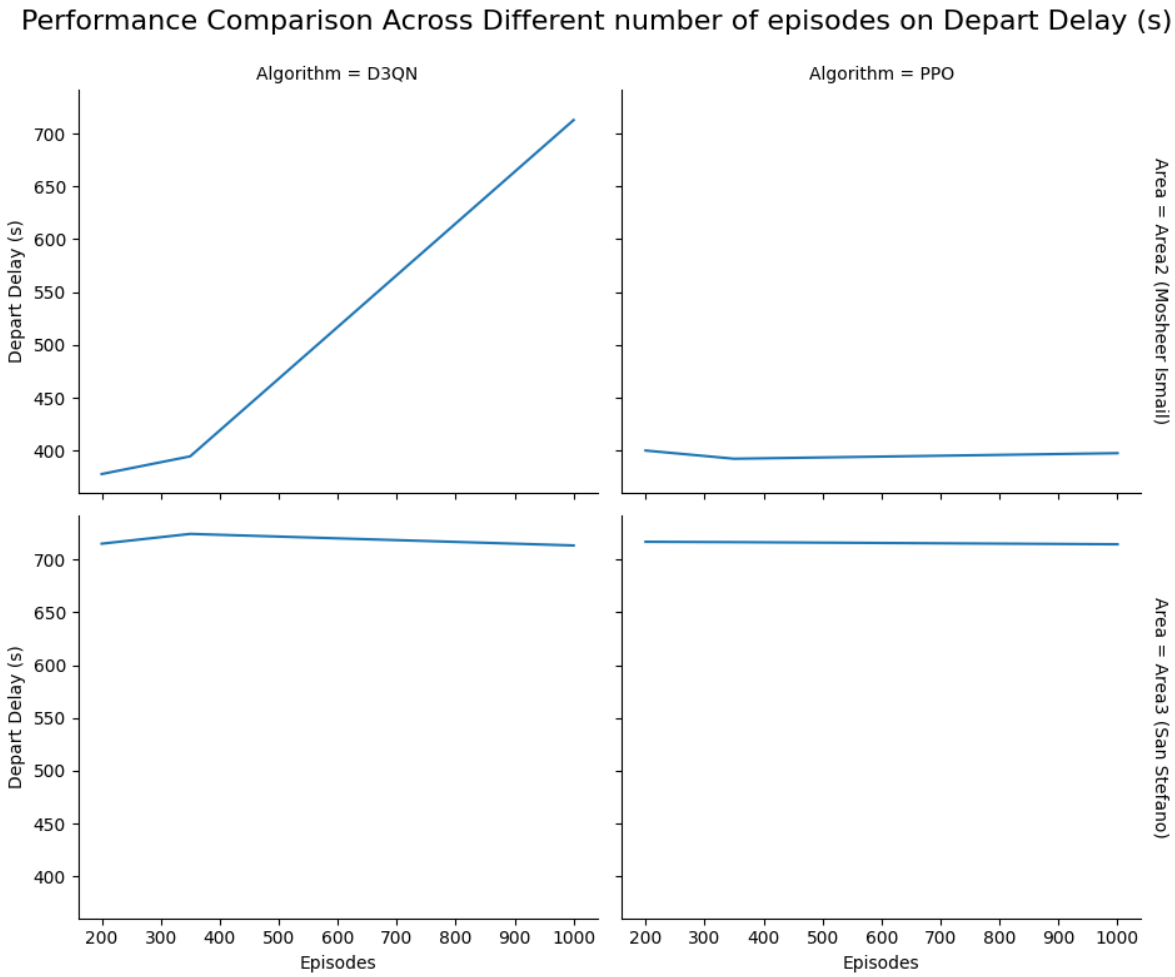


Figure 5.13 Effect of running many episodes on depart delay

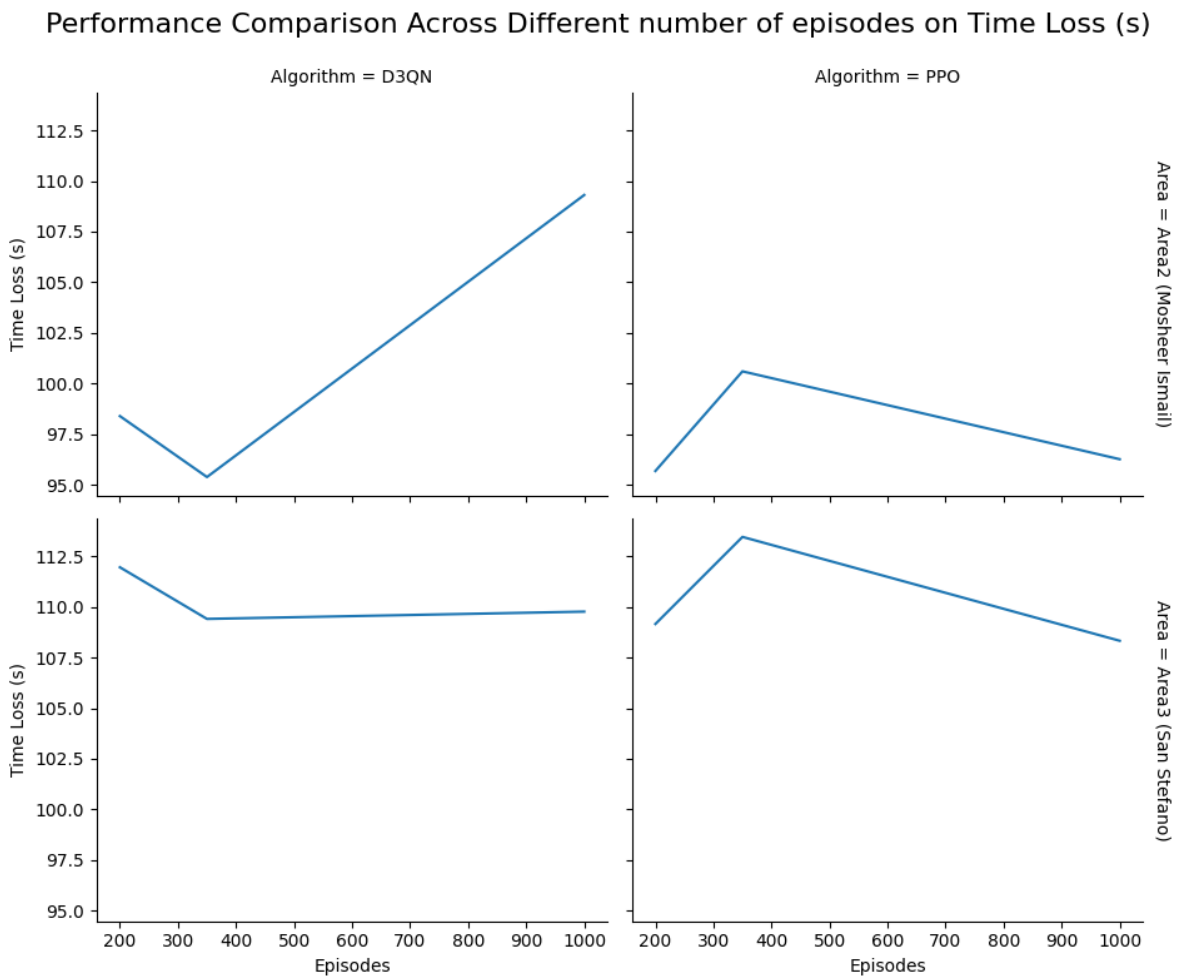


Figure 5.14 Effect of running many episodes on timeloss

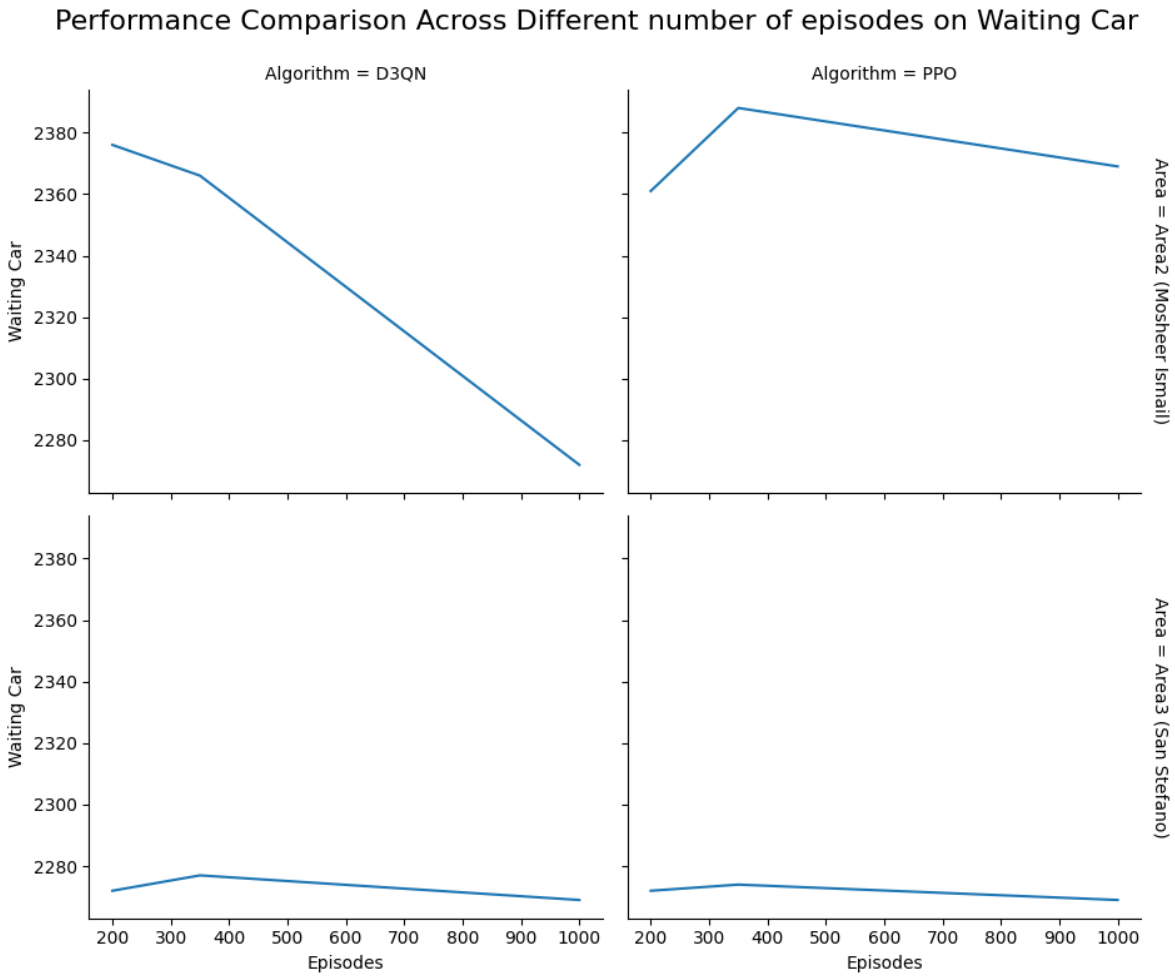


Figure 5.15 Effect of running many episodes on number of waiting cars

5.5.2.2 Behavior on different Sumo timing

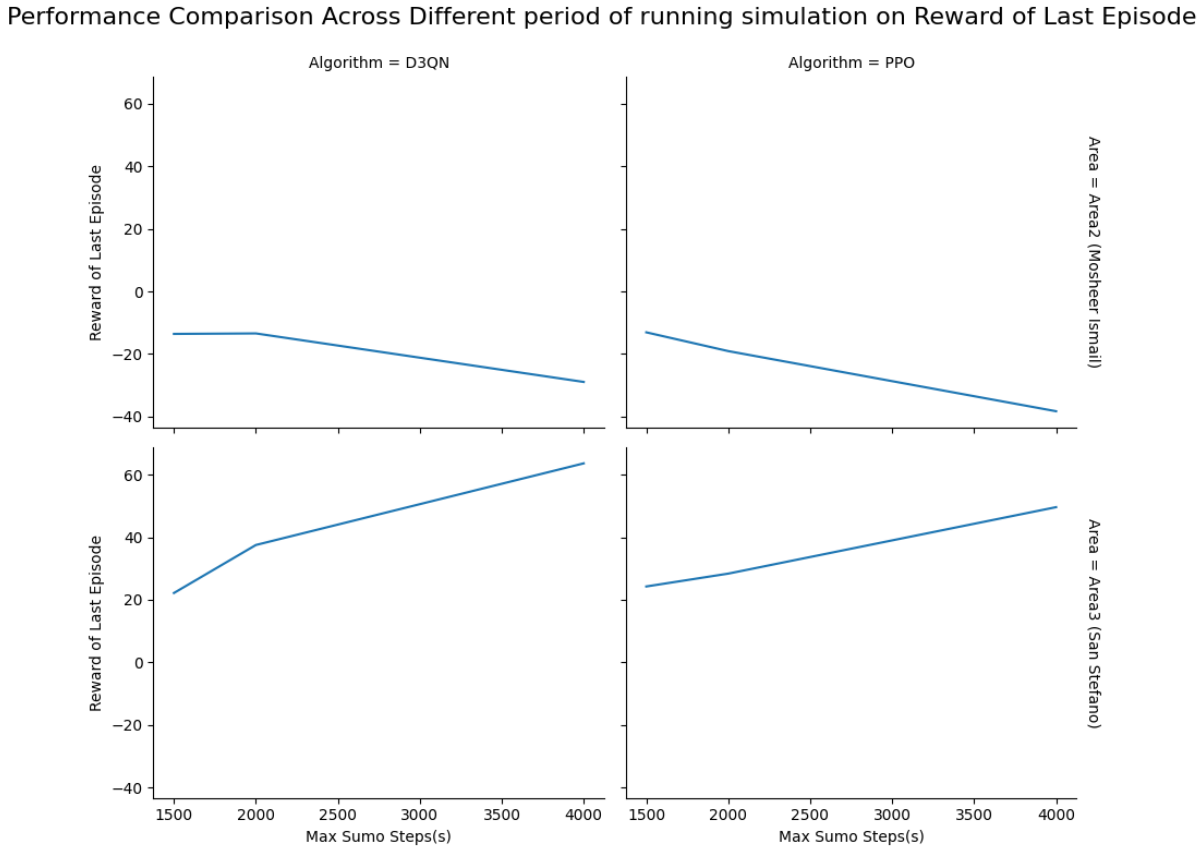


Figure 5.16 Effect of different sumo timing on reward

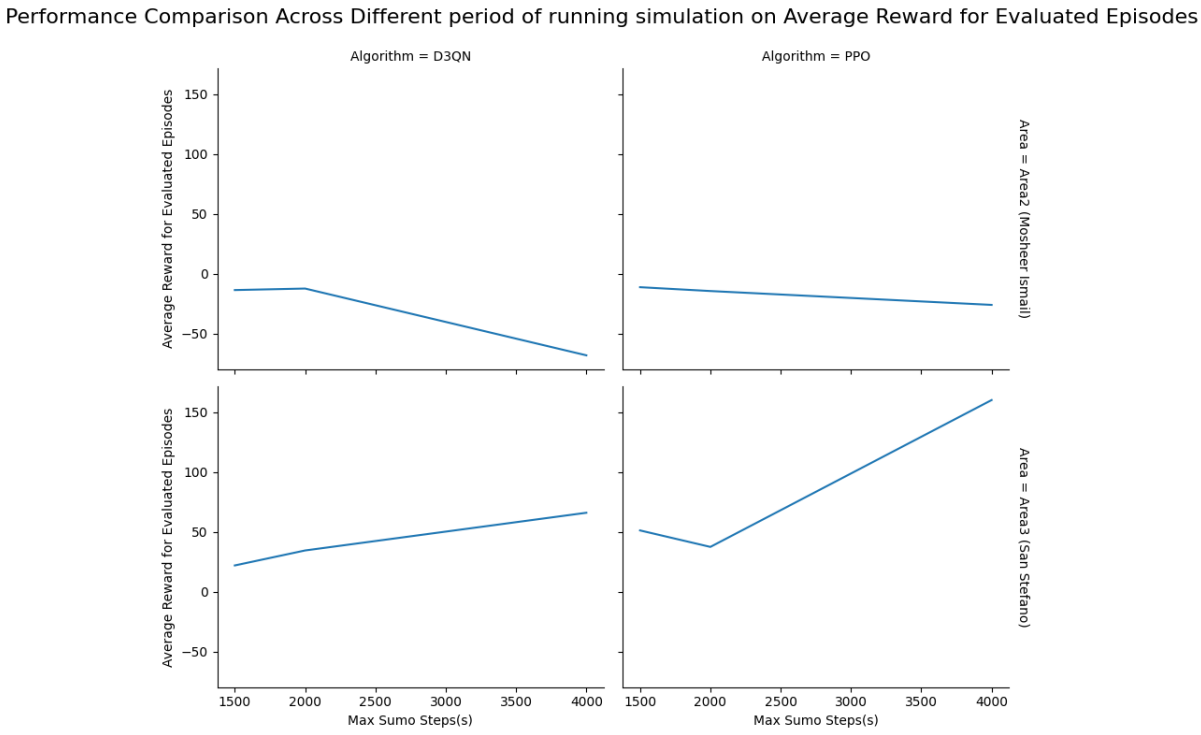


Figure 5.17 Effect of different sumo timing on evaluated reward

Figure 5.18 Effect of different sumo timing on waiting time

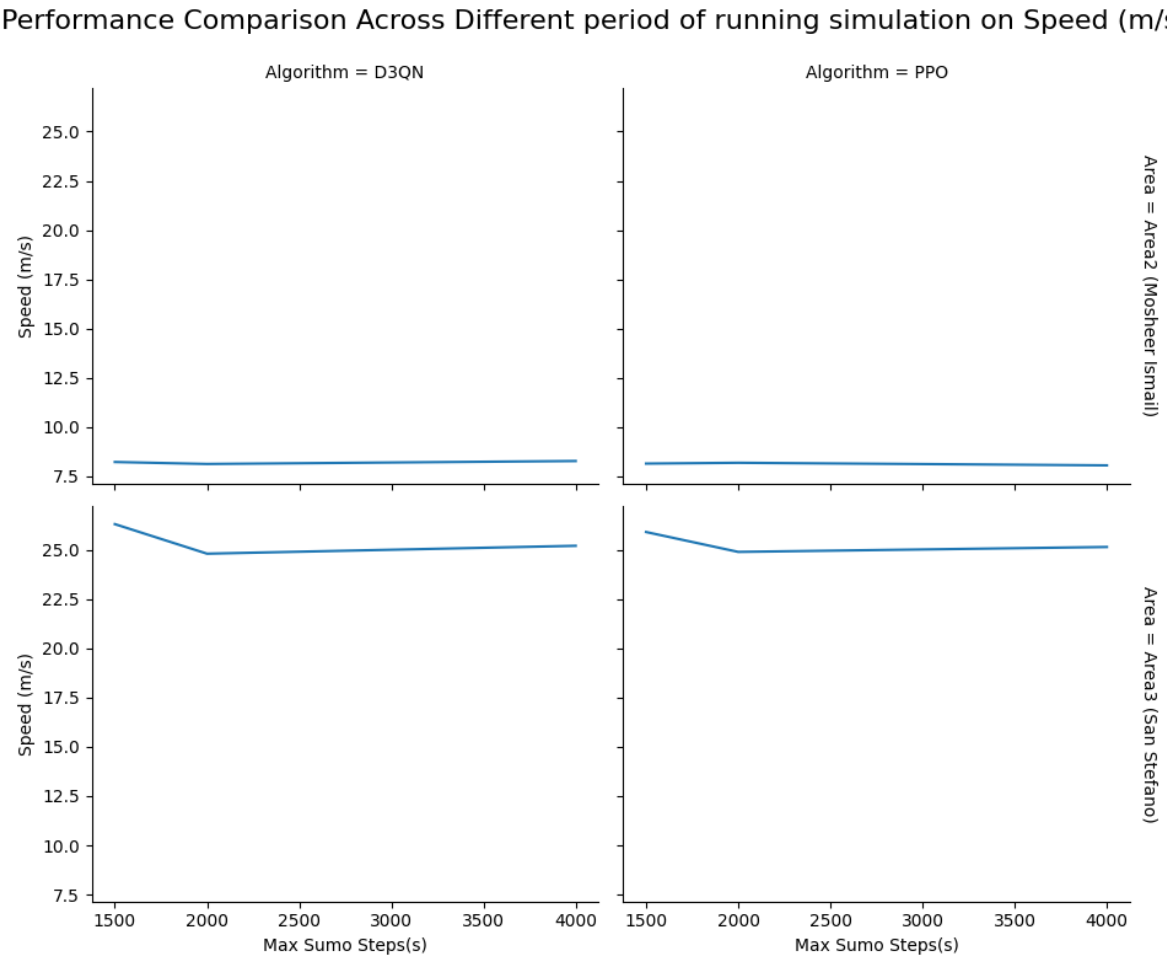


Figure 5.19 Effect of different sumo timing on speed

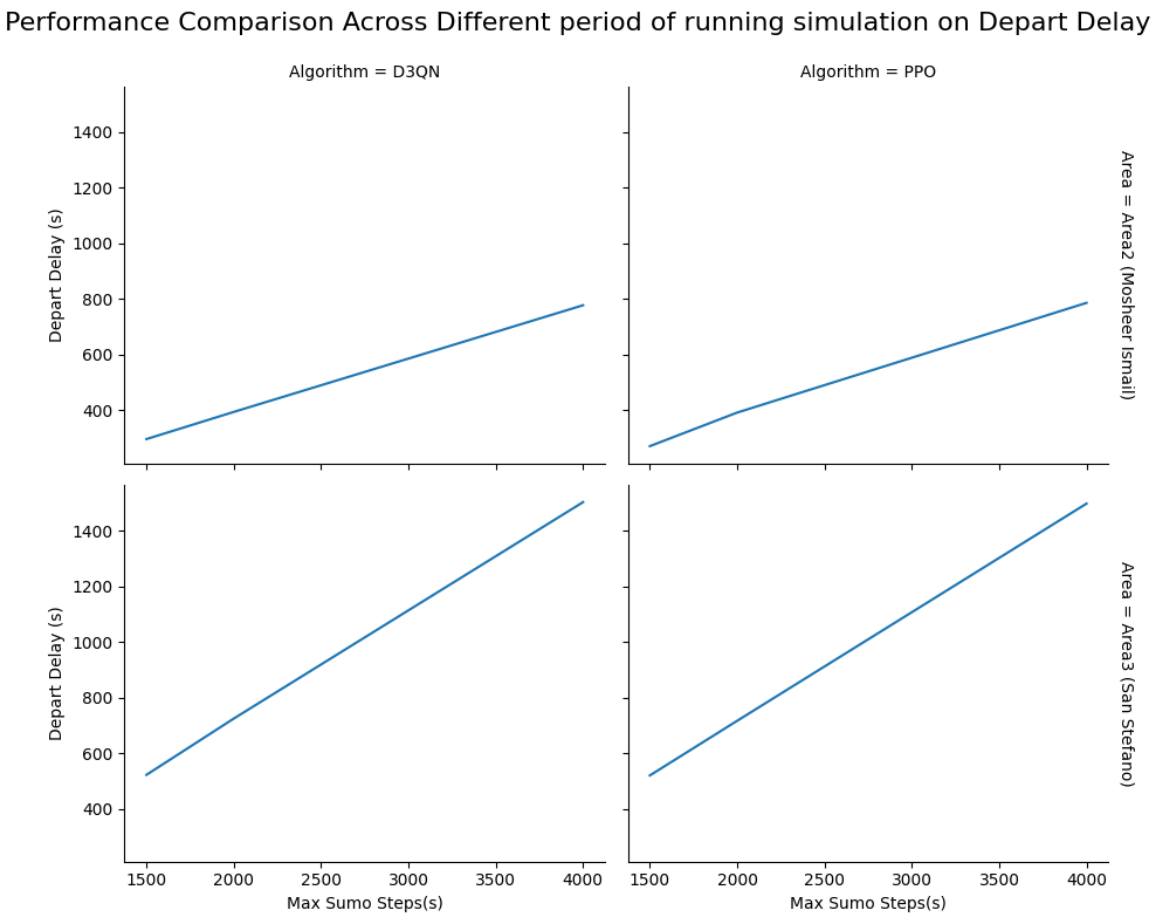


Figure 5.20 Effect of different sumo timing on depart delay

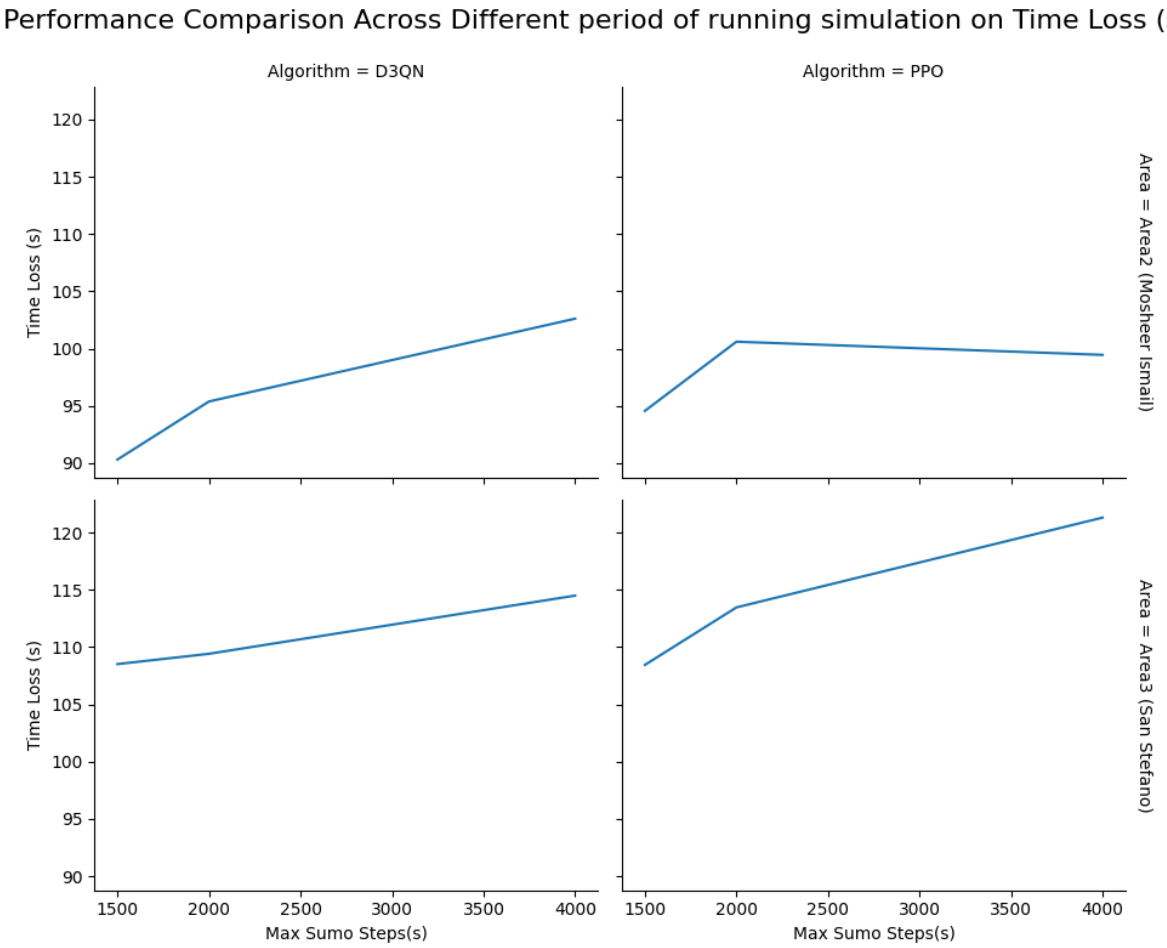


Figure 5.21 Effect of different sumo timing on timeloss

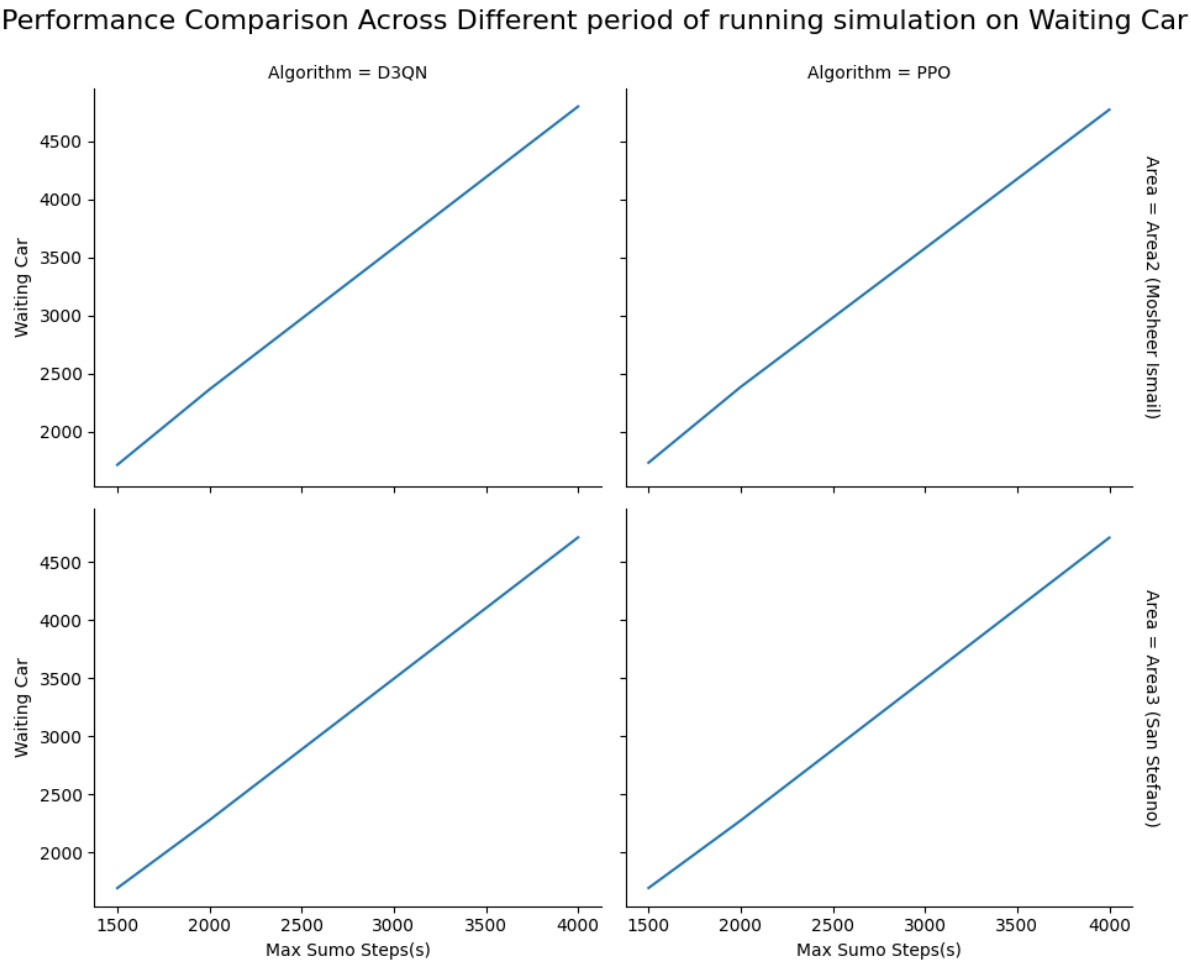


Figure 5.22 Effect of different sumo timing on number of waiting cars

5.5.2.3 Further studying on Data 2

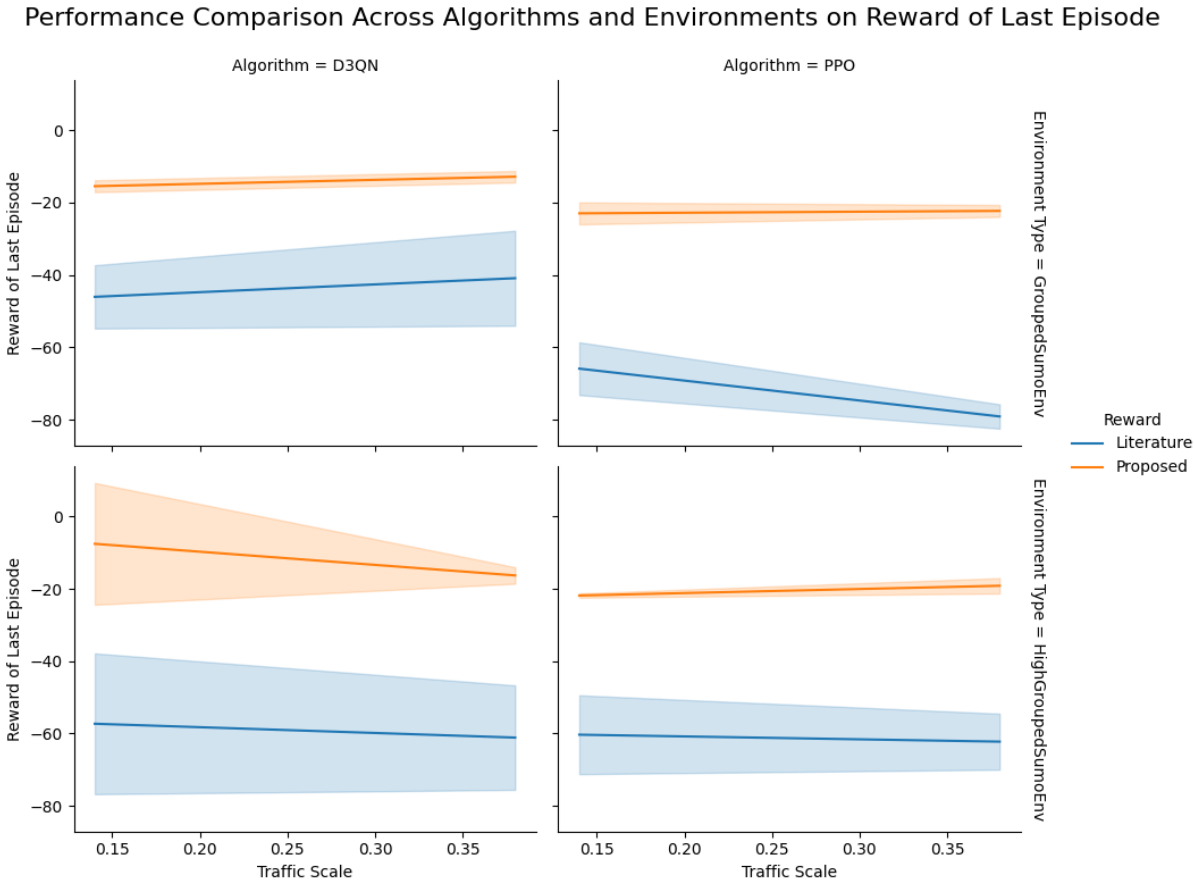


Figure 5.23 Performance comparison across algorithms and environments on reward in area 2

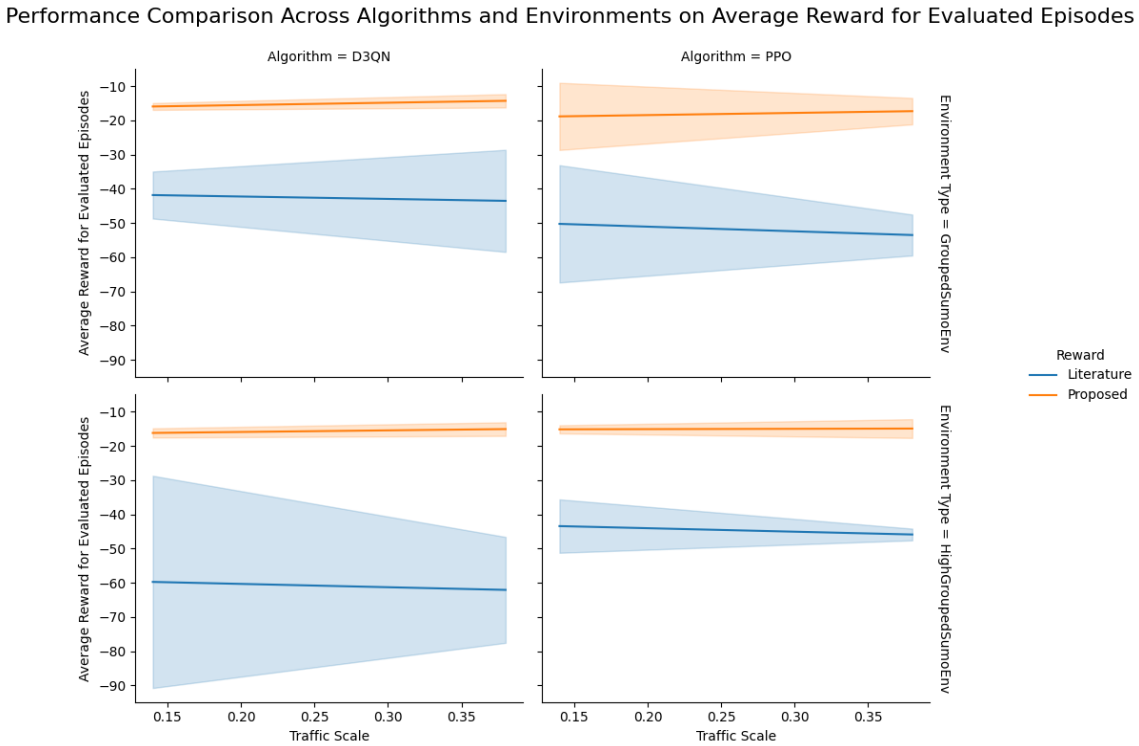


Figure 5.24 Performance comparison across algorithms and environments on evaluated reward in area 2

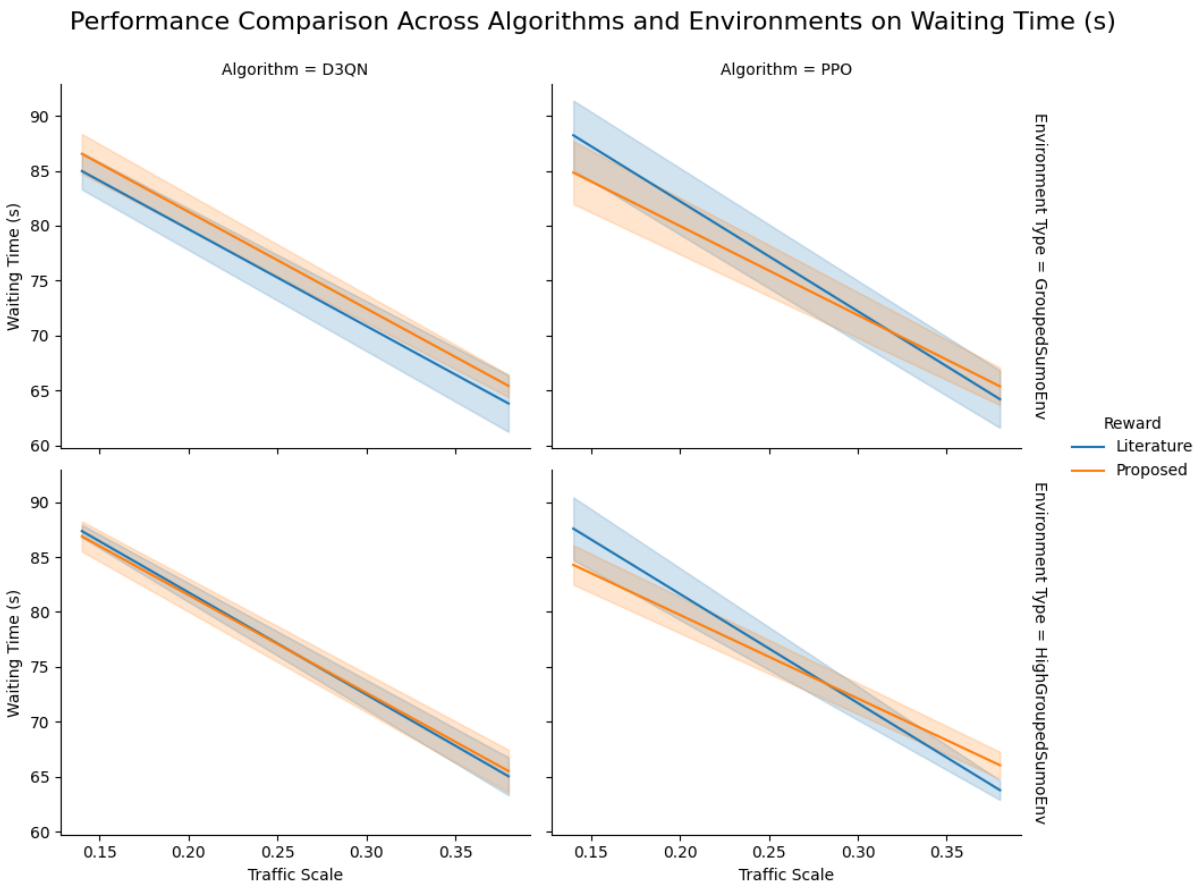


Figure 5.25 Performance comparison across algorithms and environments on waiting time in area 2

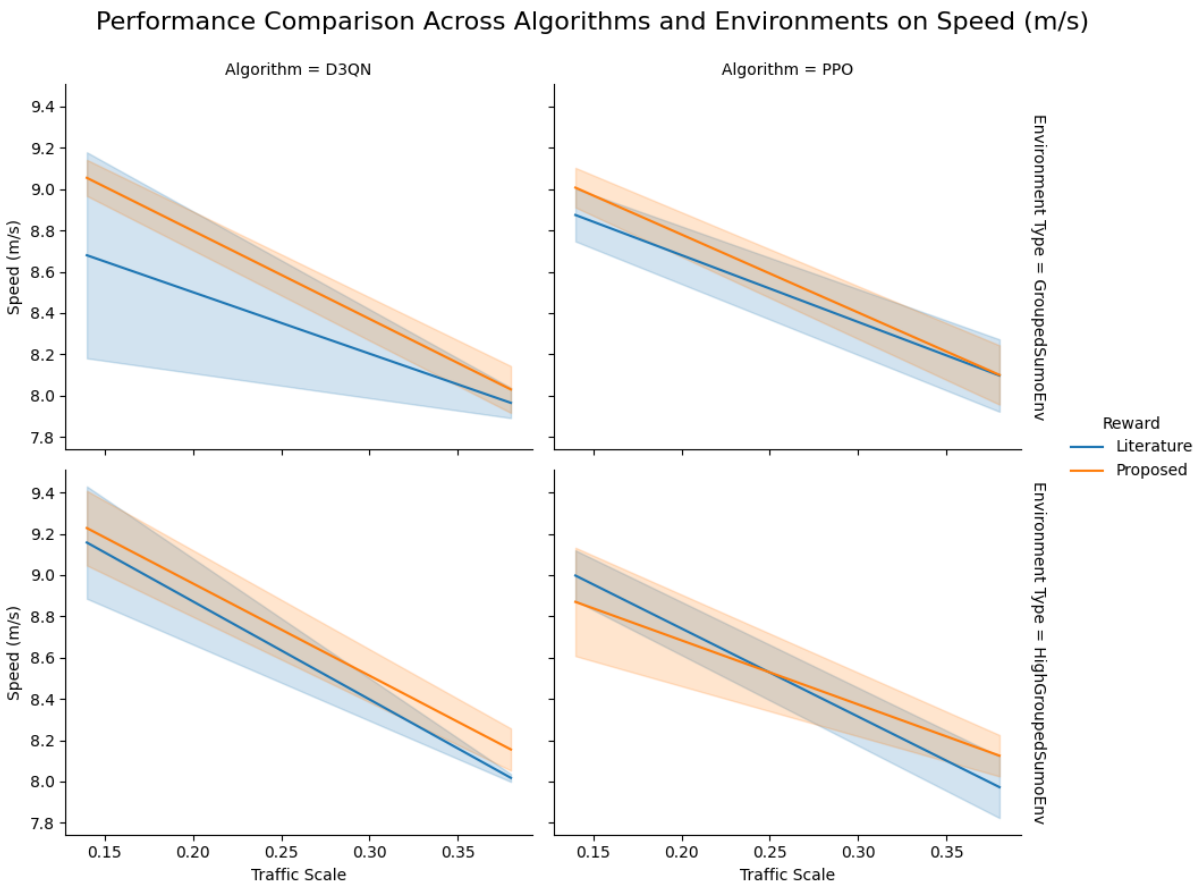


Figure 5.26 Performance comparison across algorithms and environments on speed in area 2

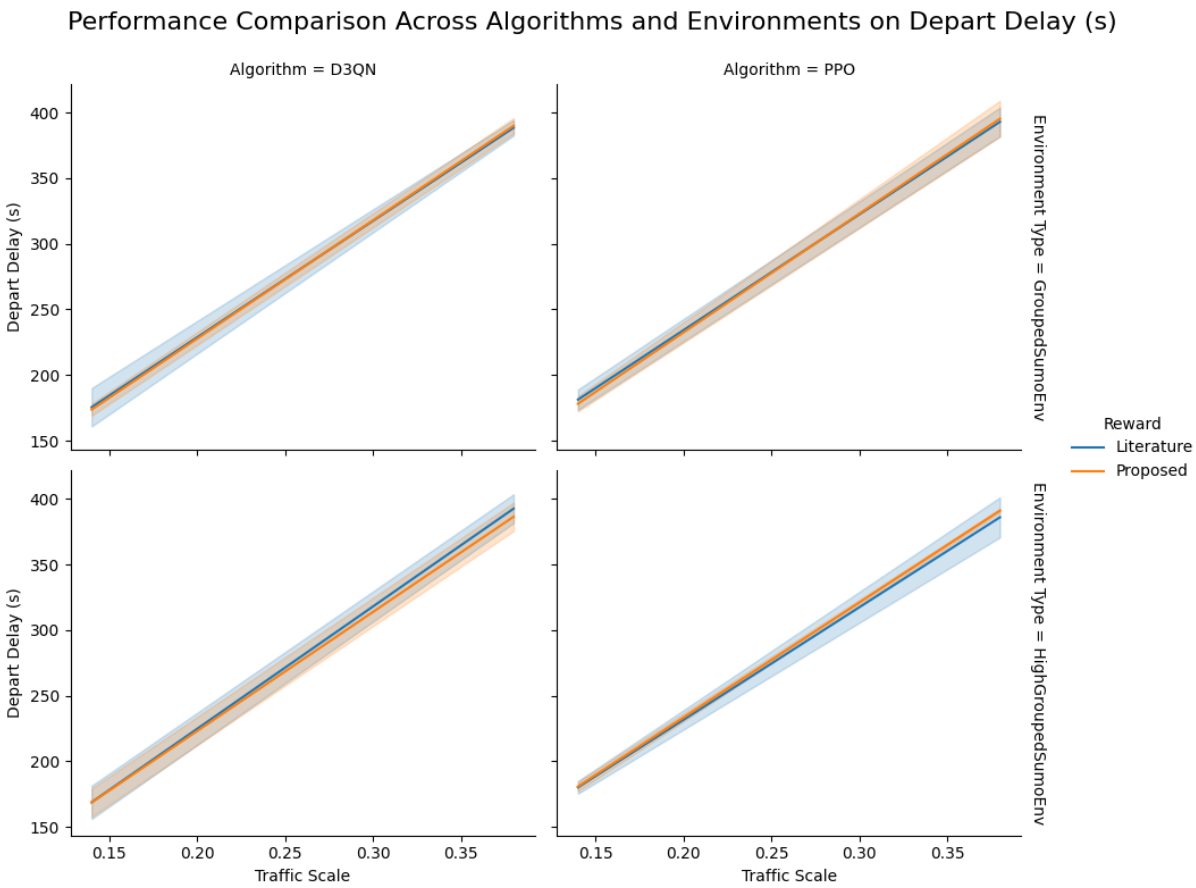


Figure 5.27 Performance comparison across algorithms and environments on depart delay in area 2

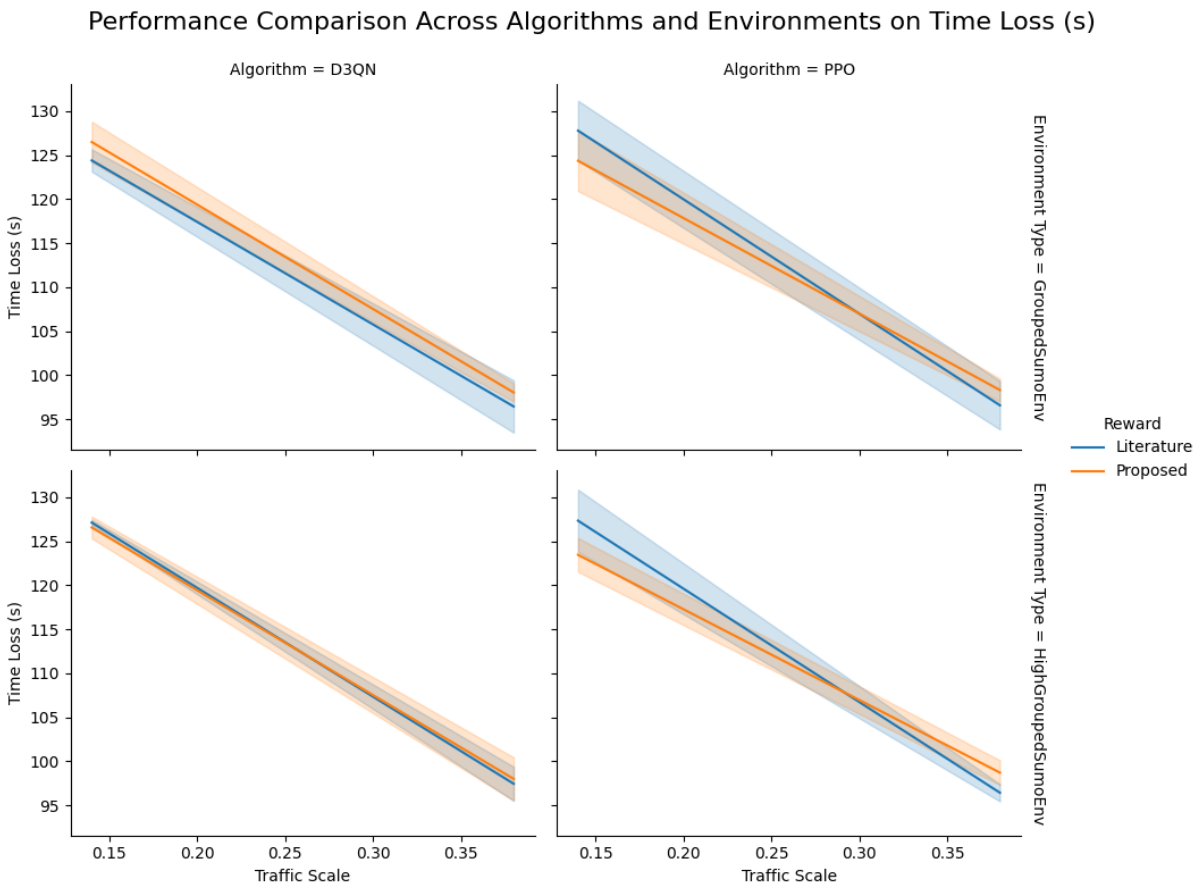


Figure 5.28 Performance comparison across algorithms and environments on timeloss in area 2

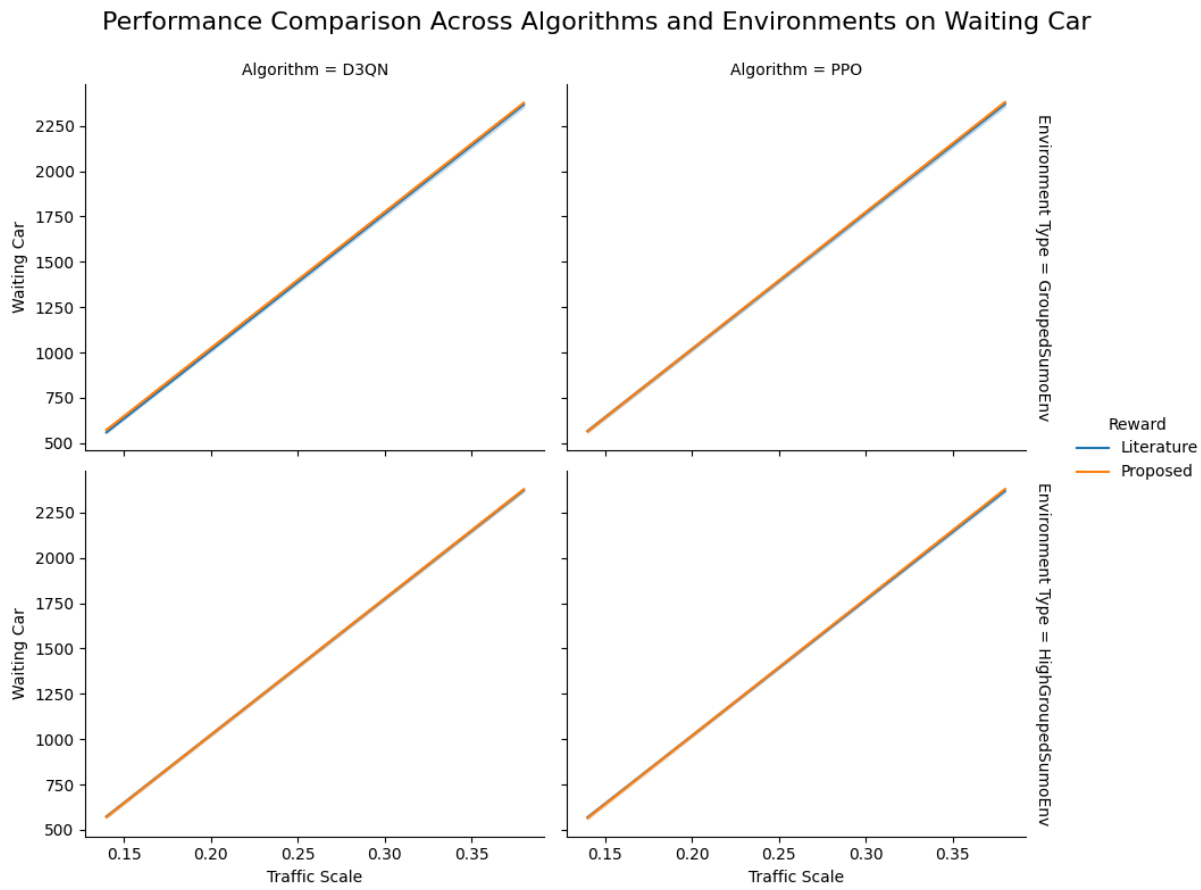


Figure 5.29 Performance comparison across algorithms and environments on number of waiting cars in area 2

5.5.2.4 Further studying on Data 3

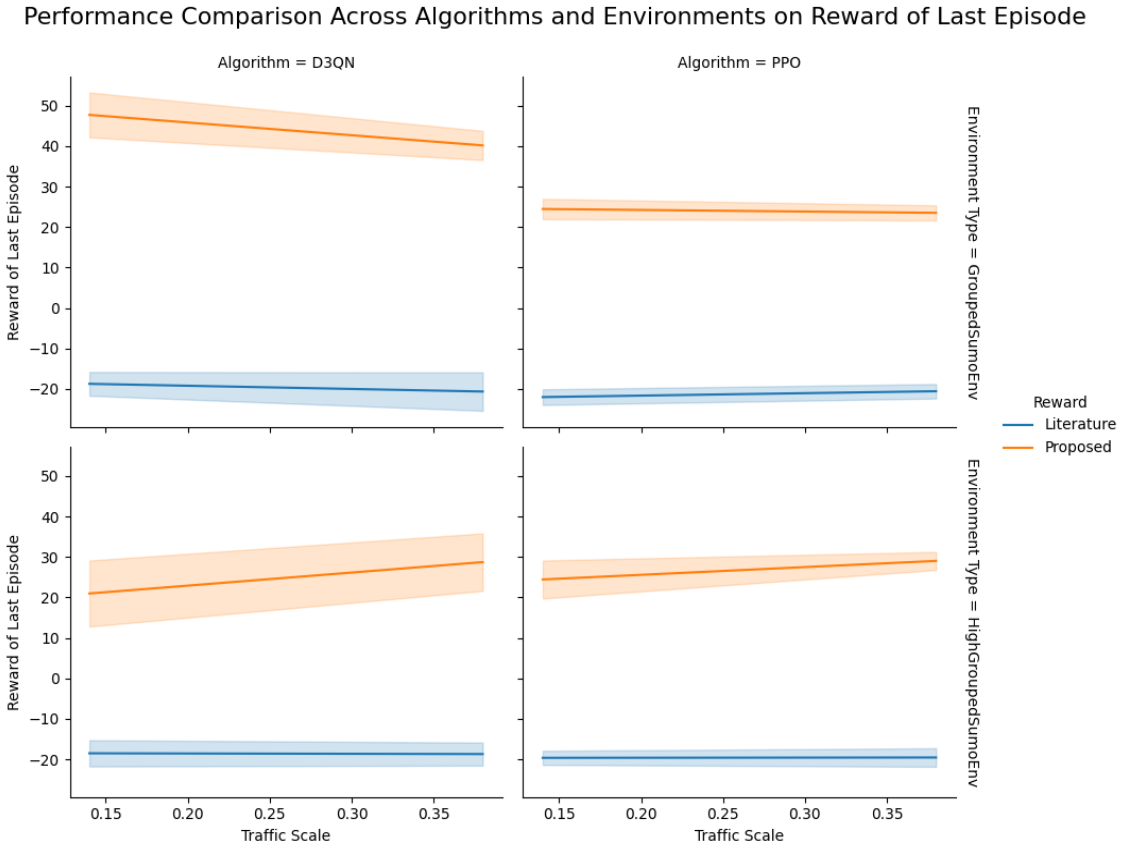


Figure 5.30 Performance comparison across algorithms and environments on reward in area 3

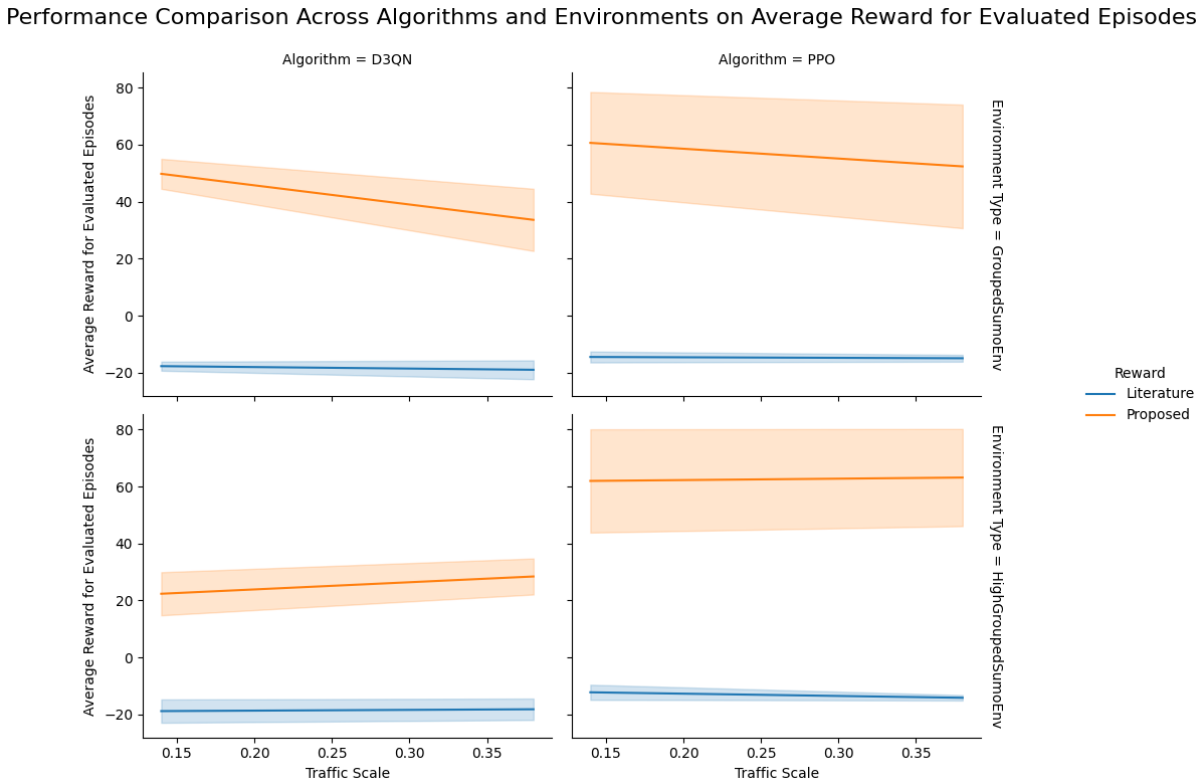


Figure 5.31 Performance comparison across algorithms and environments on evaluated reward in area 3

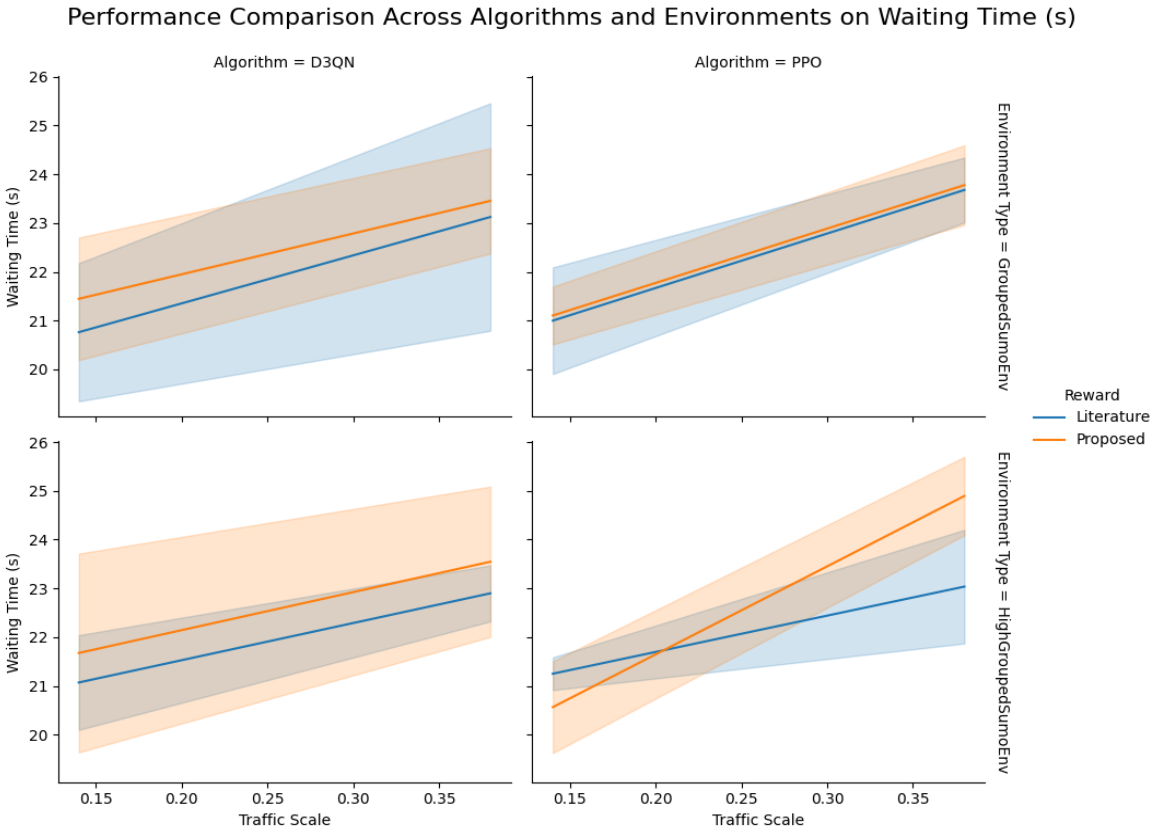


Figure 5.32 Performance comparison across algorithms and environments on waiting time in area 3

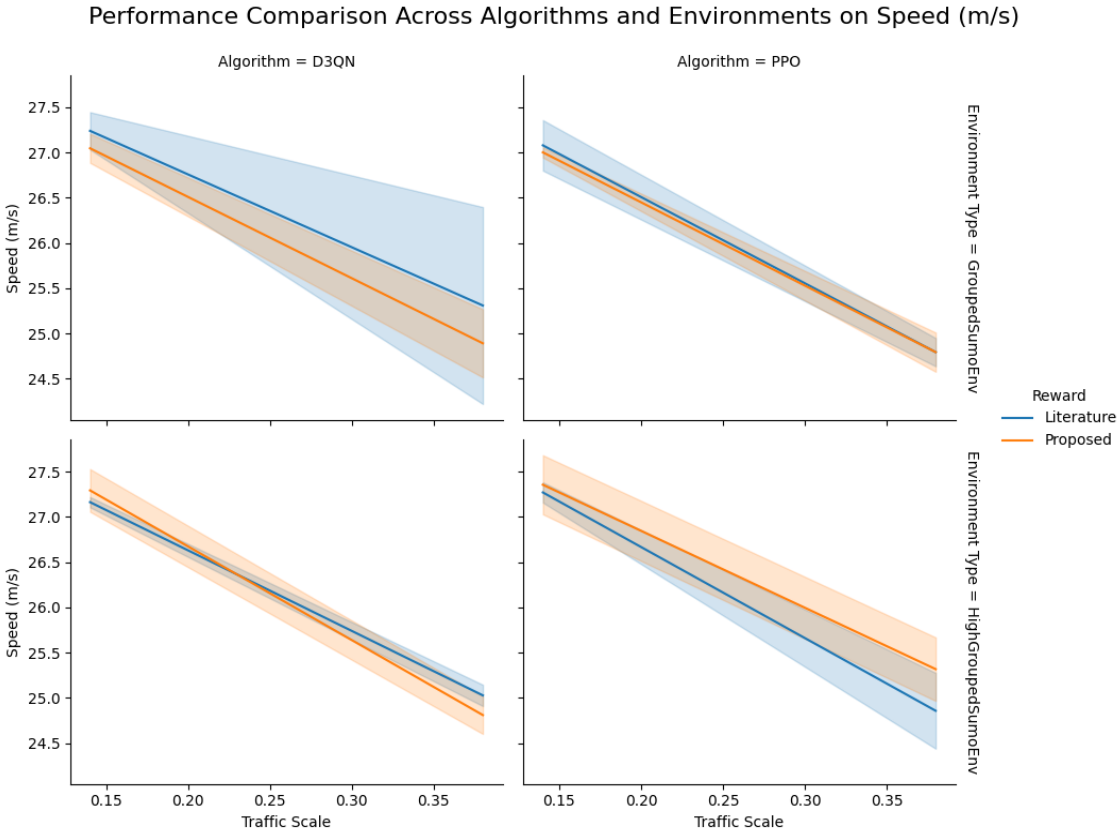


Figure 5.33 Performance comparison across algorithms and environments on speed in area 3

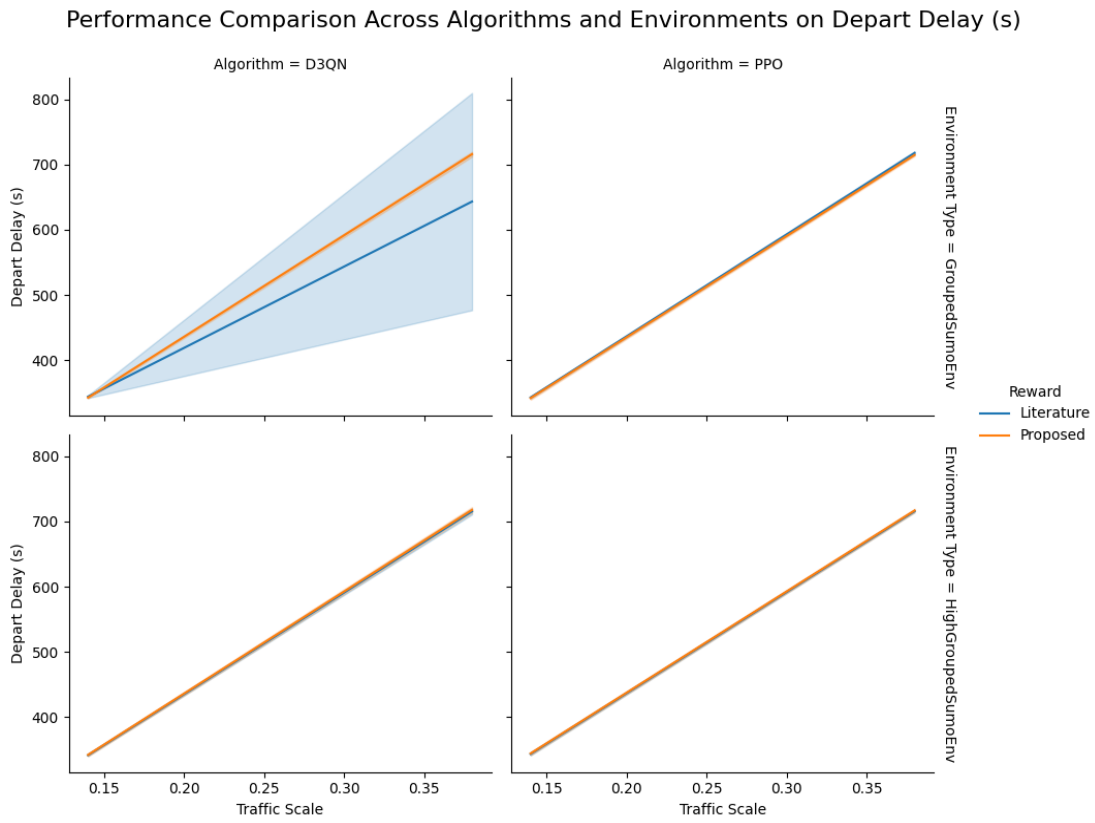


Figure 5.34 Performance comparison across algorithms and environments on depart delay in area 3

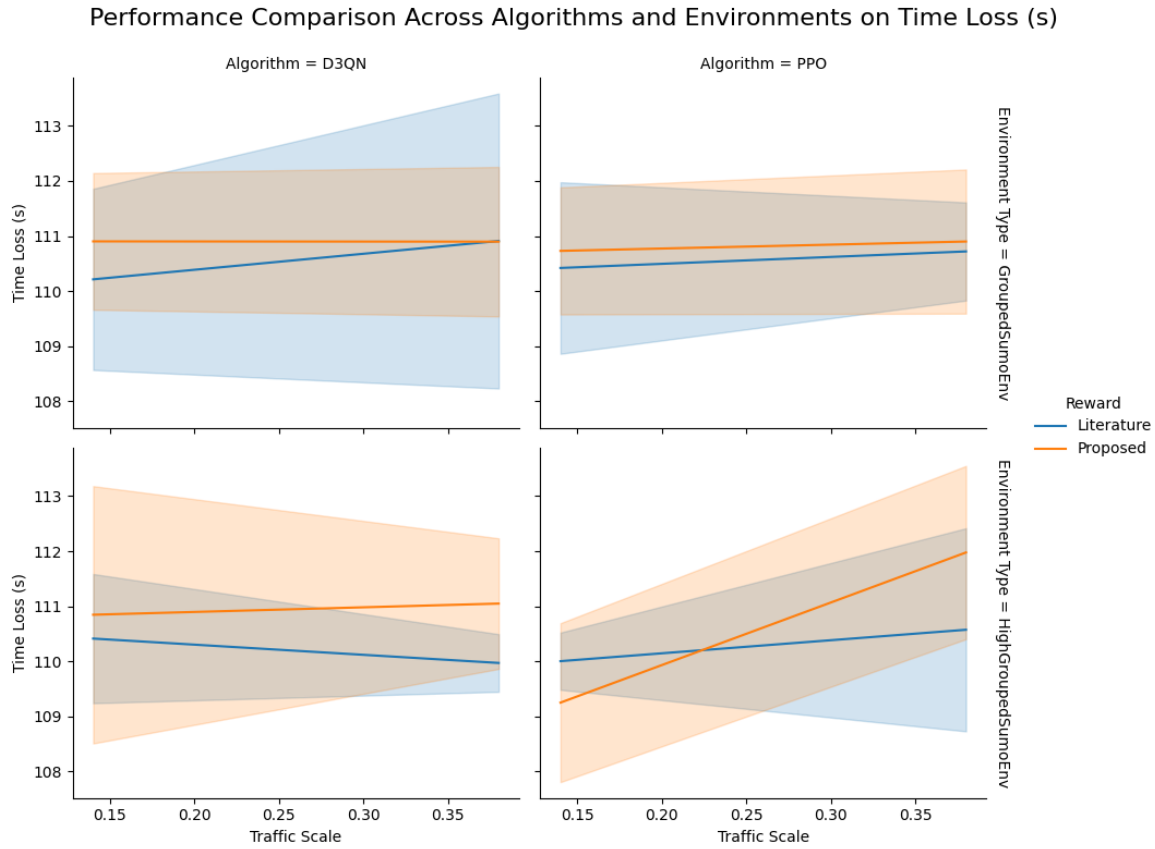


Figure 5.35 Performance comparison across algorithms and environments on timeloss in area 3

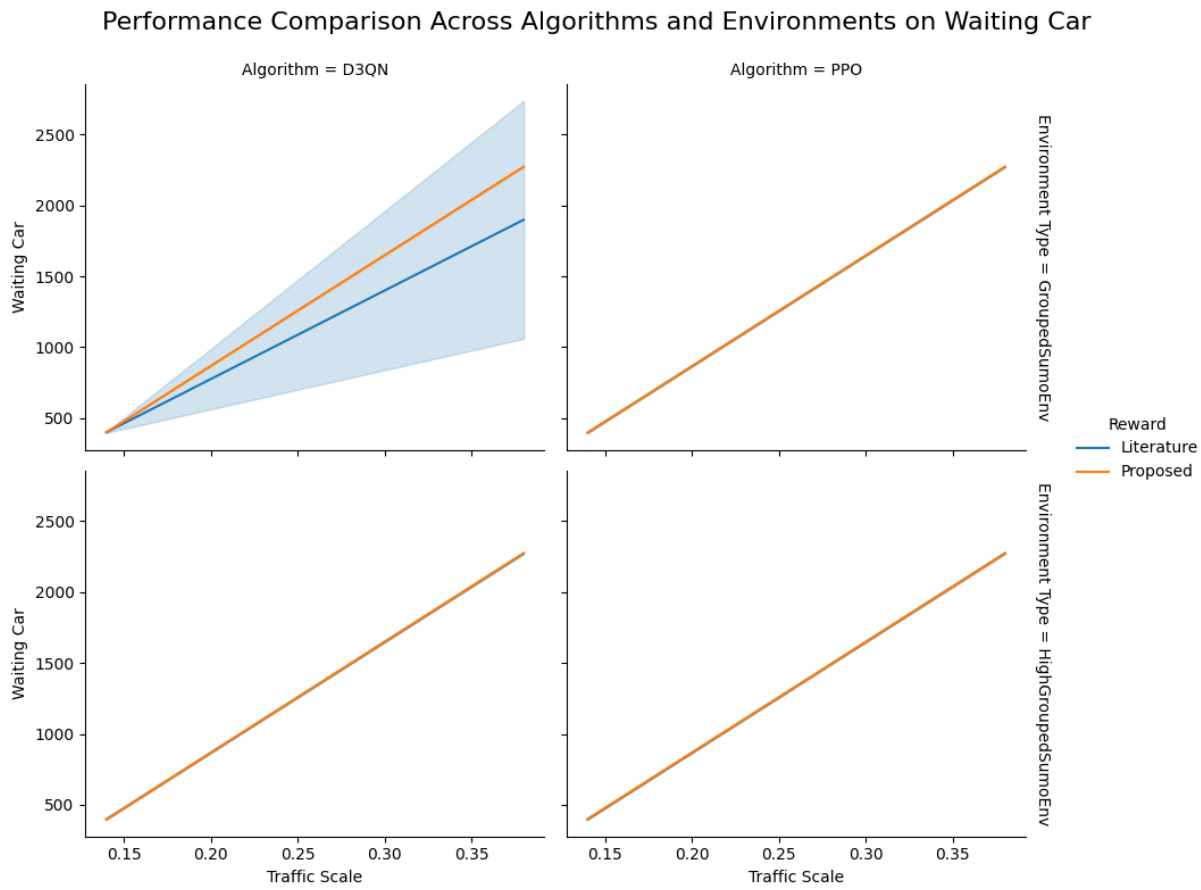


Figure 5.36 Performance comparison across algorithms and environments on number of waiting cars in area 3

5.5.2.5 Best Results

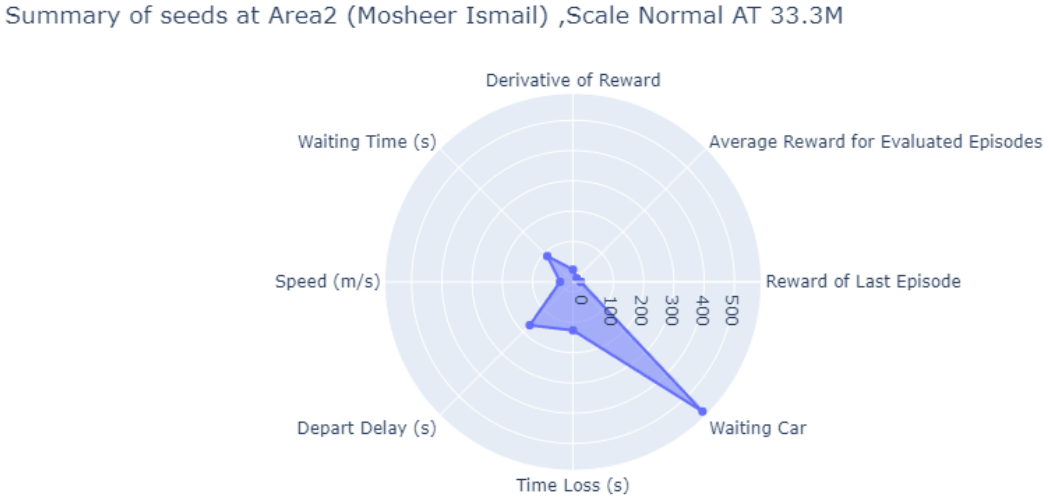


Figure 5.37 Best results at Area2 with Normal scale

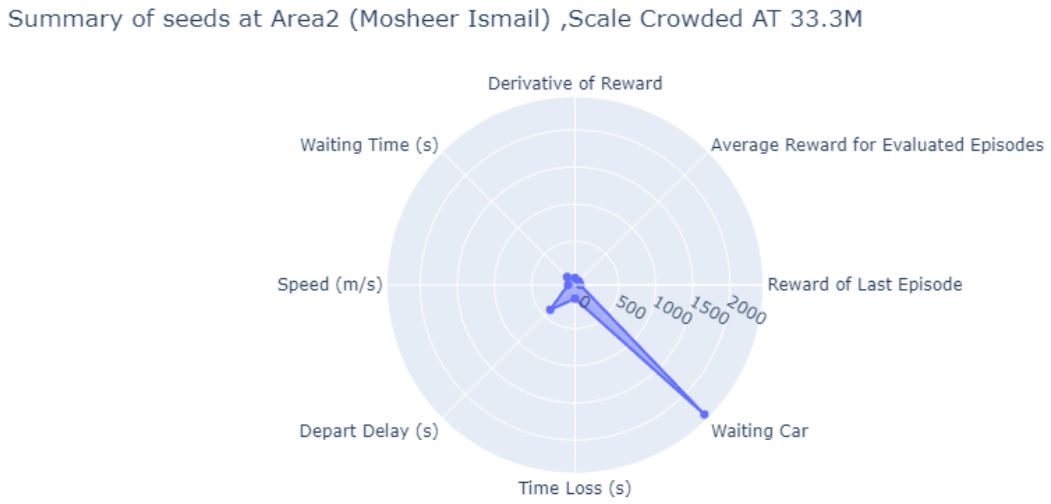


Figure 5.38 Best results at Area2 with Crowded scale

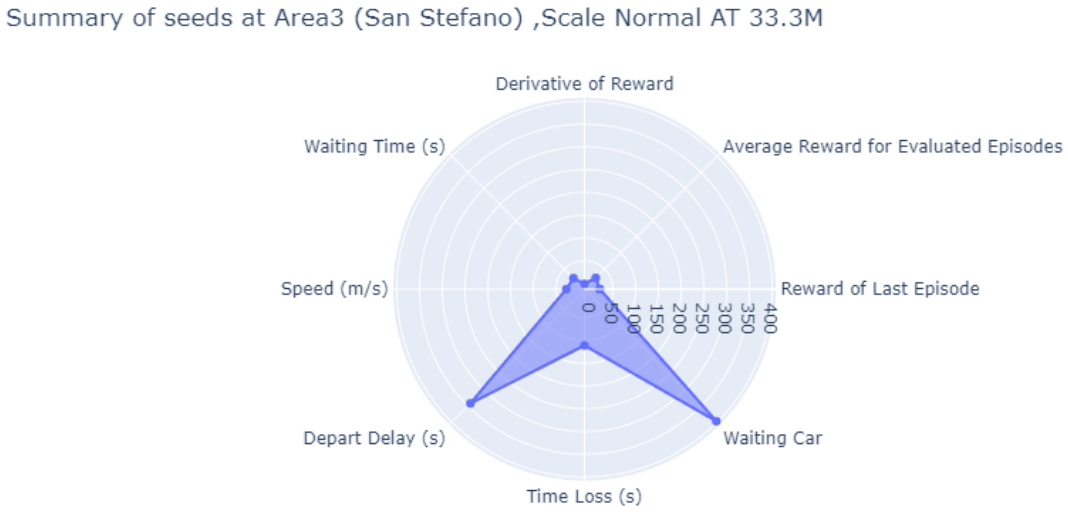


Figure 5.39 Best results at Area3 with Normal scale

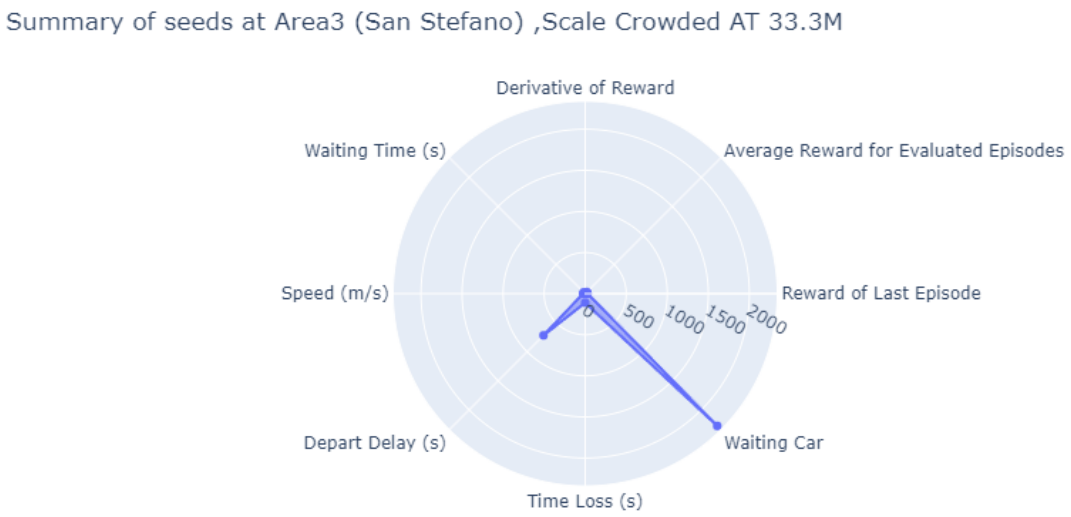


Figure 5.40 Best results at Area3 with Crowded scale

5.6 Analysis of Variance (ANOVA)

Analysis of Variance (ANOVA) is a statistical method used to determine whether there are significant differences between the means of three or more independent groups. It works by analyzing the variability within groups and between groups to assess whether the observed differences in sample means are likely due to actual effects or just random chance [41].

After the visualization was completed, the data was later analyzed using ANOVA to address the question: *Does using a high-grouped environment yield better results compared to a grouped one?*

Table 5.8 Anova results

Area	Reward	Traffic Scale	Algorithm	Reward of Last Episode	Waiting Time (s)	Speed (m/s)	Depart Delay (s)	Time Loss (s)	Waiting Car
Area3 (San Stefano)	Literature	0.14	PPO	0.211171	0.673872	0.252849	0.624318	0.631029	0.839421
Area2 (Mosheer Ismail)	Literature	0.14	PPO	0.49756	0.771623	0.214706	0.823133	0.86741	0.643226
Area3 (San Stefano)	Proposed	0.14	PPO	0.919798	0.37047	0.0773891	0.22451	0.160713	0.0398628
Area2 (Mosheer Ismail)	Proposed	0.14	PPO	0.48622	0.753739	0.363079	0.451698	0.66866	0.965445
Area3 (San Stefano)	Literature	0.38	PPO	0.321063	0.376744	0.789099	0.333174	0.891971	0.323796
Area2 (Mosheer Ismail)	Literature	0.38	PPO	0.0501503	0.766994	0.319131	0.48455	0.930467	0.95313
Area3 (San Stefano)	Proposed	0.38	PPO	0.464083	0.0984053	0.0444354	0.215673	0.332536	0.625921
Area2 (Mosheer Ismail)	Proposed	0.38	PPO	0.356003	0.546678	0.786843	0.557952	0.675678	0.765449

Area3 (San Stefano)	Literature	0.14	D3QN	0.688833	0.743306	0.494866	0.114162	0.855852	0.192344
Area2 (Mosheer Ismail)	Literature	0.14	D3QN	0.301997	0.0358339	0.143797	0.516375	0.00580131	0.00456948
Area3 (San Stefano)	Proposed	0.14	D3QN	0.00101296	0.853829	0.136805	0.805795	0.968167	0.413561
Area2 (Mosheer Ismail)	Proposed	0.14	D3QN	0.749427	0.77916	0.136595	0.426554	0.946116	0.806054
Area3 (San Stefano)	Literature	0.38	D3QN	0.732422	0.858222	0.625868	0.420143	0.517794	0.409174
Area2 (Mosheer Ismail)	Literature	0.38	D3QN	0.134828	0.460697	0.214563	0.53454	0.585142	0.431086
Area3 (San Stefano)	Proposed	0.38	D3QN	0.442762	0.924835	0.714651	0.543875	0.871118	0.562764
Area2 (Mosheer Ismail)	Proposed	0.38	D3QN	0.581691	0.933766	0.152576	0.578887	0.992819	0.877516

With a significance level of 97%, most results were found not to show significant improvement. This was initially observed from the plot. The lack of improvement may be attributed to the nature of the road being in a single direction.

5.7 Discussion of results

5.7.1 Discussion of the results of Benechmark

- The proposed environment was observed to outperform both basic SUMO and the reference project in many scenarios, and in most cases overall.
- The performance improvement may stem from multiple low-level design choices and default configurations embedded within Stable-Baselines3. These include critical hyperparameters such as target_update_interval, train_freq, and learning_starts, which were not explicitly set in the earlier implementation. Furthermore, the transition to PyTorch and potential architectural differences in the environment or training loop could also contribute. Reinforcement learning is inherently stochastic and highly sensitive to these configurations, meaning even subtle differences can lead to significant changes in learning dynamics and final performance.
- Certain performance metrics were found to be highly sensitive to random seeds, indicating variability across runs [42].

5.7.2 Discussion of the results of Environmental Experiments

Effect of Episode Count Increasing the number of episodes was observed to improve several performance metrics—specifically, waiting cars, waiting time, speed, and reward—particularly in Dataset 2 With D3QN. A less significant impact was noted in Dataset 3. However, results remained noisy, as the experiments were conducted with a single random seed due to time constraints. The configuration used included: the proposed reward function, a traffic scale of 0.38, 500 SUMO steps, the `highgrouped` level, and seed 0. These conditions are candidates for future work with more comprehensive seeding and evaluation.

Effect of SUMO Steps An increase in the number of SUMO steps logically resulted in higher waiting time and time loss, yet the reinforcement learning algorithm was able to adapt and achieve better reward performance. The derivative of average waiting time supported this learning behavior. Again, results showed high variability due to the single-seed constraint. This configuration involved: the proposed reward function, 1000 episodes, seed 0, traffic scale of 0.38, and `highgrouped` setting. Further work should explore multi-seed evaluations to mitigate variance.

Reward Function Comparison The literature-defined reward achieved better results in Dataset 3, although the differences were not substantial. In Dataset 2, results were close between both reward functions.

Seed Variability As typical in reinforcement learning, high variance was observed between different random seeds, underlining the need for multi-seed validation [42].

Learning Efficiency The proposed reward function facilitated better learning, especially in Dataset 3, where positive rewards were obtained. In contrast, Dataset 2 showed large reward derivatives across episodes and often produced negative rewards. Notably, the reference paper for the proposed reward also reported initial negative rewards, eventually improving performance by incorporating additional road-related parameters (e.g., neighboring roads, time-spatial features).

Algorithm Comparison Both algorithms converged to similar final metrics. However, D3QN exhibited greater stability, both in terms of reward derivative and training-test performance consistency. The reduced noise in training plots further supports this observation.

Effect of Grouping The grouping configuration highgrouped did not yield significant differences, possibly due to the structure of Egyptian roads, where lanes often flow in the same direction.

Difference between Areas

- Better performance was achieved in Dataset 3 San Stefano, where reward values were generally non-negative. This may be attributed to differences in road infrastructure, which affected the number of cars passing through.

- Future Work is to address geographical differences in simulation difficulty in data 2 (Mosheer Ismail). It was observed that Alexandrian roads posed greater challenges due to narrower lanes and lower vehicle speeds (e.g., \~7 m/s in AlST vs. \~18 m/s in LuST), which made the simulation both harder and slower. This should be considered in engineering applications aiming for realistic traffic modeling.

Overall Results

- The overall waiting time was achieved to be very good (20:90 s for half of hour), and for speed was max possible speed
- High number of waiting cars, with small waiting time and timeloss indicate many waited but with fairness which is wanted
- Depart delay was a larger, need further learning (can be 1.5M :13.3M)

5.7.3 Best Observed Results

Best Algorithm D3QN

Best Configuration `highgrouped` (fast, no additional grouping needed)

Best Reward Function Literature-defined reward

5.8 Conclusion

In this chapter, the environmental design was discussed, including the data, reward structure, state representation, and action space. The chapter was then dived into the presentation of experiment and results. The next chapter will have final summary and future work.

CHAPTER 6 : CONCLUSION AND FUTURE WORK

In the previous chapter, the environment design, the data used, the state space, the action space, the reward structure, and the chosen model were presented. The selection of hyperparameters and the detailed environment setup were then explained. This was followed by the presentation of the environment's results and the implementation of the system prototype.

In this chapter, the final conclusions of the report will be presented, along with an outline of potential directions for future work.

6.1 Conclusion

Urban traffic management is recognized as a critical issue affecting daily life and energy consumption. Traditional systems are often challenged by inefficient energy use and suboptimal traffic flow. In this work, a system was designed using reinforcement learning to optimize traffic, with awareness of emergencies and accidents. Low waiting times, high speeds, reliability, and real-time performance were achieved.

6.2 Future work

According to section 3.4 Future work:

- The model accuracy will be improved, and a comparison will be made with the results from the official paper.
- The data problem, as described in [5.7.2 Discussion of the results of Environmental Experiments] will be addressed.
- Training will be conducted on episodes initialized with different random seeds and on more experiments to improve robustness and generalization as described in [5.7.2 Discussion of the results of Environmental Experiments].
- The process of identifying and selecting the signal that aligns with the user's direction will be further investigated as described in [4.5.4 Mobile implementation].
- Signal duration and delay adjustments will be automated to reduce manual configuration and improve responsiveness as described in [4.5.4 Mobile implementation].
- User experience in the mobile app will be improved by notifying users, helping them select roads, predicting better routes, and gathering feedback [9],[10], [12].
- An appropriate license will be added to the project to define usage and distribution terms clearly as described in [4.8 Open-Source Availability].
- Real-life performance will be captured using real data as described in [4.5.5 Scaling the prototype].
- Online reinforcement learning (RL) will be implemented and optimized with new data as described in [4.5.5 Scaling the prototype].
- The model will be optimized for deployment on a Raspberry Pi, and data collection will be optimized using GPS and inductive loops.
- Scaling to dependent multi-agent systems will be implemented.

- Consideration will be given to state, temporal, spatial, and nearby information as a new trend [6].
- Cybersecurity measures will be considered as described in [4.5.2 Basic implemented security].
- Control will be established for Egypt's GPS to identify traffic violators [15].

REFERENCES

[1] K. Sabra, M. Moustafa, et al., "Intelligent Traffic Analysis," B. Sc. Degree, Graduation Project, Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, 2023.

[2] M. M. Dhanvijay and S. C. Patil, "Energy efficient deep reinforcement learning approach to control the traffic flow in IoT networks for smart city," J. Ambient Intell. Human Comput., vol. 15, pp. 3945–3961, Dec. 2024, doi: 10.1007/s12652-024-04869-w.

[3] TWI, "What is simulation?," TWI Global. [Online]. Available: https://www.twi-global.com/technical-knowledge/faqs/faq-what-is-simulation#WhatDoesitMean. [Accessed: Dec. 1, 2024].

[4] "Gymnasium Documentation," Gymnasium. https://gymnasium.farama.org/ (accessed Jan. 20, 2025).

[5] DLR, "SUMO Tutorials," SUMO Documentation, Available: https://sumo.dlr.de/docs/Tutorials/, Accessed: Nov. 18, 2024

[6] GeeksforGeeks, "Machine Learning," GeeksforGeeks, [online]. Available: https://www.geeksforgeeks.org/machine-learning/. [Accessed: 1-Dec-2024].

[7] Cornell University, "Heuristic Algorithms," accessed Dec. 20, 2024. [online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Heuristic_algorithms&utm_source=chatgpt.com

[8] Coursera, *Introduction to Machine Learning Specialization*, Available, Accessed: Nov. 18, 2024

[9] IBM, "What are decision trees?" IBM, [online]. Available: https://www.ibm.com/topics/decision-trees. [Accessed: 1-Dec-2024].

[10] ResearchGate, "A single neuron model," accessed Dec. 20, 2024. [online]. Available: https://www.researchgate.net/figure/A-single-neuron-model_fig1_317428704

[11] Hugging Face, *Deep Reinforcement Learning Course*, Hugging Face, Available, Accessed: Nov. 18, 2024.

[12] M. J. Kearns and D. K. DeHaan, "A Comparison of Algorithms for Learning to Classify," *Machine Learning*, vol. 6, no. 1, pp. 3-35, 1991, Available, Accessed: Nov. 18, 2024.

[13] Hugging Face, "Offline and Online Reinforcement Learning," Hugging Face, Available, Accessed: Nov. 18, 2024.

[14] H. Liu, J. Niles-Weed, K. S. Tai, and E. Wong, "Loss Landscape Smoothing: A Generalized Perspective to Sharpness-Based Regularization," arXiv, Available, Accessed: Nov. 18, 2024

[15] Hugging Face, "The Bellman Equation," Hugging Face, Available, Accessed: Nov. 18, 2024.

[16] C. J. C. H. Watkins and P. Dayan, "Q-learning," Machine Learning, vol. 8, no. 3, pp. 279-292, 1992, doi: 10.1007/BF00992698.

[17] H. Hasselt, "Double Q-learning," in Advances in Neural Information Processing Systems 23 (NIPS 2010), 2010, pp. 1-9.

[18] Hugging Face, "Advantage Actor-Critic," Deep Reinforcement Learning Course, accessed Dec. 20, 2024. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unit6/advantage-actor-critic

[19] C. Wu, I. Kim, and Z. Ma, "Deep Reinforcement Learning Based Traffic Signal Control: A Comparative Analysis," in Proc. 14th Int. Conf. Ambient Syst., Netw. Technol. (ANT), Leuven, Belgium, Mar. 15-17, 2023. Available: https://www.sciencedirect.com/proceedings/browse.

[20] M. Abdoos and A. L. Bazzan, "Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long short-term memory," *Expert Systems with Applications*, vol. 171, p. 114580, 2021.

[21] N. Kumar, S. S. Rahman, and N. Dhakad, "Fuzzy inference enabled deep reinforcement learning-based traffic light control for intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4919–4928, 2020.

[22] T. Hu, Z. Hu, Z. Lu, and X. Wen, "Dynamic traffic signal control using mean field multi-agent reinforcement learning in large scale road-networks," *IET Intelligent Transport Systems*, 2023

[23] Hugging Face, "Variance Problem," Deep Reinforcement Learning Course, accessed Dec. 20, 2024. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unit6/variance-problem

[24] M. Wang, L. Wu, J. Li, and L. He, "Traffic signal control with reinforcement learning based on region-aware cooperative strategy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6774–6785, 2021.

[25] Hugging Face, "Clipped Surrogate Objective," Deep Reinforcement Learning Course, accessed Dec. 20, 2024. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unit8/clipped-surrogate-objective

[26] S. Forsyth and J. Ponce, Computer Vision: A Modern Approach, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2012.

[27] R. Kundu, "YOLO: Algorithm for Object Detection Explained [+Examples]," V7. https://www.v7labs.com/blog/yolo-object-detection

[28] Q. U. Jamil et al., "Urban traffic signal control optimization through Deep Q Learning and double Deep Q Learning: a novel approach for efficient traffic management," pp. 1-24, 2024.

[29] M. Al-Turki, M. T. Kashifi, and N. T. Ratrout, "Traffic signal optimization framework using interpretable machine learning technique under heterogeneous-autonomy traffic environment," Neural Computing and Applications, vol. 36, no. 22, pp. 13761–13781, Aug. 2024, doi: 10.1007/s00521-024-09694-y.

[30] L. N. Alegre, "SUMO-RL," GitHub repository, 2019. [Online]. Available: https://github.com/LucasAlegre/sumo-rl. [Accessed: Dec. 7, 2024].

[31] L. Wang, Z. Ma, C. Dong, and H. Wang, "Human-centric multimodal deep (HMD) traffic signal control," IET Intelligent Transport Systems, vol. 17, no. 4, pp. 744-753, 2023.

[32]"Traffic Management Software Development," Intellias. https://intellias.com/traffic-management-softwaredevelopment/ (accessed Dec. 25, 2022
[33] Advanced traffic management system software TOPXVIEW | Telegra (telegra-europe.com)
[34] Iteris, "A leader in smart mobility," [Online]. Available: https://www.iteris.com/. [Accessed: Nov. 19, 2024].
[35] Kapsch TrafficCom, "Specializes in intelligent transportation systems, offering solutions for tolling, traffic management, and vehicle-to-infrastructure communication," [Online]. Available: https://www.kapsch.net/de. [Accessed: Nov. 19, 2024].

[36] Software Engineering Competence Center (SECC), SPIG Product Suite Handbook V1.2, Software Process Improvement Guide (SPIG), 1st ed., Giza, Egypt: Information Technology Industry Development Agency (ITIDA), Ministry of Communications and Information Technology (MCIT), 2010.

[37] OpenStreetMap contributors, "Planet dump," OpenStreetMap, 2017. [Online]. Available: https://planet.osm.org. [Accessed: Day Month Year].

[38] “Reinforcement Learning Tips and Tricks — Stable Baselines 2.10.3a0 documentation.” https://stable-baselines.readthedocs.io/en/master/guide/rl_tips.html

[39] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-Baselines3: Reliable Reinforcement learning Implementations,” 2021. http://jmlr.org/papers/v22/20-1364.html

[40] TomTom**,** "Alexandria Traffic," *TomTom Traffic Index*, accessed Jun. 12, 2024. [Online]. Available: https://www.tomtom.com/traffic-index/alexandria-traffic/

[41] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with Python," in *Proc. 9th Python in Science Conf. (SciPy)*, Austin, TX, USA, 2010, pp. 92–96.

[42] “Reinforcement Learning Tips and Tricks — Stable Baselines3 2.6.1a1 documentation.” https://stable-baselines3.readthedocs.io/en/master/guide/rl_tips.html

[43] C. Liechti, "pySerial: Python Serial Port Access Library," GitHub repository. [Online]. Available: https://github.com/pyserial/pyserial. [Accessed: Jun. 10, 2025].

[44] Streamlit Inc., “Streamlit,” ver. 1.35, [Online]. Available: https://streamlit.io/. [Accessed: Jun. 10, 2025].

[45] T. Simonini, "Generalization in Reinforcement Learning," Deep Reinforcement Learning Course. [Online]. Available: https://huggingface.co/learn/deep-rl-course/unitbonus3/generalisation. Accessed: Jun. 11, 2024.

APPENDIX I: CHALLENGE FACED

1. Project Complexity and Design

To address the inherent complexity of the system, several open-source projects were referenced during development. A review of relevant research papers was conducted to understand prior methodologies. Development began with an initial phase involving a simplified environment, which was gradually expanded. The SPI model was followed to progressively incorporate more complex components into the system

2. Action Space Complexity

A hierarchical, three-level action space was designed to manage decision complexity and enhance control mechanisms. This structure allowed for modular handling of agent behavior and facilitated scalability [5.1.4 Action Space].

3. Compatibility with SUMO and Real-World Data

To ensure compatibility between SUMO simulations and real-world video data, the logic associated with SUMO was decoupled from the main environment code. A unified Connection class was introduced, serving as a parent to both SumoConnection and RealConnection (for computer vision input). All environment communication was routed through this abstraction [4.4.5 Class Diagram].

4. Object Serialization Issues in Configuration Passing

Due to serialization issues when passing objects such as Connection during environment configuration, a workaround was implemented. The connection was instead set and retrieved using setter and getter methods, avoiding direct object transfer.

5. Frame Processing Latency with YOLO

Significant processing delays were observed due to the size of the YOLO model. To mitigate this, IP-based video capture was employed. Frames were requested based on synchronized time intervals aligned with the processing cycle, and frame timing was monitored for performance tuning [4.5.1 Computer Vision Implementation].

6. Thread-Safety Issues in IP Video Access

Accessing IP video feeds concurrently from both the browser and backend introduced thread-safety issues. A locking mechanism was implemented to ensure exclusive access to the video stream by a single thread at any given time [4.5.1 Computer Vision Implementation].

7. Backend Communication for IoT Devices

To support continuous listening by IoT devices without overwhelming the server or overloading low-end hardware, MQTT streaming was adopted. This allowed efficient, lightweight communication between clients and the server [4.5.3 Backend Implementation].

8. Shared State Management in the Backend

A mechanism was required for maintaining a shared variable representing the current system state. A centralized state file was created where shared variables and requests were globally defined and accessed as needed.

9. Access Control and User Management

To enforce secure access to the application, a user verification system was implemented. Only administrative users were granted the ability to register or add new users, ensuring controlled access to system features [4.5.3 Backend Implementation].

APPENDIX II: SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

1.Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements for the Traffic Manager System, which integrates Artificial Intelligence (AI) and Reinforcement Learning (RL) to optimize urban traffic flow. The system aims to reduce congestion, enhance mobility, and provide real-time decision-making for traffic management using IoT-enabled infrastructure.

1.2 Scope

The system shall dynamically adjust traffic signals based on real-time conditions, prioritize emergency vehicles, and offer real-time traffic updates via a mobile application. The system will be designed to scale from a single intersection prototype to a city-wide intelligent traffic control system.

1.3 References

[1],[2],[22],[24],[32],[34],[35]

2. System Requirements

2.1 Functional Requirements (FR)

FR-01: Intelligent Traffic Light Control

- The system shall utilize Reinforcement Learning (RL) to dynamically optimize traffic signals based on real-time sensor inputs.
- The system shall aim to minimize vehicle waiting times, depart delay, timeLoss and Maximize speed.

FR-02: IoT-Enabled Data Collection

- The system shall collect real-time traffic data from cameras.
- The system shall integrate computer vision to detect vehicle movement.
- The system shall store collected data securely for long-term analysis and AI model improvement.

FR-03: Emergency Vehicle Priority Management

- The system shall enable registered emergency vehicles (ambulances, fire trucks) to log in, select a route, and transmit their location.
- The system shall automatically adjust traffic signals along the emergency route by extending green light durations and minimizing red light durations.
- The system shall ensure priority handling does not disrupt overall traffic efficiency.
- The system shall allow only registered emergency users (e.g., ambulance drivers, police, fire personnel) to report accidents by specifying the location and severity.

FR-04: Accident-Based Traffic Signal Control

- The system shall allow authenticated users to report accidents by specifying the location and severity.
- Upon receiving a valid accident report, the system shall automatically adjust traffic signals near the reported location by extending red light durations and minimizing green light durations to restrict vehicle movement.

FR-05: Data-Driven Traffic Simulation and Optimization

- The system shall utilize SUMO (Simulation of Urban Mobility) to simulate traffic behavior and train initial traffic control models.
- The system shall be improved using collected offline traffic experience data (e.g., historical patterns) to refine decision-making algorithms.
- The system shall apply and validate the optimized models using real-time traffic data for live analysis and control.
- The system shall enable continuous monitoring of system performance, including metrics such as traffic flow efficiency, model accuracy.

2.2 Non-Functional Requirements (NFR)

NFR-01: Real-Time Processing & Low Latency

- The system shall process incoming data and optimize traffic flow in real-time.
- The system shall maintain low-latency decision-making for seamless traffic signal adjustments.

NFR-02: Scalability & Expandability

- The system shall be designed to support deployment at any number of intersections, regardless of whether they follow the same or different traffic flow directions.
- The system shall be designed for deployment at a single intersection with the ability to scale to city-wide deployment.
- The system shall support coordination between multiple traffic lights.

3. Tools & Environment

- Simulation Framework: SUMO, OpenAI Gym
- Machine Learning Framework: Python, TensorFlow, StableBasline3, Ultraticys
- Mobile Development: Flutter
- Hardware: IoT sensors (cameras)
- Video Processing: OpenCV
- Deployment: Aiven, Huggingface, fastapi

4. Conclusion

This document outlines the critical system requirements for the Traffic Manager Project, ensuring alignment with IEEE 1233-1998 standards. It provides a structured breakdown of functional and non-functional requirements, tools, and key system components. The system aims to deliver intelligent, adaptive, and sustainable traffic management solutions for modern urban environments.

APPENDIX III: MINUTES OF MEETING

Minutes of meetings are considered formal records of the discussions and decisions made during a meeting, essential for ensuring clarity and accountability. The date and time of the meeting, the location, and a list of attendees are typically included. The key points discussed, decisions made, and any action items assigned are captured, along with the responsible individuals for those tasks. Additionally, the time of the next scheduled meeting is documented, providing a clear reference for future follow-ups and ensuring that all participants are aligned on upcoming activities and deadlines. This comprehensive documentation helps ensure transparency and continuity across the project or organizational activities.

Minutes of Meetings – Tuesday 12/Nov/2024

Place	Online
Date and time	12th of November 2024 at 8:30 pm
Attendees	
	Taha Fawzy Anwar Elshrif Ahmed Ibrahim Hassan Kerolss Bolis Shehata Remon Ehab Ramses Youssef Ibrahim Hanfy Marwa Shebl Emad Mohammed Fawzy Mostafa Hussian Mostafa Eldeep Mayan Islam Mohamed Ahmed Bichay Adel Ramses

Minutes	
The points listed are not in the order of discussion but capture the essence of the meeting.	
1	Reviewed week work.

Figure 0III.00.1 screen of minutes of meeting

APPENDIX IV: WEEKLY TIMESHEET

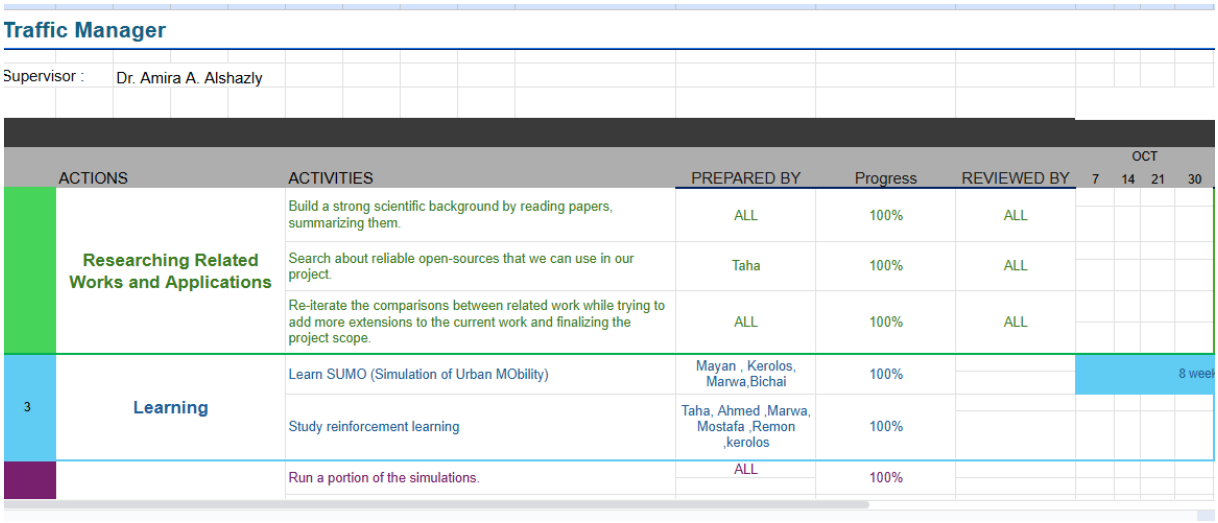
A weekly timesheet is a record that is used to track the work hours of everyone for a given week. The tasks completed and the time spent on each activity are captured. This helps ensure progress is monitored, accountability is maintained, and an overview of productivity is provided.

Graduation Project Weekly Timesheet										
Project		From	1/12/2024	To	8/12/2024					
Name	fcds Graduation Project 2024_2025		Department		Data Science					
Team Member / Activity	Taha Fawzy	Ahmed Ibrahim	Emad Mohammed	Mayan Islam	Kerolis Boils	Remon Ehab	Mostafa Hussian	Marwa Shebl	Youssef Ibrahim	Bichai adel
Writing /Updating Presentation	4	2	1	4	1	0	1	0	6	0

Figure IV0.1 screen of a weekly timesheet

APPENDIX V: JIRA PLAN

To ensure effective project management and task coordination, we utilized JIRA as our primary planning tool. The following screenshot provides an overview of our workflow, highlighting task allocation, progress tracking, and milestone planning throughout the development lifecycle.



Traffic Manager

Supervisor : Dr. Amira A. Alshazly

	ACTIONS	ACTIVITIES	PREPARED BY	Progress	REVIEWED BY	OCT 7	14	21	30
	Researching Related Works and Applications	Build a strong scientific background by reading papers, summarizing them.	ALL	100%	ALL				
		Search about reliable open-sources that we can use in our project.	Taha	100%	ALL				
		Re-iterate the comparisons between related work while trying to add more extensions to the current work and finalizing the project scope.	ALL	100%	ALL				
3	Learning	Learn SUMO (Simulation of Urban MObility)	Mayan , Kerolos, Marwa,Bichai	100%					8 wee
		Study reinforcement learning	Taha, Ahmed ,Marwa, Mostafa ,Remon ,kerolos	100%					
		Run a portion of the simulations.	ALL	100%					

Figure V.0.1 screen of a JIRA Plan