Declaration

I hereby declare that this dissertation is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have also completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

Signed: ___________________________ Date: ___________________________
Abstract

Traffic congestion has caused trouble for people for many years. With the level of traffic increasing in previous years, local, state and national governments, along with researchers, are looking for a way to manage this problem. Ramp Metering is a method developed to reduce highway traffic by controlling the flow of vehicles. What started as a manual process has now technically advanced, with the rise of AI creating adaptive solutions. However, traditional AI methods require a lot of data and communication power to carry out this. With data protection rules and regulations on the rise, this data might be difficult to source. Therefore, adaptive models are complicated to train and, as a result, create.

The proposal of Federated Reinforcement Learning (FRL) seems to be a solution to this problem in recent years. Combining reinforcement learning with federated learning, FRL can create an adaptive solution by communicating with many different client nodes while protecting the privacy of the data.

This thesis aims to create an FRL design for the problem of adaptive/intelligent ramp metering control. Through research into the background and related works of FRL, a design is proposed. This design is implemented by using SUMO, a simulation environment, and TraCI, an interface built to interact with SUMO. The design consists of a Q-Learning method integrated into an FL model. The design is then evaluated in three different scenarios; low traffic load, high traffic load and mixed traffic load. It is compared to industry-popular methods of ramp metering such as fixed-timing and Q-Learning. From the results, it can be concluded that there are no improvements in the network by using FRL. This could be due to a configuration error which will be detailed in Chapter 5. However, FRL still poses an excellent research opportunity, and further work in this area is detailed in the final chapter.
Acknowledgements

There are many people I would like to thank. Firstly, I would like to thank God for providing me with such an opportunity to learn in an area that is novel to the world and myself. This thesis has been a very significant part of my Masters journey.

I would also like to thank my supervisor, Ivana, for all her support in this research process, ready to help out and answer questions even at the last minute.

I would like to thank my parents who always checked up on me and made sure that I was taking care of my health while studying and researching.

Finally I would like to thank my friends, especially Evlin, Rvail and Kyra, who motivated me and inspired me to take on this research, while making sure I socialised and looked after my mental well-being.

Thank you all,

All of the help means the world to me,

Helen
## Contents

1 Introduction .................................................. 1
   1.1 Ramp Metering ........................................... 1
      1.1.1 Reinforcement Learning in Ramp Metering .......... 3
      1.1.2 Federated Reinforcement Learning ................. 3
   1.2 Thesis Aims and Objectives ............................. 4
   1.3 Thesis Assumptions ...................................... 4
   1.4 Thesis Contribution ..................................... 5
   1.5 Document Structure .................................... 6

2 Background and Related Work ............................... 7
   2.1 Reinforcement Learning ................................ 7
      2.1.1 Q-Learning - Value based method ................. 9
      2.1.2 Reinforce algorithm - Policy based method ....... 10
      2.1.3 Actor-critic methods ............................. 11
   2.2 Federated Learning ..................................... 12
      2.2.1 Architecture ...................................... 13
      2.2.2 Applications of FL ............................... 14
      2.2.3 Open Issues of FL .............................. 15
   2.3 Federated Reinforcement Learning ...................... 15
      2.3.1 Horizontal Federated Reinforcement Learning ..... 15
      2.3.2 Vertical federated reinforcement learning ....... 17
   2.4 Applications of Federated Reinforcement Learning ... 18
      2.4.1 FRL for communication networks .................. 18
      2.4.2 FRL for attack detection ......................... 18
   2.5 The Problem of Traffic ................................ 18
      2.5.1 Ramp Metering .................................... 19
      2.5.2 Federated Reinforcement Learning in Traffic Control 20
   2.6 Summary ................................................. 21

3 Design ......................................................... 22
3.1 Traffic Control Problem ........................................... 22
   3.1.1 State Representation ....................................... 22
   3.1.2 Action Space ................................................ 23
   3.1.3 Reward Function ........................................... 24
   3.1.4 Q-Learning Algorithm ..................................... 29
3.2 Federated Reinforcement Learning Techniques .................. 31
3.3 Summary .......................................................... 32

4 Implementation ...................................................... 33
   4.1 Simulation Environment ....................................... 33
      4.1.1 Components .............................................. 33
      4.1.2 TraCI .................................................... 34
   4.2 Federated Reinforcement Learning ............................ 35
      4.2.1 Objects ................................................ 35
      4.2.2 Functions .............................................. 36
   4.3 Summary ........................................................ 37

5 Evaluation .......................................................... 39
   5.1 Objectives ..................................................... 39
   5.2 Metrics ........................................................ 39
   5.3 Evaluation Scenarios ......................................... 40
      5.3.1 Evaluation Techniques ................................. 40
      5.3.2 Scenarios .............................................. 40
   5.4 Setup .......................................................... 41
      5.4.1 Network Layout ....................................... 41
      5.4.2 Traffic Demand ....................................... 41
   5.5 Results & Analysis ........................................... 42
      5.5.1 Low Traffic Load ..................................... 42
      5.5.2 High Traffic Load ................................... 43
      5.5.3 Mixed Traffic Load .................................. 43
   5.6 Further Evaluation Notes .................................... 43
   5.7 Evaluation Summary ......................................... 44

6 Conclusion & Future Work ......................................... 54
   6.1 Thesis Contribution .......................................... 54
   6.2 Future Work .................................................. 55
      6.2.1 Corrections .............................................. 55
      6.2.2 Areas of Development .................................. 55

v
## List of Figures

1.1 Ramp metering in the top U.S. metropolitan areas ........................................ 2  
1.2 Federated Reinforcement Learning design for ramp metering ...................... 5  
2.1 Design of a Reinforcement Learning process ............................................. 8  
2.2 Actor-Critic RL Architecture ................................................................. 11  
2.3 A user’s phone personalizes the model locally, based on their usage (A). Many  
    users’ parameters are aggregated (B) to create an update (C) to the global  
    model. This process is then repeated. .................................................. 12 
2.4 Example of HFRL process and architecture ............................................. 16 
2.5 Example of VFRL process and architecture ............................................. 17 
2.6 Comparison of mainline conditions with and without ramp metering ............. 19 
3.1 Example of Ramp Metering, showing location of TLS at entrance to dual-  
    carriageway and detector located at position A ........................................ 23 
3.2 Evaluation of Reward Function 1 ............................................................ 26 
3.3 Evaluation of Reward Function 2 ............................................................ 26 
3.4 Evaluation of Reward Function 3 ............................................................ 27 
3.5 Evaluation of Reward Function 1 with first plot showing average waiting time  
    for a vehicle and second plot showing cumulative rewards ........................ 28 
3.6 Evaluation of Reward Function 3 with first plot showing average waiting time  
    for a vehicle and second plot showing cumulative rewards ........................ 29 
3.7 Q-learning model design in Ramp Metering ............................................. 30  
3.8 Horizontal Federated Reinforcement Learning design in Ramp Metering ....... 31 
4.1 UML Class Diagram for FRL Implementation ........................................... 37 
5.1 N7 Network Layout for FRL ................................................................. 41 
5.2 Fixed Timing Results in Low Load Traffic(Baseline Results) ....................... 45 
5.3 Q-Learning Results in Low Load Traffic ............................................... 46 
5.4 FRL Results in Low Load Traffic ......................................................... 47 
5.5 Fixed Timing Results in High Load Traffic(Baseline Results) ...................... 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.6 Q-Learning in High Load Traffic</td>
<td>49</td>
</tr>
<tr>
<td>5.7 FRL Results in High Load Traffic</td>
<td>50</td>
</tr>
<tr>
<td>5.8 Fixed Timing Results in Mixed Load Traffic (Baseline Results)</td>
<td>51</td>
</tr>
<tr>
<td>5.9 Q-Learning Results in Mixed Load Traffic</td>
<td>52</td>
</tr>
<tr>
<td>5.10 FRL Results in Mixed Load Traffic</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Various FL architectures, their benefits and focuses .......................... 14
2.2 Description of algorithms used in Ramp Metering today ....................... 20
3.1 RL Reward Function Evaluation hyper-parameters .............................. 25
3.2 RL Reward Function Evaluation hyper-parameters (explicitly for Reward Func-
tion 1 and Reward Function 3 ) .............................................. 28
5.1 Parameters of FRL and Q-Learning model training .............................. 42
1 Introduction

This thesis addresses intelligent control of ramp metering as the first study on the application of Federated Reinforcement Learning (FRL) on-ramp meters. In today’s society, one of the issues faced by the ordinary person is the issue of traffic delays caused by congestion. In 2020 and for the most of 2021, a decrease in traffic congestion was seen due to the pandemic employing restrictions on travel regionally, nationally and internationally. However, with life returning to a somewhat norm after the introduction of vaccines, traffic is again on the rise (1), causing congestion. Building new roads and adding lanes on existing motorways or national highways will not reduce congestion or improve traffic speeds and instead induces traffic (2) (3). Therefore, to minimise the impact of these issues, organisations of traffic management will investigate various traffic control methods.

1.1 Ramp Metering

Ramp Metering is a method to control the traffic flow on national highways and motorways, becoming more popular in the United States and throughout Europe. However, its first implementation did not involve any automation - a police officer manually directed the traffic flow on the Eisenhower Expressway in Chicago in the United States in the 1960s. (4). After this, ramp metering spread on an experimental basis throughout the U.S. with constant development. In the 1980s, European countries also began the implementation of ramp metering, with the United Kingdom and the Netherlands being the first to experiment with it.

As ramp metering became more widespread across the globe, its technology also began to evolve, no longer needing a physical person to monitor the roads. With the use of signal heads, detectors and signage, ramp metering systems were conceived. The most common methods of implementation are fixed-time, local control and system-wide control (detailed in 2.5.1). In the U.S., in particular, ramp metering is used throughout the country, as shown in Figure 1.2.
However, since fixed-timed algorithms control most ramp metering systems, these are not adaptive to the scenarios faced by the agents. Therefore, intelligent, responsive methods of ramp metering (especially using Artificial Intelligence) have been studied in recent years. Responsive methods can be categorized into rule-based, optimization-based, and Reinforcement Learning (RL) based.

- **Rule-based**: This approach uses specific rules to update the Traffic Light System (TLS) in real-time. There have been a few proposed solutions, but the most popular one is a feedback ramp metering algorithm known as ALINEA. This algorithm would regulate the flows from the on-ramp based on the information from detectors to reach a predefined occupancy value.(5)

- **Optimization-based**: This approach uses models to predict traffic behaviour, taking into account ramp meter data and traffic states. However, one may encounter inconsistent control issues that may cause difficulty determining optimal signal timings.(6)

- **RL-based**: RL is the use of Artificial Intelligence (AI) to determine a policy for ramp
Reinforcement Learning in Ramp Metering

Reinforcement Learning is an AI technique that does not depend on preprocessed data to learn. Instead, it is brought into the environment of the problem at hand and has to learn, through trial-and-error, the most optimal decision to carry out in a situation. RL creates a model that can adapt to any number of situations once given the situations and the list of actions it can take in a particular situation. RL was initially applied to urban traffic networks to optimize traffic signal control (8) (9). In more recent years, researchers have introduced RL to ramp metering control, with Davarynejad et al. (10) exploring the use of a standard Q-learning algorithm for a local ramp metering control problem in 2011. Schmidt Dumont and van Vuuren (11) employed a decentralized form of RL using ramp metering and variable speed limits to control traffic flow on highways in 2015. However, the use of reinforcement learning comes with its challenges.

As mentioned above, RL does not learn from a processed dataset. It learns over time from the system’s behaviour about what is an appropriate action and what is an undesired behaviour. Therefore, it requires a long learning period from various situations to create an optimal model. This issue could be resolved by transferring data from model to model in a distributed system. Yet, this brings about the concern of data privacy and protection. For an RL model to learn about as many scenarios as possible, it may need to learn from situations in different cities, states, and countries. But, due to local and international laws of privacy (such as GDPR), it may not be permissible to transfer data. This issue would result in an extensive training time for RL algorithms in ramp metering.

1.1.2 Federated Reinforcement Learning

Federated Reinforcement Learning (FRL) is a novel approach to RL that considers the data privacy and scarcity of data concerns that traditional methods of RL present. It is a combination of Federated Learning (FL) and RL that aims to improve the training speed of RL by training through data distributed over several clients while also protecting the privacy of data by only learning from (encrypted) parameters of the data and not the data itself (12). Therefore, given a small state space, action space and minimal data, FRL would be able to create an intelligent adaptive system.

Researchers have investigated the application of FRL in many different areas, including communication networks and control optimization, to name a few (detailed in Section 2.4 ).
However, its study in the optimization of ramp metering is yet to be explored.

1.2 Thesis Aims and Objectives

With the ability to learn from any environment and create an adaptive solution while protecting the privacy of the data, Federated Reinforcement Learning has the potential to become the cutting edge technology used in ramp metering. The objective of this thesis is to propose the design and implementation of such a solution. This thesis aims to investigate the effectiveness of the application of Federated Reinforcement Learning in ramp metering and compare this method against methods commonly used in the industry. Using metrics such as CO2 emissions, average vehicle waiting time and average speed, this research will establish the effect of FRL in ramp metering on the environment and traffic and compare it to the impacts of industry-standard methods.

1.3 Thesis Assumptions

To complete the study being conducted in this thesis, a few assumptions have been made to limit the scope of the thesis by restricting the number of issues it addresses.

First is that all agents are assumed to be stationary. Since the agents represent the traffic lights used in ramp metering, they will not move through the environment in this evaluation. It is assumed that the agents are all programmed to be failure-free and accurately communicate with each other in the manner designed.

The state-space only includes three phases of the traffic light system - Green, Red, Yellow, in the order of Green → Yellow → Red → Green. This thesis will not investigate the use of other phases such as ‘off-blinking’ or ‘red + yellow light’.

In the simulations conducted for this thesis, the programmed control mechanism (FRL, RL or baseline) determines the behaviour of the traffic lights. In contrast, the behaviour and characteristics of the vehicles are predefined. This includes the vehicle’s starting type, position, speed, and route. Vehicle types only include passenger vehicles and trucks. During a model’s training, if there is no space for vehicles to join the simulation at the defined junction, another junction is used to insert the vehicle.

Federated Averaging, FedAvg, is the method used for aggregating the global model while integrating Federated Learning into the systems. Other forms of aggregation were not investigated as this method is the most popular used in the industry and the simplest to implement. (*REF*)

The encryption process in Federated Learning is assumed to be pre-programmed and
integrated into the system. Therefore the implementation and operation of encryption are not investigated or carried out for the scope of this thesis. Generally, the system coordinator encrypts parameters fed into the Federated Learning model before being shared with the global model. This work is assumed to be carried out separately.

1.4 Thesis Contribution

Through the investigation of FRL, this thesis motivates and proposes its use in ramp metering since this methodology of traffic system control has not been studied before. Each ramp meter (signal head and its corresponding induction loop detector) is considered an agent in this novel system. After RL, the information from the detector determines the behaviour of the signal head. The collection of ramp meters is considered to be the clients for the FRL system and will be implemented as the design below:

![Figure 1.2: Federated Reinforcement Learning design for ramp metering](source: Parsons Brinckerhoff)

The thesis evaluates this approach in SUMO, a simulation environment used to simulate the traffic flow of vehicles in different scenarios. The scenarios being considered are Ramp Metering through FRL, Ramp Metering through RL and Ramp Metering using Fixed Timed control (rule-based control). This evaluation will inform whether the use of FRL outperforms the standard RL and fixed timing methods used in ramp metering today.
1.5 Document Structure

The structure of the thesis is as follows. Chapter 2 introduces background information on Reinforcement Learning and the innovation of Federated Learning. It focuses on the techniques used for the proposed work, focusing on model-free RL and horizontal FL techniques. Chapter 3 describes the design of the proposed research. Chapter 4 presents the implementation of the strategy stated in Chapter 3. Chapter 5 describes the evaluation of the suggested study as a ramp-metering approach and analyses the findings. Chapter 6 concludes this thesis with a summary of the work and outlines the issues that remain open for future work.
2 Background and Related Work

This chapter focuses on the main technical concepts used to complete this research. The first section covers the topic of Reinforcement Learning and the different techniques used to implement this. The second section explores the topic of Federated Learning. Since this is a relatively novel method of machine learning, not only will the techniques and applications be investigated but also the open challenges for federated learning. The chapter will also discuss the various problems encountered in traffic and how the concept of Federated Reinforcement Learning can help reduce/eliminate these problems through ramp metering.

2.1 Reinforcement Learning

Generally, in Machine Learning (ML), we encounter supervised and unsupervised learning methods, where the machine learns from pre-processed data. Reinforcement Learning (RL), on the other hand, uses a series of trial-and-error rounds based on a reward system to guide the agent to learn and execute the optimal decision. Characteristics of RL include:

- Unlike other machine learning paradigms, there involves no supervisor but a reward signal.
- The feedback is delayed as time is needed to interpret whether the action resulted in consequent rewards or punishments.
- The system consists of sequential decision-making processes on non-i.i.d data.
- The agent’s actions will influence the subsequent data it receives, as each step has a consequence.

Conventionally, a reinforcement learning model consists of the following concepts:

- **Agent**: The agent in RL represents the learning algorithm/machine. An agent possesses a discrete set of agent actions, \( A \). Through these actions, the agent has the opportunity to modify their behaviour with the expectation of improving its decision-making process (by improving the rewards received)
- **Environment**: The environment is a depiction of the problem at hand to be solved.
• **Time step:** A time step involves an interaction between the agent and the environment. These time steps are so used to identify an iteration/interaction.

• **State:** A state in RL gives a representation of the current environment of the problem at hand. Actions taken by the agent can affect the state. Therefore, RL involves a discrete set of environment states, $S$; and

• **Reward:** This is a numeric value returned by the environment after the completion of an action. Hence, an RL model contains a set of scalar reinforcement (reward) signals, $R$.

![Figure 2.1: Design of a Reinforcement Learning process](image)

In a standard reinforcement-learning model, at time step, $t$, the agent receives the current state of the environment, $S(t)$. After observing $S(t)$, the agent decides on employing an action, $A(t)$ and applies this to the environment. After this, the environment adjusts its state, providing a new state, $S(t+1)$, and calculates an immediate reward, $R(t+1)$. The agent’s behaviour should adjust to this reward value and aim to increase the cumulative sum of these scalar values through trial and error steered by a range of algorithms (the coming sections of this chapter explain some of these). The cumulative sum can be summarized as:

$$G_t = R_{t+1} + R_{t+2} + ... + R_T$$  \hspace{1cm} (1)

There are two types of tasks that can run through RL. An episodic task means that the task was completed with the termination action as the last step $T$. The other type of task is known as a continuing task. There are no termination states in these tasks, meaning that these tasks can run for an infinite period. The above equation may provide different results for a continuing task. Therefore, we calculate the cumulative sum with the following:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$  \hspace{1cm} (2)

Here $\gamma$ is the discount factor and needs to satisfy $0 \leqslant 1$. The value of $\gamma$ refers to how
much the agent anticipates future rewards. When $\gamma = 0$, the agent only sees the current reward. As $\gamma$ moves closer to 1, the agent will be able to see more of the future rewards available. This factor will encourage the agent to focus not only on immediate gain but also on future, more valuable gains, leading to a series of actions that result in low reinforcement to arrive at a state with a high reward finally. This process is known as delayed reinforcement, and Markov Decision Processes (MDPs) can model situations with delayed reinforcement. A model can only be considered Markov if the state transitions are independent of the agent’s previous state of action. This model, along with the set of actions and states, also consists of two value functions:

- a reward function $R: S \times A \rightarrow r$; and
- a state transition function $T: S \times A \rightarrow \Pi(S)$

The reward function calculates the reward of a given action and state. The state transition function specifies the next state given the current and chosen action.

### 2.1.1 Q-Learning - Value based method

Q-learning is a model-free RL method, meaning that the system learns the controller rather than a model. The objective here is to calculate the value of an action in a particular state and find the most optimal action. This method is a value-based method, meaning that in each round of learning, the value function, $Q_\pi(s, a)$, is re-calculated, and the algorithm is improved based on this. The goal here is to find the value of every action in a given state, to provide the optimal $Q_\pi(s, a = a^*)$, where $a^*$ is the best action possible. The strategy for Q-learning involves gathering the Q values of all actions and choosing the action with the highest Q-value for each given state which would maximize the cumulative sum of rewards for all future actions from the current state.

Q-learning uses a Q-table to store the values of all Q-value functions. Each row in the Q-table represents a state, and each column represents an action the agent can take. Updating the Q-value is a critical process in Q-learning. Each $(s, a)$ pair corresponds to a Q-value and $Q(s, a)$ is updated in the following manner

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a' \in A} Q(s', a') - Q(s, a)] \quad (3)$$

where $r$ is the reward calculated through the reward function taking into account the action, $a$ at state, $s$ at time step, $t$. $s'$ and $a'$ represent the state and action taken by the agent at the next time step. $\alpha$ is the learning rate to determine how well the agent needs to learn from the errors, and $\gamma$ is the attenuation of future reward.

Once the agent has continuously accessed all the available (state, action) pairs, the
algorithm will converge to reach the optimal Q-function. Therefore, Q-learning is suitable for simple problems with small state and action spaces, meaning that it will quickly converge to create an optimal model.

2.1.2 Reinforce algorithm - Policy based method

The policy is the agent’s approach to taking an action, \( a \) for a state, \( s \). The above method shows that value-based methods aim to update the value function and reach a policy iteratively. However, policy-based methods aim to generate the optimal policies and policy parameters given sufficient training. The algorithm provides the action once a state, \( s \), is given as input, rather than receiving a Q-value of \( s \).

The algorithm, takes a parameterized policy, \( \pi_\theta \) and attempts to maximize the expected reward using the objective function as defined below:

\[
J(\pi_\theta) = E_{\pi_\theta} \left[ \sum_{t=0}^{T-1} R(s_t, a_t) \right] = E_{\pi_\theta} [R(\tau)]
\]  

(4)

\( R(s_t, a_t) \) is the reward received at time-step, \( t \), after carrying out action \( a_t \) at the state \( s_t \), which can also be represented as \( R(\tau) \). Calculating the gradient ascent finds the parameters that would maximise the objective function. This can be represented through the derivative of Equation 4, which can be written as:

\[
\Delta J(\pi_\theta) = \Delta E_{\pi_\theta} [R(\tau)]
\]

(5)

Since the sum of rewards needs to be calculated, the policy gradient algorithm can only be updated after each episode.

REINFORCE is an example of a policy-based method. This algorithm applies Monte Carlo to approximate the average value of multiple samples of the rewards. After each Monte Carlo sampling, the algorithm generates a trajectory. The cumulative reward can be calculated by collecting these, using the approximations as the loss function for the gradient ascent. (13)

Policy-based methods are appropriate when there is a significant action space or a continuous action space - computations and the memory space required can grow quickly with a larger action space. Still, since policy-based methods learn from a set of parameters, it does not need this space.
2.1.3 Actor-critic methods

Actor-Critic is similar to REINFORCE discussed above. It is a temporal difference (TD) method with a separate memory representing the policy independent of the value function. Using neural networks, its architecture (as shown in Fig 2.2) contains two components:

- **Actor**: Samples an action given a state and carries it out. This would be considered the policy structure.
- **Critic**: Informs the actor of the effect of its chosen action and the adjustments needed through a scalar figure known as a TD error.

![Figure 2.2: Actor-Critic RL Architecture](image)

Source: Reinforcement Learning: An Introduction (14)

TD error is the only signal that the model has for learning and it is calculated after the performance of each action using the following equation:

\[
\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)
\]

where \(V\) represents the value function used by the critic. \(\delta_t\) evaluates whether the action chosen at time, \(t\), \(a_t\), produced a desirable output. If \(\delta_t\) is a positive value, it increases the likelihood of \(a_t\) being selected in the future. If \(\delta_t\) is negative, it decreases the chances of \(a_t\) being chosen in the future. (14) Several domains apply the actor-critic algorithms. In robotics, the application of these algorithms showed signs of success from the early stages itself. Nakamura et al. produced an algorithm that made a robot walk stably in (15).
Another successful implementation was conducted by H. Kimura et al. in enabling a four-legged robot to walk as seen in (16). The methodology was also applied in logistics. When given the problem of dispatching forklifts in a warehouse, the algorithm was able to find an optimal solution that performed 20% better in comparison to other algorithms (17). The actor-critic method was also used in determining the dynamic pricing of electronics in the market in (18).

### 2.2 Federated Learning

Federated Learning (FL) is used to train models in a collaborative environment without exchanging their training data. This process detaches machine learning from the need to store data in a cloud. In other terms, it is a decentralized form of machine learning.

The FL process (as shown in Figure 2.3) would involve a machine downloading the most updated version of a model. The machine then learns from this model, improves it with its data and then encrypts the changes made. Only the encrypted changes are then sent forward to the cloud. The training data remain locally. The data sent forward is then averaged with other users’ data to improve the model.

![Figure 2.3: A user’s phone personalizes the model locally, based on their usage (A). Many users’ parameters are aggregated (B) to create an update (C) to the global model. This process is then repeated.](image)

Source: Google (19)

This distributed machine learning approach has many features and addresses problems associated with conventional methods. The most prominent advantage of FL is the privacy it offers. No local data is exchanged or uploaded to the cloud and any parameters shared are
extensively encrypted. While data leakage is still possible, protective measures in the encryption and uploading process significantly reduce this risk. In addition to this, though a federated learning model provides great insight given global data, it can also offer personalization. This can be done by generating two models simultaneously, allowing the opportunity to learn and generalize the model given their local data. FL is also favourable from a legal perspective, giving tech firms the data they require while adhering to current European GDPR (General Data Protection Regulation) terms.

Each client (device) - server interaction is a federated learning round. Each round consists of the following steps:

1. Selection: The server selects a subset based on specific goals from the devices that meet the eligibility criteria and are willing to cooperate with the server. Rejected devices can come back at a later time.

2. Configuration: The server is configured for the selected devices based on chosen aggregation process. The server then sends the FL plan (a data structure that includes a model graph, the necessary hyper-parameters, and instructions on executing relevant tasks) and an FL checkpoint with the global model to all devices selected.

3. Reporting: The participating devices send updates to the server. The server has to receive these updates within a given time frame. Any reports provided after this time will be disregarded. As the reports are received, they will be aggregated. The round is considered successful if enough reports are collected. Otherwise, the round is disregarded.

### 2.2.1 Architecture

The architecture of FL varies with the learning problem at hand. However, the model’s parameters usually remain uniform through the designs. Common parameters seen in FL designs are:

- **$T$**: Number of FL rounds
- **$K$**: Total number of devices in the network
- **$C$**: Fraction of devices used in each iteration/FL round
- **$B$**: Local batch size used at each iteration/ FL round

Various architectures and their focuses have been detailed in the table below:

The distribution of the data determines the architecture used. Looking at Horizontal FL (HFL) and Vertical FL (VFL), these architectural designs are different in how they are defined and structured. HFL learns from clients that have similar features but vary in terms


<table>
<thead>
<tr>
<th>Architecture</th>
<th>Benefit(s)</th>
<th>Focus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizontal FL</td>
<td>Independence</td>
<td>Security</td>
</tr>
<tr>
<td>Vertical FL</td>
<td>Encryption</td>
<td>Privacy</td>
</tr>
<tr>
<td>Federated Transfer Learning</td>
<td>Higher accuracy</td>
<td>Reduces accuracy loss</td>
</tr>
<tr>
<td></td>
<td>Encryption</td>
<td></td>
</tr>
<tr>
<td>PerFit</td>
<td>Cloud-based</td>
<td>IoT applicability</td>
</tr>
<tr>
<td></td>
<td>Local-sharing</td>
<td></td>
</tr>
<tr>
<td>FedHealth</td>
<td>Powerful models</td>
<td>Healthcare</td>
</tr>
<tr>
<td></td>
<td>Increased generalization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Increased applicability</td>
<td></td>
</tr>
<tr>
<td>Blockchain-FL</td>
<td>High efficiency</td>
<td>Industrial IoT</td>
</tr>
<tr>
<td></td>
<td>Enhanced security</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Various FL architectures, their benefits and focuses

**Source:** Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications (20)

of the data they contain. On the other hand, VFL learns from clients that have data sets with similar sample IDs, but the features vary.

Therefore, when applying FL, one needs to examine the data distribution to decide on a suitable architecture.

### 2.2.2 Applications of FL

Data privacy is a rising area of concern for many users, and as a result, FL is becoming a more popular method of Machine Learning.

Providing a personalized experience for a customer is very important for any company to increase the customer retention rate (21). However, training a model requires a lot of the customer’s data, and with data protection restrictions, recommendation algorithms are becoming more and more challenging to build and regulate. FL features allow training models on local devices while protecting the user’s privacy since the data does not leave the user’s device (22).

FL can also be utilized in the area of smart healthcare. The medical field is an area where data is key to identifying illnesses through symptoms and genetic makeup. For example, it is possible to build a tumour detector through AI; however, it is very sensitive and extremely difficult to obtain data to train such a model. Medical field research can exploit the privacy and data protection that FL offers and drive great innovation in medical science and technology (23).
2.2.3 Open Issues of FL

Though FL presents an opportunity for great innovation through its unique characteristics, it also presents some challenges.

Since an FL model requires communication between clients in the network, it consumes a lot of power. As the network grows and the level of computation grows, this creates a slower model. There have been studies that approach the communication challenges of FL (24) (25), but they do not address the issue of energy consumption.

FL is still vulnerable to adversarial attacks, with poisoning attacks and inference attacks being the most prevalent (26). Poisoning attacks prevent the machine from learning accurately about the data by interfering in the local learning process or injecting a bias into the data. Inference attacks target the user’s data and attempt to create a leak through the weighted parameters that are being communicated through the network (27). Further work is needed to make FL resistant to such attacks.

2.3 Federated Reinforcement Learning

Federated Reinforcement Learning (FRL) is a novel approach whose techniques were developed to solve the problems of data shortage and data protection encountered in Reinforcement Learning. This is done by integrating it into a Federated environment, allowing machines to learn more efficiently with more security. FRL can be divided into two different categories: Horizontal Federated Reinforcement Learning (HFRL) and Vertical Federated Reinforcement Learning (VFRL).

2.3.1 Horizontal Federated Reinforcement Learning

Horizontal Federated Reinforcement Learning (HRFL) is applied in situations where the agents are placed in different locations (geographically). Still, they face similar problems/tasks and have little-to-no interaction with each other in the environment. The environment, state-space, and action space would replace the data set, feature space, and label space of basic FL.
The methodology for carrying out HFRL is outlined below:

1. Initialization: This process can be separated into two cases. One is where the agent does not have a model locally. In this case, the agent must download the shared global model from a coordinator. The second case is when the agent has a local model, in which case the agent would have to confirm the model type and parameters with the coordinator.

2. Training: Each agent observes its environment and trains its local model based on a private strategy. As seen in RL, the reward is determined based on the action through the reward function. The state-action-reward-state (SARS) cycle continues until the model is trained sufficiently.

3. Submission: Once the trigger conditions for submission are met, the agent can transmit the local model parameters, once encrypted, to the coordinator.

4. Aggregation: Once aggregation trigger conditions are met, the coordinator runs an aggregation algorithm to update the global model.

5. Transmission: The coordinator sends the aggregated models to the agents.

6. Updation: The local agents update their corresponding models using the federated model.

Figure 2.4: Example of HFRL process and architecture

Source: Federated Reinforcement Learning: Techniques, Applications, and Open Challenges (13)
2.3.2 Vertical federated reinforcement learning

In Vertical Federated Reinforcement Learning (VRFL), the data sampled from different agents belong to a common group or user, containing various features. Here the RL agents may belong to the same environment but may encounter different interactions, resulting in further observations.

The methodology for carrying out VFRL is outlined below:

1. Initialization: All agents' models are initialized
2. Training: Each agent observes their environment and receives the states. Actions vary from agent to agent, and rewards are obtained after interacting with the environment. The state-action-reward-state (SARS) cycle is used to train all models
3. Submission: Each agent performs an averaging function and submits the encrypted parameters directly to the global federated model
4. Aggregation: Once aggregation trigger conditions are met, the global model runs an aggregation algorithm and trains itself based on the result.
5. Transmission: Global model encrypts the parameters and passes these to all other agents
6. Updation: The local agents update their corresponding models using parameters received.

Source: Federated Reinforcement Learning: Techniques, Applications, and Open Challenges (13)

Figure 2.5: Example of VFRL process and architecture
2.4 Applications of Federated Reinforcement Learning

2.4.1 FRL for communication networks

Traditional ML methodologies for managing communication networks are becoming less reliable with increased risk of data leakage and less efficient due to their centralized data processing architecture. With the rise of these concerns and the parallel development of FRL, researchers are studying the use of this privacy-enhanced method of ML in the realm of communication networks. The use of Hybrid Federated Deep Reinforcement Learning (HDRL) was explored in a device association scheme by Liu et al. (28). An FRL approach was also investigated in Network Function Virtualization (NFV), a component that allows for scalability and flexibility in communication network services (29).

2.4.2 FRL for attack detection

FRL approaches are also utilized in the area of attack detection. Detection models can be exposed to many different scenarios and create an optimized, powerful model. Mowla et al. presented a defence strategy against jamming attacks using FRL (30), and Wang et al. researched the use of FRL to create an anomaly detection strategy (31).

2.5 The Problem of Traffic

Traffic congestion is a serious problem in today’s urban society. If we take Dublin as an example, it is the most congested city in Ireland and the 16th most congested city in the world, according to an Inrix report. Having lost almost 246 hours in traffic in 2018 alone, Dublin drivers suffer the slowest traffic in Europe. With increasing vehicle usage rates and a lack of funding from governments to support the necessary development of public transport, these issues are only set to worsen for commuters to the city.

An increase in congestion leads to more wasted time and an increase in traffic accidents and is also a great benefactor to the greatest threat to our planet right now - global warming. Fuel consumption and exhaust emissions during rush hour periods can be up to 20 to 45% higher - leading to devastating environmental pollution. According to a case study of Brussels on the influence of traffic conditions on fuel consumption and emissions, we should re-orient the traffic management system to minimize fuel consumption and harmful emissions.
2.5.1 Ramp Metering

Ramp Metering is the installation of traffic lights on the on-ramps/entrances of dual-carriageways/motorways. This aims to control the number of vehicles entering the traffic flow on these roads. This would reduce congestion by breaking up any platoons that make it difficult to merge into the stream of traffic.

![Freeway Without Ramp Metering](image1)

![Freeway With Ramp Metering](image2)

**Figure 2.6: Comparison of mainline conditions with and without ramp metering**

**Source:** Washington State Department of Transportation

According to a report by the Federal Highway Administration of the U.S. Department of Transportation (32), benefits of Ramp Metering would include:

- **Increased Mobility, Reliability, and Efficiency:** Since ramp metering reduces overall congestion, it improves mobility on the dual-carriageway and the traffic throughout. A reduction in travel time is also noted, improving travel-time reliability.

- **Safety:** As ramp metering breaks up platoons of vehicles entering the dual-carriageway, there will be smoother merging manoeuvres and a significant reduction in crash rates.

- **Reduced Environmental Impacts:** Ramp metering eliminates prolonged periods of stop-and-go conditions, which reduces vehicle emissions and fuel consumption on
the road. The city of Minneapolis alone noted a net annual savings of 1160 tons of emissions after the implementation of ramp metering.

Currently, the two main means of ramp metering control include:

- **Fixed-Time Control** Installed in many large cities in the states, this method involves the use of pre-configured timings based on historical trends to meet local conditions. This is the easiest method to implement but is not adaptive to the real-time variations in traffic flow.

- **System Control** These systems are less popular due to the high installation and maintenance costs. However, system controlled ramp meters produce much better results as they are responsive to the flow, creating a more optimized solution.

Some of the algorithms used by System Control is described in the table 2.2

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Used By</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALINEA</td>
<td>Metering is based on the level of occupancy at detectors located downstream from the meter. The occupancy is maintained at a target value to reach target throughput</td>
<td>Other Algorithms</td>
</tr>
<tr>
<td>Corridor Adaptive Ramp Metering Program (CARMA)</td>
<td>Metering is based on the speeds on the motorway and local conditions. System will allow maximum vehicles on the ramp when speed is high and no vehicles when speed is near optimal level</td>
<td>Kansas City, Missouri</td>
</tr>
<tr>
<td>Heuristic Ramp Metering Coordination (HERO)</td>
<td>Metering using ALINEA as basis to manage traffic conditions. Once queues at ramps meet threshold conditions, control algorithm is activated, metering upstream ramps for downstream levels to meet target value.</td>
<td>Melbourne, Australia</td>
</tr>
<tr>
<td>Stratified Zone Metering (SZM)</td>
<td>Metering based on density measurements from detectors upstream of the ramp merge, mainline exit ramps and on the mainline. This is measured to sustain that the total vehicles exiting a zone is exceeding the total vehicles entering.</td>
<td>Minneapolis, Minnesota</td>
</tr>
<tr>
<td>System-Wide Adaptive Ramp Metering (SWARM)</td>
<td>Metering rates are based on the current density, the density required and the number of vehicles that should be added or removed to a zone between each ramp. Once a ramp has exceeded its capacity to balance the zone density, additional ramps will be activated. SWARM can be managed with different algorithms or manually.</td>
<td>Orange County, California. Portland, Oregon</td>
</tr>
</tbody>
</table>

Table 2.2: Description of algorithms used in Ramp Metering today

Source: Ramp Metering: A Proven, Cost-Effective Operational Strategy—A Primer (32)

### 2.5.2 Federated Reinforcement Learning in Traffic Control

Though FRL has not been used in Ramp Metering, it has been used for the control of other devices used in traffic. There are many devices used in traffic control. These include:

- **Monitor Agents:** These are components that monitor the traffic system to detect
situations and inform other agents to trigger an action to react to these situations, for example, in emergency events.

- **Sensor Agent:** Receives data which can then be processed and sent to another component, if necessary, for analysis.

- **Traffic Light Agent:** Manages the traffic control phases according to the traffic conditions.

Many studies of traffic control have applied FRL. In 2021, Nguyen et al. proposed a new traffic control framework using Federated Double Deep Q-learning. Federated Learning was used to improve the performance on top of the DDQN algorithm used. This research showed that the framework was successful in providing a consistent traffic granularity level (33). In 2018, an adaptive traffic signal control mechanism was proposed by Wang et al. using Multi-agent Reinforcement Learning and Federated Learning. An actor-critic method was used for the RL, and FedAvg was used as the aggregation method. This model showed that FL was very appropriate in a sparse traffic network, and FedAvg can improve the system’s capability. (34).

It is to be noted that all these studies have been conducted in the last five years.

### 2.6 Summary

This chapter presented the key topics and techniques used in this thesis. It introduced RL and the various categories of methods in RL while exploring the Q-learning algorithm and REINFORCE algorithm. Then the concept of FL was introduced, illustrating FL’s design and protocols, applications, and challenges. After this, the amalgamation of RL and FL, FRL, was explored. FRL is a novel concept and has very few applications as of yet. This chapter explored the designs proposed for implementing FRL and some applications.

The chapter also details the issue of traffic and how Ramp Metering can control traffic by exploring the benefits and different algorithms of ramp metering. The chapter concludes with a brief review of some existing FRL implementations on traffic control systems.
3 Design

This chapter outlines the design involved in creating the various components of the Traffic Control Problem as an FRL problem. This section will explore the algorithm used for RL and how FL will be incorporated into the algorithm.

3.1 Traffic Control Problem

The Traffic Control Problem is formulated as a Federated Reinforcement Learning Problem. Using the N7 national road as the state environment and implementing TLS (Traffic Light Systems) on compatible entrances to the N7 (entrances with junctions), the system will attempt to control the traffic intelligently by assessing the waiting time and occupancy. The occupancy is the percentage of time a detector was occupied by a vehicle. A detector is installed at the first lane after every entrance on the ramp. This is known as ramp metering (RM). Ramp metering is an active traffic management (ATM) strategy that manages the surge of vehicles merging into the mainline traffic on a dual-carriageway/motorway.

In this problem representation, the agent will look at the state at a time step, $t$, and use this information to take an action, $a$, which configures the TLS. After taking the action, $a$, the agent transitions to the next state and time step and, assessing the environment, receives a reward. This process will be detailed further below.

3.1.1 State Representation

States are a representation of the current environment of the task. The detectors placed on the ramp assess the current environment of the problem. The current environment of the problem can be assessed through the detectors placed on the ramp. These detectors will inform the system of the occupancy at the last time step.

Three states were defined:

1. **Low-level Occupancy**: 0 - 49% of the time vehicle present on the detector
2. **Mid-level Occupancy**: 50 - 79% of the time vehicle present on the detector
3. **High Occupancy**: 80 - 100% of the time vehicle present on the detector

These are the states in our State space, $S$. At each timestep, the environment will be in a state, $S_t$. Each of the states will be represented by a row in the Q-table, which will be used to store our Q-values as mentioned in section 2.1.1.

### 3.1.2 Action Space

The set of actions is as follows. $A = \{0, 1, 2\}$. Each number corresponds to the green, yellow and red phases of a TLS, respectively. The action $a$ selected by the environment will be from this set.

The epsilon-greedy method selects an action. This approach can achieve a balance between exploration and exploitation. With a probability of $\epsilon$, the algorithm will choose a random action and not exploit an action the algorithm has already encountered. This ensures that the algorithm discovers the optimal action quicker, without constantly repeating the same actions. However, the action with the highest Q-value is performed most of the time.
The implementation is as follows:

**Algorithm 1: Epsilon-Greedy Algorithm to choose Action**

**Data:** $Q$: Q-table currently generated, $\epsilon$: number between 0 and 1, $S$: current state

**Result:** Selected action

$n \leftarrow$ random generated number between 0 and 1;

if $n < \epsilon$ then

$A \leftarrow$ random action from action space;

else

$A \leftarrow \max Q(S)$;

end

return selected action $A$;

3.1.3 Reward Function

Defining the reward function is one of the more difficult tasks in RL. This function is the mechanism that informs the agent whether the action performed was appropriate or inappropriate given the scenario. Multiple different reward functions were evaluated at a particular TLS and its corresponding detector to select a suitable reward function.

One such reward function is as follows. Take $x_t$ to be the occupancy at the detector at time step, $t$. Take $y_{i,t}$ to be the $i^{th}$ vehicle’s waiting time at time step, $t$.

$$Y_t = \sum_i y_{i,t}$$

This reward function intends to represent the occupancy proportionally and the waiting time inversely proportionally. The proportionality is brought forth by $k$, a constant of proportionality.

Therefore, the occupancy can be represented as

$$X_t = k x_t$$

The inversely proportional waiting time can be represented as

$$W_t = \frac{k}{Y_t}$$

Therefore, the reward at time step $t$ is denoted by $R_t$ which can be summarized as the following:

$$R_t = X_t \ast W_t$$  \hspace{1cm} (1)
To put it simply, the reward function would reward a higher occupancy and punish a higher waiting time.

Another explored reward function was rewarding the agent based on the occupancy (i.e. based on the state).

\[
R_t = \begin{cases} 
-10, & x_t \leq 49 \\
10, & 50 \leq x_t \leq 79 \\
100, & 80 \leq x_t \leq 100 
\end{cases} 
\]  

(2)

This function would punish lower occupancy and reward higher occupancy, giving no regard to the waiting time.

A third equation was also evaluated that would reward based on the occupancy and waiting time difference. Therefore, the reward would increase as the occupancy increased and decrease as the cumulative waiting time increased. Hence the following reward equation was formulated:

\[
R_t = x_t - Y_t 
\]  

(3)

The table below shows the hyper-parameters that were used during the evaluation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>7</td>
</tr>
<tr>
<td>Time steps</td>
<td>177716</td>
</tr>
<tr>
<td>Exploration ( \epsilon )</td>
<td>0.1</td>
</tr>
<tr>
<td>Learning rate ( \alpha )</td>
<td>0.1</td>
</tr>
<tr>
<td>Discount rate ( \gamma )</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The definition of the parameters is as follows:

- **Episodes**: A sequence of states, actions and rewards that ends with a terminal state - in this example, there are no terminal states. Instead, an episode terminates at the final time step.

- **Time steps**: An interaction between the agent and the environment.

- **Exploration, \( \epsilon \)**: As described in 3.1.2, this is the probability that the agent explores an action instead of exploiting what it already knows.

- **Learning rate, \( \alpha \)**: This is the parameter that determines the speed at which the model learns the problem, traditionally configured to 0.1.
• **Discount rate**, $\gamma$: This parameter determines how much importance the agent gives to rewards in the distant future as opposed to caring about immediate rewards. The closer this value is to 0, the more it will care about immediate rewards.

Evaluation of the Reward Function 1 is seen below.

![Figure 3.2: Evaluation of Reward Function 1](image)

(a) Cumulative Rewards  
(b) Cumulative Waiting Time of Vehicles (s)

With each episode, the Q-table is further updated; therefore, rewards will be more accurate. With this reward function, the rewards decrease with each episode. The function may have over-rewarded initially and then began to align itself with each iteration. The cumulative waiting time of the vehicles is on a downward trend with each episode, which is the preferred behaviour.

Evaluation of the Reward Function 2 is seen below.

![Figure 3.3: Evaluation of Reward Function 2](image)

(a) Cumulative Rewards  
(b) Cumulative Waiting Time of Vehicles (s)

This reward function, though the cumulative rewards are on an upward trend and the cumulative waiting times are on a downward trend, produces very low rewards in comparison
to reward function 1. Though the agent shows the capability to learn with more episodes, this will mean that training will be more costly. Therefore, this reward is not favorable to reward function 1.

This reward function also satisfies the requirement of high rewards and decreasing cumulative waiting times. Though the cumulative rewards are on an upward trend, it is to be noted that the cumulative waiting times are higher after 7 episodes than they are for Reward Function 1.

Since there was a similar trend of results for Reward Function 1 and Reward Function 3, another round of evaluation was conducted to get a more vivid picture of which reward function should be used to converge the model faster.

This time a different set of parameters were used. The number of episodes was reduced to 5, and epsilon decay was used. The reduction in episodes was to see which reward function produces better results in cumulative rewards after a shorter training period. The use of epsilon decay was to determine if any reward function would converge more quickly with more exploration at the beginning of training in each episode and then, with a slow decay of the epsilon value, conform more to exploitation. This evaluation was performed on all agents, not just one TLS as in the first evaluation, to find if the trend applies to all agents and not just one as an outlier. Another difference between the first evaluation and this evaluation is that instead of the cumulative waiting time of vehicles per episode, the averaging waiting time of a vehicle per episode will evaluate the quality of the reward function. This figure will be easier to read as it is a smaller value and more comprehensible for evaluation.

The updated parameters are seen in Table 3.2

From Figure 3.5, it can be seen that after 5 episodes, 3 out of the 8 agents, produce an average vehicle waiting time greater than 0. These values range from 0.2(s) to 3.1(s). Each
Table 3.2: RL Reward Function Evaluation hyper-parameters (explicitly for Reward Function 1 and Reward Function 3)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>5</td>
</tr>
<tr>
<td>Time steps</td>
<td>177716</td>
</tr>
<tr>
<td>Exploration $\epsilon$</td>
<td>$1 \rightarrow 0$</td>
</tr>
<tr>
<td>Learning rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Discount rate $\gamma$</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Figure 3.5: Evaluation of Reward Function 1 with first plot showing average waiting time for a vehicle and second plot showing cumulative rewards.

An agent also shows a linear increase in cumulative rewards per episode. After 5 episodes, the cumulative rewards range between $0.1 \times 10^7$ and around $2.1 \times 10^7$. Though one might assume that the ideal expected result of a reward function is a decreasing average waiting time of a vehicle, it must also be noted that as cumulative rewards increase, it is also an indication of an increase in occupancy, meaning that there are more vehicles on the motorway, with only a slight increase in waiting times after 5 episodes of training. It is also to be noted that for 3 agents, the waiting time peaked at episode 2 and then decreased and didn’t peak higher than the result of episode 2. This is an indication that the agents are learning to reduce the waiting time while increasing occupancy.

Figure 3.6 illustrates the behaviour of Reward Function 3. From the first plot, it can be seen that apart from 3 TLS agents, all other agents have an average vehicle waiting time of 0(s). However, it is after episode 4 that the model started showing any signs of learning. As a result of this another 2 episodes to the original 5 episodes of training were run to get a
better idea of the behaviour of this reward function. From episode 4, 3 agents experience a
spike in the average waiting time of a vehicle and it indicates an increasing trend. Though
rewards are increasing, it is only after episode 4 of training. Compared with reward function
1, at the 5th episode, reward function 3 only produces a range of cumulative rewards from
around $0.1 \times 10^7$ to around $0.65 \times 10^7$, which is a far smaller range. Though after 5 episodes,
the range of waiting time for reward function 3 is smaller than 1, the increasing trend is not
a desirable feature.

Therefore, from this evaluation, Reward Function 1 is the most appropriate for this problem,
showing effective results.

### 3.1.4 Q-Learning Algorithm

Q-Learning is a simple RL algorithm that has been studied for the use of ramp-metering and
the results have shown a decrease in travel time and an increase in travel speeds (35). Also,
since the Q-table containing the state space and action space is quite small, and a large
amount of memory or computation power is not required by this problem, Q-learning, the
value-based method, seemed most appropriate for the scope of this research. The design of
the model is shown in the figure below.
Algorithm 2 along with Algorithm 1 implement Q-learning as described in 2.1.1. This algorithm will train not just one ramp meter, but all ramp meters on the N7. However, each ramp meter will be trained individually, learning only from its own individual environment, actions and rewards. A convergence check is conducted after each episode of training and if a ramp meter is considered to have converged it will no longer be trained. Only once this algorithm is inserted into a federated setting will the ramp meters also be able to learn from each other.

Algorithm 2: Epsilon-Greedy Q-Learning Algorithm

Data: $\alpha$: learning rate, $\gamma$: discount rate, $\epsilon$: exploration rate
Result: A Q-table containing Q(S,A) pairs defining optimal policy

/* Initialize Q-table to matrix of 0 */
$Q \leftarrow 0$;

/* For each step in each episode, the Q-value is calculated and the Q-table updated likewise */
for each episode do
  /* Initialize/Reset environment */
  Initialize/Reset Environment;
  for each timestep do
    $x_t \leftarrow$ number of vehicles at detector;
    Observe state $S$ /* Select action */
    $A_t \leftarrow$ Algorithm_1($x_t, \epsilon$);
    $Y_t \leftarrow$ sum of waiting times of each vehicle;
    /* Calculate the reward for action */
    $R_t \leftarrow$ reward($x_t, Y_t$);
    Observe reward $R_t$ and next state $S'$;
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a)] - Q(S, A)$;
    $S \leftarrow S'$
  end
  for each ramp_meter do
    ConvergenceCheck(ramp_meter)
  end
end
3.2 Federated Reinforcement Learning Techniques

Since the agents studied in this investigation are dispersed geographically but face a similar problem in their respective environments, HRFL technique is most suited. The design of HFRL in ramp metering is shown in Figure 3.8. The main component of the HFL is the aggregation method that the coordinator performs and sends to the global model. Federated Averaging (FedAvg) is the method being used in this thesis. Proposed by Google in 2017 (19), this is the most popular federated optimizer.

![Figure 3.8: Horizontal Federated Reinforcement Learning design in Ramp Metering](image)

The steps to this method are as outlined below:

1. The coordinator initializes a global model.
2. The global model is sent to all agents. The data from the global model is used to update each agents local model.
3. Each agent trains its local model using the Q-learning algorithm.
4. Once an episode of training is finished, the coordinator randomly selects a subset of agents’ data.
5. The coordinator averages the (assumed to be encrypted) parameters of these agents (Q-tables).
6. The aggregated data is used to update the model.
7. Steps 2 - 6 is repeated until all episodes of training is completed.

**Algorithm 3: Federated Epsilon-Greedy Q-Learning Algorithm**

**Data:** $\alpha$: learning rate, $\gamma$: discount rate, $\epsilon$: exploration rate  
**Result:** A Q-table containing Q(S,A) pairs defining optimal policy

/* Initialize Q-table to matrix of 0 */  
$Q \leftarrow 0$;  
/* Initialize Federated Coordinator & Global model */  
$C \leftarrow$ Federated Coordinator;  
$G \leftarrow$ initialize Global model;  
/* For each step in each episode, the Q-value is calculated and the Q-table updated likewise */  
for each episode do  
  for each device $D$ in network do  
    $D \leftarrow G$;  
    /* Initialize/Reset environment */  
    Initialize/Reset Environment;  
    for each timestep do  
      $x_t \leftarrow$ number of vehicles at detector;  
      Observe state $S$;  
      $A_t \leftarrow$ Algorithm\_1($x_t, \epsilon$)/* Select action */  
      $Y_t \leftarrow$ sum of waiting times of each vehicle;  
      /* Calculate the reward for action */  
      $R_t \leftarrow$ reward($x_t, Y_t$);  
      Observe reward $R_t$ and next state $S'$;  
      $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \text{max}_a Q(S', a)] - Q(S, A)$;  
      $S \leftarrow S'$;  
    end  
  end  
  fedAvg($C, G$);  
end

### 3.3 Summary

This chapter presented the design and the decisions required to implement the design of an FRL agent in a ramp metering problem. It presented the state space, action space and reward function (through its evaluation) of the RL component of the design. The algorithms and the techniques used to implement Q-Learning into this model were also described. Then, it illustrated the architecture used for FRL in this thesis and the algorithm and aggregation algorithm required to integrate RL into FL and create FRL. The next chapter will present the implementation of this design.
4 Implementation

This chapter outlines the implementation of FRL in ramp metering using SUMO as the simulation environment. The functions, parameters and network between classes will be detailed below.

4.1 Simulation Environment

In order to simulate the environment the agent will be trained on, SUMO was used. SUMO (Simulation of Urban MObility) is an open source, highly portable, microscopic and continuous multi-modal traffic simulation package designed to model large networks (37). SUMO can be used to evaluate the performance of Intelligent Traffic Systems. Since the project is being built using Python, there is a Python API, TraCI, that was also used to interact with the simulator. Through this API, information on components (such as traffic lights, vehicles, detectors, lanes etc.) can be accessed and manipulated.

SUMO is a package that includes two modules - NETEDIT and SUMOconfig. NETEDIT is an interface that allows for definition and manipulation of traffic components, such as TLS and induction loops (e1 detectors). SUMOconfig provides users the ability to define simulation information such as simulation period, waiting time etc. Vehicle emitter and detector files can be generated to assist in simulating the flow of traffic. The traffic load and various scenarios can be emulated with SUMO as well.

As mentioned in *REF*, the ramp metering control will be simulated on the N7. Gueriu et al. have created the network for the N7 with the vehicle and road network files already generated (38). The components needed to implement ramp metering were added onto this network.

4.1.1 Components

To implement ramp metering, the two components needed to configured through SUMO are:

- **Traffic Signal Heads** The signal heads typically used for ramp metering are
two-sectioned heads (with only red and green lights) or three-sectioned heads (with red, green and yellow lights). Three-sectioned heads were implemented in this design as they are the most common signal heads. They were initially installed with fixed timed phases however, this will be manipulated as needed with the system being used.

- **Detectors/Induction Loops** Detectors are used to monitor the flow on the ramp. The TLS/Signal head will change to a green phase when a vehicle is detected on the induction loop. The occupancy (percentage of time the detector is occupied by a vehicle) will be used to implement the reward function and the system will receive this data through the detector.

4.1.2 TraCI

Traffic Control Interface (otherwise known as TraCI) allows users to gain access to a running simulation, giving them the option to retrieve, send and manipulate information.(37)

The implementation used the following commands from TraCl:

- **start()**: Opens connection to client port, starts simulation on SUMO
- **close()**: Closes connection, stops simulation and shuts down SUMO
- **simulationStep()**: A step in simulation, taken as the timestep for this model.
- **simulation.getMinExpectedNumber()**: This returns the minimum number of vehicles to leave the network, including the vehicles that have yet to start on their route. Simulation/Training runs until this value reaches 0.
- **trafficlight.getPhase(TL_id)**: Returns the current phase of the TLS Agent. Used in calculating transitions and changing values in the Q-table
- **trafficlight.setPhase(TL_id, phase)**: Set the phase given the id of the TLS Agent and the phase.
- **inductionloop.getLastStepOccupancy(det_id)**: Returns the percentage of time the detector was occupied by a vehicle in the last time-step
- **inductionloop.getLastStepVehicleNumber(det_id)**: Returns the number of vehicles at the detector in the last time-step (used in calculating averages)
- **inductionloop.getLastStepVehicleIDs(det_id)**: Returns the ids of the vehicles on the detector at the last time-step (used to retrieve individual vehicle values)
- **vehicle.getWaitingTime(veh_id)**: Returns the consecutive time-steps in which this vehicle was immobile.
• vehicle.getCO2Emission(veh_id): Returns the vehicles CO2 emissions within the last time-step
• vehicle.getSpeed(veh_id): Returns the speed of the vehicle within the last time-step

4.2 Federated Reinforcement Learning

Following the design described in Chapter 3, Algorithm 3 (which is Algorithm 1 and Algorithm 2, combined with FL) was implemented using the following objections and functions.

4.2.1 Objects

To implement this technique, two class objects had to be defined:

• Q_Learning_Model: This class contains the parameters and methods needed to implement Q-learning functionality. Each ramp meter TLS is considered an agent (object) and the parameters needed for each agent is identified below:
  – TL_id: The id of the TLS
  – det_id: The id of the detector used by the ramp meter object
  – environment_rows: The rows in the q-table representing the states
  – environment_columns: The columns in the q-table representing the actions
  – waiting_times_per_episode: An array of waiting times collected during an episode
  – rewards_per_episode: An array of rewards collected during an episode
  – waiting_times: An array of cumulative waiting times for each episode of training
  – rewards: An array of cumulative rewards for each episode of training
  – q_values: A matrix initialised to 0 in the shape of (environment_rows, environment_columns) that contains the Q-values as a Q-table

• FederatedCoordinator: This class contains the parameters and methods needed to implement federated learning functionality. There is one FederatedCoordinator for the system:
  – models: An array of pointers to all local models that the coordinator has access to.
4.2.2 Functions

Each class had functionalities to be implemented through methods/functions. There were also some other functions outside of these two classes, in the main Python file, to capture the results of the experiments. These methods will be described in detail below:

- **Q_Learning_Model**:
  - `_init_(self, TL_id, det_id)`: This is the constructor method for the ramp meter TLS agent, initialising all the properties for the agent using the TLS ID (TL_id) and the detector ID (det_id) as parameters for identification.
  - `get_next_action(self, current_row_index, epsilon)`: This defines an epsilon greed algorithm defined in Algorithm 1 which will choose which action to take next from the Q-table, using the current state which is passed in as the current row index (current_row_index) and the epsilon value (epsilon)
  - `get_next_location(self)`: This function will get the next state of the agent based on the result of the action that was performed. Based on the occupancy levels described in Section 3.1.1, the next state is decided and stored as an integer that represents the row index of the Q-table.

- **FederatedCoordinator**:
  - `_init_(self, models, global_model)`: This is the constructor method for the Federated Agent initialising the Global Model (global_model) and all other properties of the Federated Coordinator.
  - `fedAvg(self)`: This method carries out the aggregation method used to calculate the average of the models results as described in *INSERT REF HERE* and updating the Global Model and feeding this model to each local Q_Learning_Model for it to be updated.

- **Main Class**: This is where the functions used to run the entire program reside.
  - `main()`: Here, SUMO is initialised and the local models as well as the Federated Coordinator are initialised. The first simulation and the call for the experiments also reside in here.
  - `run(models, global_model)`: This method runs the training for the model, making necessary calls to some of the functions above in order to complete the
training.

- reset_simulation(): This method resets the simulation after the completion of each episode in training.

The above can be summarized with the following UML class relationship diagram:

![UML Class Diagram for FRL Implementation](image)

Figure 4.1: UML Class Diagram for FRL Implementation

### 4.3 Summary

In this chapter, the implementation of the algorithms and design presented in Chapter 3 were detailed. The application of the simulation environment SUMO, its components and TraCI were also presented. The FRL model implemented for this study contains 2 different
classes, \textit{Q\_Learning\_Model}, which implements the RL aspect and the \textit{FederatedCoordinator}, which implements the FL aspect. The attributes and functions of each of these classes are also described. Finally, a UML diagram (5.1) was provided to visualise the classes and their interaction with the main program and TraCI.
5 Evaluation

This chapter evaluates the performance of FRL as a solution for Ramp-Metering. The different metrics, evaluation techniques and scenarios will be explored. The experiments and results will be analysed in detail as well.

5.1 Objectives

The objective of the evaluation is to determine how the use of FRL in ramp metering performs in comparison to other commonly used methods. The main objective of the design of FRL was to provide a novel approach on RL in the optimization of ramp metering. In Chapters 3 and 4, the design and implementation of such a system was addressed. However, FRL can only be proven successful if it performs (or outperforms) against other commonly used techniques in responsive ramp metering in various scenarios. FRL can be concluded as successive if it satisfies the following requirements:

- FRL outperforms existing static Ramp Metering techniques within the scope of this thesis.
- FRL outperforms existing RL techniques in regards to training time and performs at least in an equal manner.
- FRL is not impacted by communication challenges presented by FL

5.2 Metrics

Two metrics that are popular measurements of evaluation are cumulative rewards (39) and average vehicle waiting times (40)(41). However, since a non-RL method is being evaluated as well, cumulative rewards cannot be applied to all models, therefore it will not be measured. The average CO2 emissions from vehicles and the average speed of vehicles are also used as metrics. These metrics are used in each experiment conducted after every episode. An episode consists of x time steps and each time step, t, amounts to 1 simulated second. The metrics mentioned are defined as the following:
• **Cumulative/Average Waiting Times:** This metric is used to measure the effect of the reward function applied by the model in the given scenario. Though average waiting times are used, to scale down the rewards in training, cumulative waiting times are used. The goal of the agent and this system is to reduce the waiting time and this metric will be able to evaluate whether this goal can be achieved.

• **Average CO2 Emissions/Average Vehicle speed:** According to (32), Ramp Metering is shown to have reduced CO2 emissions and increased the average vehicle speed. Therefore, this needs to be investigated in the evaluation to conclude what benefits the system has to the environment.

### 5.3 Evaluation Scenarios

In this section, the scenarios used to evaluate the suitability of FRL in ramp metering are described. The techniques and individual scenarios will be detailed below.

#### 5.3.1 Evaluation Techniques

- **Ramp Metering with Fixed Time.** The traffic lights within a ramp meter changes phases according to a fixed time. There is no adaptation to traffic load or training involved. Red phases take $x$ seconds, yellow phases $y$ seconds and green phases $z$ seconds. This method is commonly used in ramp metering. This will be considered as the baseline.

- **Ramp Metering with Q-Learning.** This evaluates a more intelligent method of ramp metering that is also used in the industry.

- **Ramp Metering with Federated Reinforcement Learning.** A new method to the industry, the objective of this thesis is investigated and compared with the above techniques.

#### 5.3.2 Scenarios

Each technique will be evaluated in the following scenarios:

- **Low traffic load.** This is a scenario that depicts low traffic load on the N7. Data is estimated from (42) where the number of vehicles is reduced. This would represent the normal flow of traffic at off-peak hours. The busiest times on the N7 on the busiest day was used to model the data.

- **High traffic load.** This is a scenario that depicts high traffic load on the N7. Data is estimated from (42) where the number of vehicles is extremely high. This would
represent the normal flow of traffic during peak-hours. The lowest traffic time between 6am and 9pm on the least busiest day of traffic was modelled. A time between these hours were chosen as these are normal times where people travel. Outside of these hours, there is not much travel being conducted on the N7.

- Mixed traffic load. This is a scenario that depicts a mixed level of traffic load on the N7. Data is again estimated from (42) where the number of vehicles is an average flow or mid-peak. Data was taken from a weekday in-between office hours to model traffic on an average day.

5.4 Setup

5.4.1 Network Layout

The N7 network has 8 entrances (on-ramps) where a traffic signal head can be installed. Signal heads were configured here with a fixed timing of 80 seconds on green, 10 seconds on yellow, 30 seconds on red placed on the N7 Eastbound and Westbound entrances as shown in the figure below.

![Figure 5.1: N7 Network Layout for FRL](image)

5.4.2 Traffic Demand

The traffic demand for the different evaluation scenarios were generated through SUMO. Two vehicle types were defined, passenger vehicles (representing light-weight load) and trucks (representing high-weigh load). Each scenario was conducted for one hour which equates to 3600 time-steps. For low traffic load. For low traffic load, around 2900 vehicles were loaded onto the network. For high traffic load, around 9200 vehicles were loaded onto
5.5 Results & Analysis

Here, the performance of FRL is analysed with regard to the evaluation objectives described in Section 5.1. The efficiency of the FRL model against Q-learning and Fixed Timed ramp metering will be examined in three scenarios described in Section 5.3.2. Given the parameters of each model, the results will be analysed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Episodes</td>
<td>5</td>
</tr>
<tr>
<td>Time steps</td>
<td>87000</td>
</tr>
<tr>
<td>Learning Rate $\alpha$</td>
<td>0.1</td>
</tr>
<tr>
<td>Exploration $\epsilon$</td>
<td>1.0 → 0</td>
</tr>
<tr>
<td>Exploration decay</td>
<td>0.995</td>
</tr>
<tr>
<td>Discount Factor $\gamma$</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 5.1: Parameters of FRL and Q-Learning model training

Some points to note:

- Each local model/TLS agent is plotted as a different colored line.
- The speed limit on the N7 is set at 100 km/h
- There may be some discrepancies in graph titles, please disregard this and take sub-figure caption as graph title.

5.5.1 Low Traffic Load

In this section, the results of the three different techniques, FRL, Q-Learning RL, and Fixed timing in a low traffic load, will be evaluated.

Figures 5.7a to 5.7c are the results from the baseline Fixed Timing model. The waiting time is observed to be a flat line at 0 seconds, indicating that there are no consecutive seconds with a vehicle at a standstill. The majority of average vehicle speed values ranges between 90-100, with some time steps showing average 20 above or below the speed limit. CO2 emissions are at around 50g/s for the most part with a few spikes to around 125g/s. This is as expected at low traffic load.

Looking at Q-Learning’s performance in this scenario (from Figures 5.3a to 5.3c), the results does not show variation. This can be expected due to the model’s performance. Since there is very little traffic, much difference in results may not be seen. The same can be said for the FRL model as shown from Figures 5.4a to 5.4c.
5.5.2 High Traffic Load

Here, the above techniques are evaluated in a High Load setting. The baseline model (Figures 5.5a to 5.5c) shows much more variation in traffic. Average CO2 emissions are peaking at nearly 300g/s and this occurs more regularly than in low load traffic. The average speed is much slower at 2 detectors, with what can be described as traffic jams, given the average waiting time. The average vehicle is seen to be waiting at around 80 seconds in traffic at these two detectors.

When other two models, the model of research, FRL, and the Q-learning model are evaluated, identical results can be seen again. This raises the concern that there is an error present in the design or implementation of the research. There may be three explanations for this:

1. Baseline is Optimal model: The baseline model, the model of fixed timing, may be the most optimized model and the other models may also be optimized giving rise to identical results.

2. QL parameters have not be tuned: The parameters found in QL have not been tuned as necessary and therefore have not learned as much from training.

3. TLS in SUMO is incorrectly configured: This may be the case as the TLS is being updated at every second in this implementation. This may render the traffic light ineffectual and the signal can be read by the vehicles as a result of it constantly changing.

Apart from the first reason, the other two reasons give cause for a re-evaluation of this model. However, given that each model takes about 15 - 24 hours to train, and the time restraints in place, this was not possible. Nonetheless, these points should be taken into consideration for future study in this area of research.

5.5.3 Mixed Traffic Load

Mixed load again provides identical results from all three techniques. With speeds ranging around 80-100km/h and no waiting time, this result, like the other scenarios seems appropriate for the load of traffic. However, the identical results as shown from Figures 5.8a to 5.10c, are not satisfying the objectives of this evaluation.

5.6 Further Evaluation Notes

The training time for each model is worth noting.
The fixed models did not require any training and therefore it is the fastest form of ramp metering control.

The Q-Learning method only converged one of the models in 5 episodes. It also took an average of 20 hours to train this model.

The FRL model converged within the 5th episode and only took around 10 hours of training.

Given this information, it can be concluded that FRL does outperform Q-learning in terms of training speeds, converging at a much faster rate.

5.7 Evaluation Summary

In this chapter the evaluation design was outlined, providing the objectives, metrics and scenarios. Three techniques, namely, Fixed Timing, Q-Learning and FRL were evaluated for ramp metering. From the results, it can be concluded that this design is not appropriate as it does not provide any learning to the adaptive (Q-Learning and FRL) models. Further work is need to improve this design to create an adaptive model. However, given training times and speed of convergence, it can be said that FRL outperforms Q-learning.
Figure 5.2: Fixed Timing Results in Low Load Traffic (Baseline Results)
Figure 5.3: Q-Learning Results in Low Load Traffic

(a) Average CO2 Emissions of Vehicles (g/s)

(b) Average Speed of Vehicles (km/h)

(c) Average Waiting Time of Vehicles (s)
Figure 5.4: FRL Results in Low Load Traffic

(a) Average CO2 Emissions of Vehicles (g/s)

(b) Average Speed of Vehicles (km/h)

(c) Average Waiting Time of Vehicles (s)
Figure 5.5: Fixed Timing Results in High Load Traffic (Baseline Results)
Figure 5.6: Q-Learning in High Load Traffic
Figure 5.7: FRL Results in High Load Traffic

(a) Average CO2 Emissions of Vehicles (g/s)

(b) Average Speed of Vehicles (km/h)

(c) Average Waiting Time of Vehicles (s)
Figure 5.8: Fixed Timing Results in Mixed Load Traffic (Baseline Results)
Figure 5.9: Q-Learning Results in Mixed Load Traffic

(a) Average CO2 Emissions of Vehicles (g/s)

(b) Average Speed of Vehicles (km/h)

(c) Average Waiting Time of Vehicles (s)
Figure 5.10: FRL Results in Mixed Load Traffic

(a) Average CO2 Emissions of Vehicles (g/s)

(b) Average Speed of Vehicles (km/h)

(c) Average Waiting Time of Vehicles (s)
6 Conclusion & Future Work

In this chapter, a conclusion is provided for the thesis. The critical contributions of the research and any other areas of study will be discussed.

6.1 Thesis Contribution

This thesis is the only work to address the application of FRL in ramp metering.

Chapter 1 outlines the motivation for this work by introducing traffic problems and the use of ramp metering to reduce the effects of these problems on highways. Federated Reinforcement Learning is yet to be applied to this problem area and given its features of privacy and faster training of models, it seemed like a feasible area of research. Taking this into consideration, the thesis aims and assumptions were defined.

Chapter 2 explores the background of this thesis. It describes the topic of Reinforcement Learning and the different categories of RL, while exploring different applications and related works of this area. Federated Learning is a very new concept introduced by Google. Therefore the various architectures, applications and open challenges of FL were studied. Finally, the different architectures of FRL were introduced and, after this a detailed study of current ramp metering systems was conducted.

The design of the idea proposed by the thesis was outline in Chapter 3. It provided the state and action space and the reward function. The chapter details the different algorithms needed to implement this. Then the implementation was illustrated in Chapter 4. Here, information was given about the simulation environment, the selected network, and objects used.

Chapter 5 explains the evaluation conducted for the proposed design. It describes the various metrics, scenarios and techniques involved in the evaluation process. Then a detailed analysis of the results is conducted. The results are appropriate for each scenario (low, high and mixed load) provided. However, neither the FRL system nor the Q-learning model outperforms the Fixed algorithms. In fact, the results are identical. From these results it can be assumed that there may be an error in the implementation of the design. The corrections
needed for future work in this area will be explored below. However, a difference in training times must be noted between FRL and Q-learning. FRL was much quicker in training and converged all models within 5 episodes. This is note of success, in that it can be concluded that the FRL outperforms an existing RL method of ramp metering control.

6.2 Future Work

After the design and evaluation of the design of FRL in ramp metering, a few areas where this could develop and design corrections were discovered.

6.2.1 Corrections

- **Parameter tuning of exploration, learning, and discount:** When implementing this design, commonly used values for these parameters was used for this design of FRL. This decision may not have been the most appropriate to make. These parameters have a key role in how the model learns, and they need to be finely tuned. This has the potential to produce an optimized model quickly.

- **Evaluating aggregation method in FL:** Just as the reward function was evaluated, the aggregation method chosen (FedAvg), along with other methods of aggregation, should have been studied and evaluated. How the local models’ parameters are aggregated is very important to FL and, as a result, to FRL. It determines what the global model learns from the parameters and, in turn, what the local models will learn. Therefore, in future implementations, the evaluation of this method needs to be conducted.

- **Configuring the Traffic Light Signals:** It was during the evaluation of the design that this issue was discovered. The traffic light signals were being configured and set every second. This design, in reality, would cause many issues, as a continuously changing traffic light can be considered purposeless, unusable and faulty. Drivers would find it very difficult to interpret these signal heads. Therefore, the signal heads should only set the phase at regular intervals during the subsequent implementation. After analyzing the average level of traffic between the last time the traffic light phase was set and the current time, it should choose an action.

6.2.2 Areas of Development

- **Varying Traffic Load:** This study was conducted with focus on one level of traffic in each scenario and for only one hour. Further studies should evaluate models on a day of traffic, with low, high and mixed (average) load. This evaluation would measure how the design can perform throughout an average day of traffic.
• **Different metrics:** Using different metrics such as ramp queue length, average vehicle travel time to calculate the reward and evaluate the model may provide better, faster optimization of the ramp metering system.

• **Homomorphic Encryption:** Though the idea homomorphic encryption (HE) was first proposed in 1978, the concept is only gaining substantial development in recent years. Homomorphic Encryption aims to allow operations to be conducted on encrypted data (43). For example, in relation to the study of this thesis, HE would allow for the local models to send encrypted data (or parameters) to the global model, which would not have to decrypt it to perform the aggregation operation on the data. From this, the FRL concept can gain more data protection and privacy.

• **Design Cost Reduction:** Though many studies have proved how FRL and other intelligent systems can improve the efficiency of processes it fails to mention the cost of implementing these systems and use that as a comparison. Studies of the use of adaptive ramp metering systems (32) mention that even though the newly proposed systems may be more efficient, the cost of installing these systems is not encouraging. Therefore, research into building economical models for ramp metering will be beneficial.
Bibliography


[29] Haojun Huang, Cheng Zeng, Yangmin Zhao, Geyong Min, Yingying Zhu, Wang Miao, and Jia Hu. Scalable orchestration of service function chains in nfv-enabled networks:


