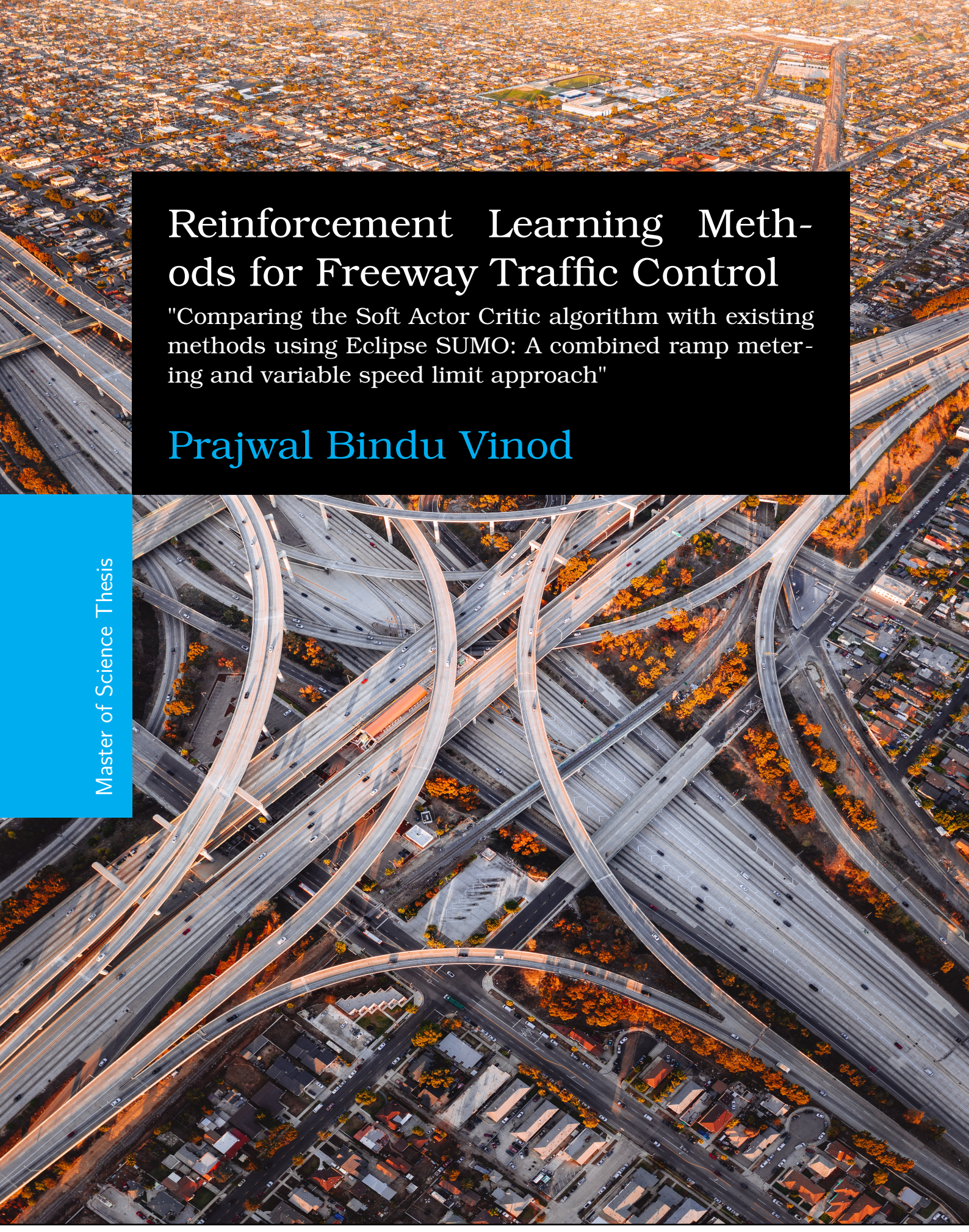


Reinforcement Learning Methods for Freeway Traffic Control

"Comparing the Soft Actor Critic algorithm with existing methods using Eclipse SUMO: A combined ramp metering and variable speed limit approach"

Prajwal Bindu Vinod

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Reinforcement Learning Methods for Freeway Traffic Control

"Comparing the Soft Actor Critic algorithm with existing methods using Eclipse SUMO: A combined ramp metering and variable speed limit approach"

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Prajwal Bindu Vinod

June 2, 2025

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Rapid and continuous advances in communications and computer technology are spurring a host of new concepts in road traffic control. From simple traffic control measures like lane segregation for wheeled and pedestrian traffic in the 14^{th} century to the use of artificial intelligence to control traffic, we have come a long way. The advancement of technology in the automotive sector along with the industrial revolution led to the number of vehicle owners seeing a drastic increase in the 20^{th} century. There was a growing need for infrastructure to handle this new volume of vehicles which led to the construction of several road networks in both urban and freeway settings. Adding infrastructure was a valid method to handle traffic problems until we realized that each new addition required a significant amount of land that was slowly reducing. This brought along the need for newer and more efficient traffic control techniques due to increased vehicles on the road, the different kinds of vehicles itself and the need to reduce construction of roads as a means of reducing traffic.

In this thesis report, we discuss the different control methods using reinforcement learning to tackle the freeway traffic control problem. The thesis covers the fundamentals of freeway traffic control, reinforcement learning and the agents used for control. It focuses on the creation of the freeway network, environment setup for reinforcement learning application and the choice of agents mainly SAC in order to implement continuous actions to improve combined ramp metering and variable speed limit control in more complex scenarios.

Important terms- **Freeway traffic control, ramp metering, variable speed limits, reinforcement learning, SAC**

Table of Contents

Preface xi

Acknowledgements xiii

1 Introduction 1
- 1-1 Motivation . . . 2
  - 1-1-1 Enhancing Road Safety . . . 2
  - 1-1-2 Reducing Traffic Congestion . . . 3
  - 1-1-3 Environmental Sustainability . . . 3
  - 1-1-4 Economic Efficiency and Productivity . . . 3
  - 1-1-5 Technological Advancements and Innovation . . . 3
- 1-2 Research . . . 4
- 1-3 Objectives . . . 5
- 1-4 Thesis Outline . . . 5

2 Background 7
- 2-1 Reinforcement Learning Algorithms . . . 7
  - 2-1-1 Reinforcement Learning and Markov Decision Processes . . . 8
  - 2-1-2 Deterministic Policy Gradient . . . 11
  - 2-1-3 Deep Deterministic Policy Gradient . . . 12
  - 2-1-4 Twin Delayed Deep Deterministic Policy Gradient . . . 13
  - 2-1-5 Soft Actor-Critic . . . 14
- 2-2 Effects of Hyperparameters on DDPG and SAC . . . 16
  - 2-2-1 Brief Overview of DDPG and SAC . . . 17
  - 2-2-2 Key Hyperparameters and Their Effects on DDPG Training . . . 17
  - 2-2-3 Key Hyperparameters and Their Effects on SAC Training . . . 18
- 2-3 Impact of Hyperparameters Based on RL Environment . . . 19

2-3-1 Environment Dynamics . . . 19
2-3-2 Reward Structure . . . 19
2-3-3 State and Action Space Complexity . . . 19
2-3-4 Stochastic vs. Deterministic Environments . . . 20
2-4 Reinforcement Learning in Freeway Traffic . . . 20
2-4-1 Freeway Traffic Model as Markov Decision Process . . . 23
2-4-2 Review of Related Literature . . . 24

3 Combined RM-VSL using Reinforcement Learning 29
3-1 Outline of Problem Statement . . . 29
3-2 Proposed Methodology . . . 29
3-2-1 Network Description . . . 29
3-2-2 Demand Profile . . . 30
3-3 Environment Setup . . . 32
3-3-1 Observation Space . . . 32
3-3-2 Action Space . . . 32
3-3-3 Reward . . . 32
3-3-4 Important Features in SUMO Traffic Environment . . . 33
3-3-5 Commands and Functions . . . 34
3-4 Reinforcement Learning Agent Setup . . . 36
3-4-1 RL Agents . . . 36

4 Simulations and Results 37
4-1 Baseline . . . 37
4-1-1 No Control vs Fixed Time Control . . . 37
4-2 ALINEA . . . 39
4-3 Reinforcement Learning . . . 41
4-3-1 DDPG . . . 42
4-3-2 SAC . . . 44
4-4 Comparison and Final Results . . . 47

5 Conclusion 53
5-1 Summary of Results . . . 53
5-2 Analysis of Results . . . 54
5-2-1 Reward Function . . . 54
5-2-2 Continuous Control . . . 54
5-2-3 Conflict between Objectives . . . 55
5-2-4 Demand Modelling . . . 55
5-2-5 RL Agent and SUMO Limitations . . . 55

6 Further Work 57

A Appendix A **59**
A-1 DDPG 60
A-2 TD3 61
A-3 SAC 62

B Appendix B **63**
B-1 Lemma 1 63
B-2 Lemma 2 63
B-3 Theorem 1 64

Bibliography **65**

Glossary **69**
List of Acronyms 69
List of Symbols 69

List of Figures

2-1 Agent-Environment Interaction in an MDP 8
2-2 Freeway Link 22

3-1 Full Network 30
3-2 Junction 30
3-3 Demand Profile in SUMO 31

4-1 Metrics for No Control 38
4-2 Metrics for Fixed Time Control 39
4-3 Metrics for ALINEA Control with High Demand 40
4-4 ALINEA Ramp Rate for High Demand 40
4-5 Metrics for ALINEA Control for Lower Demand 41
4-6 ALINEA Ramp Rate for Lower Demand 41
4-7 Metrics for DDPG (RM) 43
4-8 Control Input for DDPG (RM) 43
4-9 Metrics for DDPG (RM-VSL) 44
4-10 Control Input for DDPG (RM-VSL) 44
4-11 Metrics for SAC (RM) 45
4-12 Control Inputs for SAC (RM) 45
4-13 Metrics for SAC (RM-VSL) 46
4-14 Control Inputs for SAC (RM-VSL) 46
4-15 DDPG Learning Curves 48
4-16 SAC Learning Curves 48
4-17 Average Speed 49
4-18 Delay on Mainstream 49
4-19 Queue Length 49

4-20 Total Time Spent 50
4-21 Average Speed across the Network 51
4-22 Delay on Mainstream 52
4-23 Queue Vehicles on the Ramp 52

List of Tables

2-1 Summary of Reinforcement Learning papers for Freeway Traffic Control 24

3-1 Demand Profile 31
3-2 Formulae 35
3-3 TraCI Value Retrieval 36

4-1 Values No Control 38
4-2 Values FTC 38
4-3 Values ALINEA 40
4-4 Values ALINEA for Lower Demand 40
4-5 Results DDPG (RM) 42
4-6 Results DDPG (RM-VSL) 43
4-7 Results SAC(RM) 45
4-8 Results SAC (RM-VSL) 46
4-9 Hyper-parameters for RL Agent Training 47
4-10 Final Values over 300 Iterations 50

Preface

Traffic control has been a growing area of research ever since the industrial revolution and increasing vehicles on the road. Freeway traffic control came into prominence when engineers realized that just building infrastructure could not solve all the traffic problems. More freeways meant more space required for construction and more energy spent on possibly ineffective infrastructure. Thus freeway traffic control became an area of research where in engineers worked to make traffic on freeways and subsequently freeway networks flow more efficiently. The idea for the thesis was first brought up in a meeting between Dr.Dabiri and myself where she suggested the topic of reinforcement learning in freeway traffic control and we discussed the possible directions and potential for the thesis. As someone who has stayed in the Netherlands for two years and having seen the transport system that has been adopted here along with my interest in AI, I felt a connection which I wanted to pursue. The complete idea came into being after a meeting with Dingshan Sun, a post Doctoral fellow in the Civil Engineering department. This thesis report will deal with the application of reinforcement learning agents mainly SAC for freeway traffic control. The report also includes the additional issue of dealing with continuous action spaces in the freeway traffic control problem. The aim of this report is to show that continuous action spaces are viable and that the choice of SAC as the RL agent to control the combined ramp metering and variable speed limit problem is justified.

Acknowledgements

First, I would like to thank my daily supervisor Dinghsan Sun for his assistance during the writing of this literature report. He was always available to clear any doubts, help with understanding results and providing feedback on the content of the report. I would also like to thank him for constantly checking in to monitor the progress of the report which made me feel more more confident in what I was doing.

I would also like to thank my professor and thesis supervisor Dr. Azita Dabiri for her guidance in framing this report and to improve the quality of the simulation and results. Her advice helped immensely in setting the direction of the thesis towards an achievable yet comprehensive goal which helped me a lot over the thesis duration.

Additionally, I would like to thank them both for their patience in dealing with my own situations through the course of the thesis without which it would have been impossible for me to come this far.

I would also like to thank my parents who have stood by me through this tough time without complaint. Their support means the utmost and this thesis is a testament to that support. I would also like to thank Tanay Naik, a former TU Delft Systems and Control student and fellow batch mate from India for being a friend through and through and checking up on me.

Last but not the least, I would like to hail Pickwick and Nescafe whose beverages have gotten me through these past two years.

Delft, University of Technology — Prajwal Bindu Vinod
June 2, 2025

"Education's purpose is to replace an empty mind with an open one."
— *Malcolm Forbes*

Chapter 1

Introduction

From the advent of mankind, learning through experience and interaction has been a natural instinct for our species. Most innovations took place through observations and interaction with the environment and a rigorous process of trial and error. As an infant left to its devices, they have no explicit teacher or knowledge of the world but it does have a direct sensorimotor connection to its environment. Through this connection we gain a wealth of information about cause and effect, actions, their nature and the consequences and the know how on performing specific tasks to achieve certain goals. These interactions provide the foundation for learning and are an inexhaustible source of knowledge and information as they happen throughout the lifespan of the human species. Regardless of the actions undertaken by humans, we are extremely aware and observant of how our environment responds and in-turn we seek to influence what happens through our behaviour. Learning from interaction is a foundational idea underlying nearly all theories of learning and intelligence.

Reinforcement learning at its core is learning how to map situations to actions in order to maximize a numerical reward. This type of learning is showcased wherein the learner must discover which actions lead to best reward by trial and error and not through direct instructions. The choice of actions can not only determine the immediate reward but also consequently affect subsequent rewards which can vary from a case to case basis. The core distinguishing features of reinforcement learning can be said to be - trial and error approach and delayed rewards - as described above.

Reinforcement learning is considered to be the third paradigm of machine learning and must not be grouped with supervised or unsupervised learning. The fundamental difference between supervised learning and reinforcement learning lies in the fact that the former uses a set of examples provided by an external supervisor and learns through the set. The objective of supervised learning is to extrapolate or generalize the structure learned from the examples to instances not present within the training set. This approach although important sets it apart from learning through interaction which is the basis of RL. When it comes to unsupervised learning there is an urge to either classify reinforcement learning as a kind of unsupervised learning since the latter aims to find patterns hidden in unlabelled data and does not explicitly

rely on examples of correct behaviour but this does not address the fundamental problem in reinforcement learning of maximizing the reward signal.

Due to the nature of learning through interaction, one of the challenges in reinforcement learning is the trade-off between exploration and exploitation. The success of the reinforcement learning agent is dependent upon the variety of actions it takes to identify those that are effective in producing a reward but in order to discover such actions, new actions that were not performed before need to be selected. Thus, in order to succeed in obtaining a reward the agent has to exploit what it has learned but to improve the reward obtained the agent also has to explore new actions. In order to succeed at the task neither exploration or exploitation can be performed exclusively. A key feature of reinforcement learning is that it considers the whole problem of a target-centric agent with an uncertain environment which is in contrast with many approaches that consider sub-problems without addressing their value in solving the bigger problem. Reinforcement learning starts off with a complete, interactive, goal seeking agent with explicit goals that can sense their environment and furthermore choose actions to influence said environment. When reinforcement learning is used in conjunction with other approaches, important sub-problems ,which play an important role in the reinforcement learning agents, are isolated and studied even if all the details of the agent cannot always be filled.

As technology has improved, reinforcement learning has a had a growing substantive relationship with other disciplines and has distinguished itself from other approaches by placing an emphasis on learning from interaction with the environment with an agent without the need for a complete model of the environment.. Reinforcement learning uses a formal framework of Markov decision processes as a simple way of representing the essential feature of the artificial intelligence problem, which are, a sense of cause and effect, uncertainty and nondeterminism and having explicit goals.

1-1 Motivation

Freeway traffic control has become a critical issue in modern society due to the exponential increase in vehicular movement, urbanization, and the corresponding challenges in maintaining road safety, reducing congestion, and minimizing environmental impact. Effective traffic management on freeways is not just about regulating the flow of vehicles but also about ensuring sustainability, economic efficiency, and the overall quality of life for urban populations. In the aftermath of COVID, which caused significant changes to traffic patterns in both urban and freeway settings, it has also become necessary to incorporate any such potential situations to while planning traffic control strategies in the long run.

This section explores the motivations behind freeway traffic control, focusing on safety, efficiency, environmental sustainability, and technological advancements.

1-1-1 Enhancing Road Safety

One of the primary motivations for freeway traffic control is enhancing road safety. Freeways are often high-speed environments where even minor disturbances can lead to severe accidents. According to the World Health Organization (WHO), road traffic injuries are one

of the leading causes of death globally, particularly among young adults aged 15-29 years. Implementing traffic control measures such as speed limits, dynamic lane management, and automated enforcement can significantly reduce the risk of collisions and fatalities. Intelligent Traffic Systems (ITS) and adaptive signal controls are increasingly being used to predict and mitigate potential traffic incidents, thereby enhancing overall safety on freeways.

1-1-2 Reducing Traffic Congestion

Congestion on freeways results in significant economic losses, wasted time, and increased fuel consumption. An example of the above would be the Texas A&M Transportation Institute's Urban Mobility Report (2023)[18] which estimates that congestion costs the United States economy over $200 billion annually in lost productivity and excess fuel consumption. Traffic control measures, including ramp metering, congestion pricing, and real-time traffic information systems, help manage the flow of vehicles and reduce bottlenecks. By optimizing the use of available road capacity, these measures not only alleviate congestion but also improve travel time reliability for commuters [29].

1-1-3 Environmental Sustainability

The environmental impact of freeway traffic is another critical concern. Vehicle emissions contribute to air pollution and climate change, with transportation being responsible for a significant share of global greenhouse gas emissions. Freeway traffic control systems can play a vital role in reducing emissions by minimizing stop-and-go driving conditions, which are known to increase fuel consumption and emissions. Strategies such as promoting carpooling, encouraging the use of electric vehicles through dedicated lanes, and integrating traffic management with public transportation networks contribute to more sustainable freeway use [3].

1-1-4 Economic Efficiency and Productivity

Efficient freeway traffic control also contributes to economic productivity. By reducing delays and improving the predictability of travel times, businesses can better plan logistics, leading to more efficient supply chains. The implementation of smart traffic management systems, which use data analytics and machine learning to optimize traffic flow, can lead to substantial economic benefits. These systems help reduce the time vehicles spend idling, lower transportation costs, and improve the overall efficiency of freight movement on highways [27].

1-1-5 Technological Advancements and Innovation

The advent of advanced technologies such as the Internet of Things (IoT), big data analytics, and artificial intelligence (AI) has revolutionized freeway traffic control. These technologies enable the development of smart infrastructure that can adapt to real-time traffic conditions, predict congestion, and even communicate with autonomous vehicles to enhance traffic flow. For instance, connected vehicle technology allows vehicles to share information about road

conditions, traffic incidents, and optimal speeds, which can be integrated into traffic management systems to improve safety and efficiency [24]. The motivation to leverage these technologies stems from the need to create more resilient and adaptive transportation networks that can meet the demands of modern society.

Freeway traffic control is motivated by a combination of factors, including the need to improve safety, reduce congestion, promote environmental sustainability, and enhance economic efficiency. The rapid advancement of technology further drives the implementation of sophisticated traffic management systems, which are essential for addressing the challenges posed by modern urbanization and increasing vehicular traffic. As society continues to evolve, so too must our approaches to freeway traffic control, ensuring that our transportation systems remain safe, efficient, and sustainable for future generations.

1-2 Research

The focus of the literature report leading upto the thesis was to gain knowledge regarding the existing freeway traffic control methods and the existing reinforcement learning algorithms. Research on the former gave me a broad understanding of the methods used to mimic and generate traffic scenarios using mathematical models like METANET and more recently using software such as SUMO, VISSISM or GTSim. The method of freeway traffic control most commonly used from literature was ramp metering by either controlling the flow through the ramp or controlling the signal timings of the ramp signal with most common control algorithm being ALINEA. Then, variable speed limits were introduced to control the maximum speed on the highway which showed some good results. Eventually, a combination of both ramp metering and speed limits started being experimented with to improve the overall conditions of the highway and to tackle multiple issues at once such as delays due to weather, accidents, reduction of all while focussing on keeping the traffic flowing smoothly on the highway. As the complexity of the control methods started increasing, there was a need for better algorithms to match the need to control more variables. Thus, reinforcement learning algoritms were introduced into freeway traffic control in order to provide an environment where each and every variable required by the user could be defined and applied in order to get the best result possible. The advent of reinforcement learning in freeway traffic control meant that taking advantage of the structure of the environment, the user could include any and all metrics as observation and model rewards as required while defining the control input as actions. The main advantage of using reinforcement learning was the fact that one could mould the reward function with respect to what the user deemed to be the most important metrics for any particular scenario. From literature, several RL algorithms like simple Q-learning, DQN, DPG, A2C, PPO, DDPG and TD3 have been used across various papers to improve freeway traffic conditions. In general, the literature is divided when it comes to using discrete or continuous observations but they all use only discrete actions. The literature report aimed to find some unexplored areas of interest and to take the thesis in that direction. The main objectives and outline of the thesis is discussed in Sections 1-3 and 1-4 respectively.

1-3 Objectives

The objective of this thesis is to expand on the existing literature regarding freeway traffic control and apply known control methods with improvements and new reinforcement learning algorithm to further the work that has been done in this field. The main objectives of this thesis can be broadly summarized as follows:

1. Create a simple easy to follow structure for designing a freeway traffic environment
2. Application of continuous actions for both ramp metering and variable speed limits to improve fine control and reduce restrictions caused by discrete, specific values
3. Applying the soft actor critic(SAC) algorithm
4. Comparing the results between different control strategies and reinforcement learning algorithms

1-4 Thesis Outline

In this thesis, we discuss the freeway traffic control problem with increased complexities both inherently and with respect to the reinforcement learning agent. The two main features of the proposed method for control in this thesis are the use of continuous actions to control ramp metering and variable speed limits and the implementation of SAC reinforcement learning agent and compare the results to existing control methods and DDPG reinforcement learning agent. This thesis report presents the methodology and results for the aforementioned problem in detail. In chapter 2, a background of reinforcement learning and some important RL agents is provided along with a review of literature with respect to reinforcement learning in freeway traffic control. Chapter 2 also provides insight into some important hyperparameters and the interdependencies on the environment. In chapter 3, the proposed network, RL environment and RL agents setup are described in detail. The simulation settings along with the results and accompanying plots are provided in chapter 4. Chapter 5 highlights the results obtained and their importance to the field of traffic control. Lastly, chapter 6 gives an optimistic outline of future prospects and improvements that can be made to the methodology and results provided in this thesis.

Chapter 2

Background

2-1 Reinforcement Learning Algorithms

Reinforcement learning at its core is learning how to map situations to actions in order to maximize a numerical reward. This type of learning is showcased wherein the learner must discover which actions lead to best reward by trial and error and not through direct instructions. The choice of actions can not only determine the immediate reward but also consequently affect subsequent rewards which can vary from a case to case basis. The core distinguishing features of reinforcement learning can be said to be - trial and error approach and delayed rewards - as described above.[31]

Reinforcement learning is considered to be the third paradigm of machine learning and must not be grouped with supervised or unsupervised learning. The fundamental difference between supervised learning and reinforcement learning lies in the fact that the former uses a set of examples provided by an external supervisor and learns through the set. The objective of supervised learning is to extrapolate or generalize the structure learned from the examples to instances not present within the training set. This approach although important sets it apart from learning through interaction which is the basis of RL. When it comes to unsupervised learning there is an urge to either classify reinforcement learning as a kind of unsupervised learning since the latter aims to find patterns hidden in unlabelled data and does not explicitly rely on examples of correct behaviour. But this does not address the fundamental problem in reinforcement learning of maximizing the reward signal.

Due to the nature of learning through interaction, one of the challenges in reinforcement learning is the trade-off between exploration and exploitation. The success of the reinforcement learning agent is dependent upon the variety of actions it takes to identify those that are effective in producing a reward but in order to discover these, new actions that were not performed before need to be selected. Thus, in order to succeed in obtaining a reward the agent has to exploit what it has learned, and in order to improve the reward obtained the agent also has to explore new actions. In order to succeed at the task neither exploration nor exploitation can be performed exclusively [14]. A key feature of reinforcement learning is

that it considers the whole problem of a target-centric agent with an uncertain environment, which is in contrast with many approaches that consider sub-problems without addressing their value in solving the bigger problem. Reinforcement learning starts off with a complete, interactive and goal seeking agent with explicit goals that can sense their environment and furthermore choose actions to influence said environment. When reinforcement learning is used in conjunction with other approaches, important sub-problems are isolated and studied even if all the details of the agent cannot always be filled.

As technology has improved, reinforcement learning has had a growing substantive relationship with other disciplines and has distinguished itself from other approaches by placing an emphasis on learning from interaction with the environment without the need for a complete model of the environment. Reinforcement learning uses a formal framework of Markov decision processes as a simple way of representing the essential feature of the artificial intelligence problem.[31].

In this section, the reinforcement learning algorithms that will be applied to the freeway traffic control problem in the thesis are discussed to give the readers an introduction to reinforcement learning.

2-1-1 Reinforcement Learning and Markov Decision Processes

Markov decision process(MDP) is a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations or states, through future rewards. They represent a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made. This section will introduce the elements of the mathematical structure of the MDP problem such as returns, value functions, and Bellman equations which will aid in the understanding of the following algorithms and subsequent sections.

Agent-Environment Interface

Markov decision processes are meant to be a direct approach to the concept of learning with interaction. The learner and decision maker is called an *agent* and everything outside the agent that it interacts with is called the *environment*. The interaction between the agent and the environment happens continuously, wherein, the agent selects the *action* and the environment responds to said actions and presents new situations to the agent.

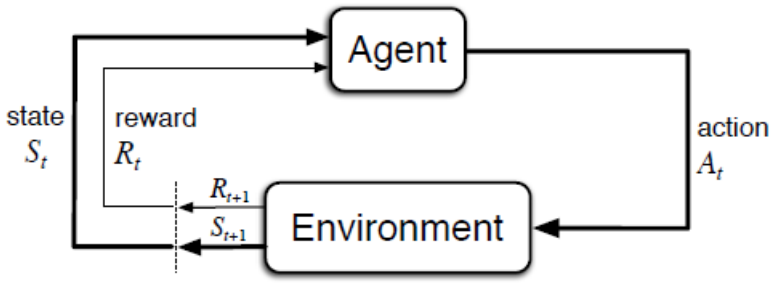


Figure 2-1: Agent-Environment Interaction in an MDP

The agent and the environment interact at a sequence of discrete time steps t, at which the agent receives a representation of the environments state $S_t \in \mathcal{S}$ on which basis it selects the action $A_t \in \mathcal{A}$. As a consequence of the performed action at time step t, the agent receives a reward $R_{t+1} \in \mathcal{R}$ in the subsequent time step $t+1$ and a new state S_{t+1}. The MDP and agent together thereby give rise to a sequence or trajectory that begins as follows, $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$.

In a finite MDP, the set of states, actions and rewards all have a finite number of elements. In this case, the random variables R_t and S_t have discrete probability distributions that are dependent only on the previous state and action. The function p, that defines the dynamics of the MDP is shown below:

$$p\left(s', r \mid s, a\right) \doteq \Pr\left\{S_t = s', R_t = r \,|S_{t-1} = s, |\, A_{t-1} = a\right\}, \tag{2-1}$$

for all s', $s \in \mathcal{S}$, $r \in \mathcal{R}$, $a \in \mathcal{A}(s)$.

The function in Eq.2-1 is a deterministic function of four arguments with conditional probability p, that specifies a probability distribution for each choice of s and a, that is,

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p\left(s', r \mid s, a\right) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s) \tag{2-2}$$

In an MDP, the probabilities given by p completely characterize the environment's dynamics, i.e, the probability of each possible value for S_t and R_t depends only on the immediately preceding state and action, S_{t-1} and A_{t-1}, and, given these, not on all earlier states and actions. This is observed as a restriction on the state not the decision process. The state must include information about all aspects of the past agent–environment interaction that make a difference for the future. If it does, then the state is said to have the Markov property. From the four-argument dynamics function, state transition probabilities p, can be computed, with slight abuse of notation,

$$p\left(s' \mid s, a\right) \doteq \Pr\left\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\right\} = \sum_{r \in \mathcal{R}} p\left(s', r \mid s, a\right) \tag{2-3}$$

Expected rewards for state-action pairs as a two argument function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, can be computed as,

$$r(s, a) \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a\right] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p\left(s', r \mid s, a\right), \tag{2-4}$$

and expected rewards for state–action–next-state triples as a three-argument function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ can also be computed as follows,

$$r\left(s, a, s'\right) \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right] = \sum_{r \in \mathcal{R}} r \frac{p\left(s', r \mid s, a\right)}{p\left(s' \mid s, a\right)} \tag{2-5}$$

The MDP framework is abstract and flexible and can be applied to a variety of problems. The flexibility of the MDP framework stems from the definitions of what can be used as variables

in the decision process. For example, time steps need not be actual real time intervals but can be based on stages for decision making and acting. The states of the agent can take a variety of forms, from low-level sensor readings to abstract high-level symbolic descriptions of objects. They can also be from completely subjective past memories, mental or computational. The actions taken by the agent can also be defined as low-level basic actions like, application of voltages, or high-level abstract actions such as, decisions taken in daily life. The general rule followed is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment and it is not assumed that everything in the environment is unknown to the agent.

The MDP framework is a considerable abstraction of the problem of goal-directed learning from interaction. The framework proposes that regardless of the details of the problem and the objective to be achieved, a reinforcement learning problem can be reduced to three signals communicated between an agent and its environment: one signal to define the agent's choices (the actions), one signal to define the basis for the choices (the states), and one signal to define the agent's goal (the rewards). This framework may not be sufficient to represent all decision-learning problems usefully, but has proved to be widely useful and applicable.

Policy and Value Functions

The reinforcement learning algorithms discussed in this section involve estimating value functions — functions of states (or of state–action pairs) that estimate goodness of the agent in a given state (or how good it is to perform a given action in a given state). The notion of performance here is defined in terms of future rewards that can be expected, or in terms of expected return. The rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular ways of acting, called policies.

Policies are a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Reinforcement learning methods specify how the agent's policy is changed as a result of its experience.

The value function of a state s under a policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, it can be formally defined as,

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S}, \tag{2-6}$$

where $\mathbb{E}_\pi[.]$ denotes the expected value of a random variable given that the agent follows policy π, and t is any time step. The value of the terminal state, if any, is always zero. The function v_π, is called the state-value function for policy π.

Similarly, the value of taking action a, in state s, under a policy π, denoted by $Q_\pi(s, a)$, can be defined as follows:

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \tag{2-7}$$

The function q_π, is called the action value function for policy π. The value functions defined in Eq.2-6 and Eq.2-7 are estimated from experience as will be seen in the following sections.

2-1-2 Deterministic Policy Gradient

In an environment which provides a continuous action space we can derive a deterministic policy by using Deterministic policy gradient (DPG). Rather than returning a probability distribution over actions $\mathcal{A}$ given a state, a deterministic policy $\mu(s) = a$ returns a single action in a deterministic way. The main objective $J(\theta)$ in an off-policy actor-critic algorithm, which mainly optimizes the value function is defined as:

$$J(\theta) = \int_{\mathcal{S}} d^{\mu}(s) Q\left(s, \mu_\theta(s)\right) ds, \tag{2-8}$$

where θ are the parameters and $\mathcal{S}$ is the state space.

$$d^{\mu}\left(s'\right) = \int_{\mathcal{S}} \sum_{k=1}^{\inf} \gamma^{k-1} d_0(s) d^{\mu}\left(s \rightarrow s', k\right) ds, \tag{2-9}$$

is defined as the discounted sum of state visitation probability density at state $s' \cdot d^{\mu}(s \rightarrow s', k)$ giving the probability density from state s to state s' after moving k steps by using policy μ. $d_0(s)$ is the initial distribution over states. The gradient of $J(\theta)$ using the Deterministic policy gradient theorem can now be computed as,

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_{\mathcal{S}} d^{\mu}(s) \nabla_a Q^{\mu}(s,a) \nabla_\theta \mu_\theta(s)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s \approx d^{\mu}}[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu}(s,a)|_{a=\mu_\theta(s)}] \end{aligned} \tag{2-10}$$

First, the chain-rule yields the gradient of $Q \nabla_a Q^{\mu}(s,a)$ with respect to a. Second, the gradient of the deterministic policy $\nabla_\theta \mu_\theta(s)$ with respect to theta, which optimizes our policy is derived. As an example to show how to compute updates, DPG in combination with on-policy actor-critic policy SARSA is considered. First, the TD-error in SARSA is computed,

$$\delta_t = R_t + \gamma Q_w\left(s_{t+1}, a_{t+1}\right) - Q_w\left(s_t, a_t\right) \tag{2-11}$$

The parameter update of the value function is defined as:

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q_w\left(s_t, a_t\right) \tag{2-12}$$

Then, the Deterministic policy gradient theorem is used to compute policy parameter updates of θ using Eq.2-10:

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_a Q^{\mu}(s,a) \nabla_\theta \mu_\theta(s)|_{a=\mu_\theta(s)} \tag{2-13}$$

One problem of using DPG is exploration because of the deterministic nature of the policy we optimize. One way to prevent this is to add noise to the parameter space or action space, which in this case would result in an off-policy nondeterministic policy.

2-1-3 Deep Deterministic Policy Gradient

Deep deterministic policy gradient is an off-policy, model free reinforcement learning algorithm that was introduced to solve problems in environments with continuous action spaces. It can be seen as a variation on deep Q-networks, as it combines deterministic policy gradient with deep Q-learning, experience replay, target values and policy network, which thus can be said to be an actor-critic algorithm. DDPG has 4 neural networks: actor, critic, target actor and target critic, wherein, the actor is a policy function $\pi(s)$ that computes an action for a given state. The critic, or the Q-value function, computes the Q-value as a numerical value which represents the discounted future reward for state-action pairs, and also is the main objective to be optimized by taking the maximum real value[2]. The optimal policy is derived from the minimization of loss between the output of approximation and the Bellman equation, shown below,

$$Q^*(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma max_{a_{t+1}} Q^{'*}(s_{t+1}, \pi_{\prime}(s_t)] \tag{2-14}$$

In the Eq.2-14, γ represents the discount rate that diminishes the additional reward of steps in the future. This shows that in this algorithm the immediate rewards take precedence over future rewards. From the above equation, given target function and neural networks as function approximators, we derive the loss function as,

$$L(\theta^Q) = \mathbb{E}[(Q(s_t, a_t|\theta^Q) - Y_t)^2], \tag{2-15}$$

such that, Y_t represents the target computed as follows,

$$Y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}|\theta^Q), \tag{2-16}$$

where θ^Q represents the function parameters for policy μ and value function Q.

Exploration Noise:

As observed in Section 2-1-2, the updates from DPG could inhibit exploration depending on the environment. DDPG takes this into account and employs the use of some noise ($\mathcal{N}$) added to the to the actions of the policy network μ to create a new exploration policy $\mu^{'}$, that guarantees exploration in the continuous action space. This new policy is simply shown as,

$$\mu^{'}(s_t) = \mu(s_t) + \mathcal{N} \tag{2-17}$$

Target Value and Policy Networks:

DDPG uses frozen copies of value and policy functions to compute the target due to the decreasing stability of the learning process which occurs because of the change in weights during optimization.

- Step 1: Batch of training data is sampled from the experience buffer

- Step 2: Loss function (L) is computed using target values and policy networks to get the target.

The target networks then get soft updated by the following scheme,

$$\begin{aligned} \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \end{aligned} \tag{2-18}$$

.

In the Eq.2-18, the terms on the LHS represent the parameters of the target value and policy networks and the terms on the RHS represent the parameters of the given value and policy networks. The variable τ is a hyperparameter that is much less than 1, which therefore scales down the step update in order to realize a soft update and slows down the change of parameters of the target compared to the actor-critic networks. The pseudocode for the DDPG algorithm is given in A-1.

Experience Replay Buffer:

The experience sample buffer used in DDPG consists of samples generated by the interaction of the policy and environment. These samples are then used in batches to perform updates using the value function, bellman equation and DPG. The experience replay is a fixed-size buffer that holds the most recent transitions collected by the policy. It greatly improves the sample efficiency of the algorithm by enabling data to be reused multiple times for training, instead of throwing away data immediately after collection, and also improves the stability during training. It is typically implemented as a circular buffer wherein an old transition makes room for a newer transition which are sample at fixed intervals for training. The sampling strategies can be used like uniform sampling, prioritized experience replay and distributed experience replay buffer.[7]

2-1-4 Twin Delayed Deep Deterministic Policy Gradient

As shown in Section 2-1-3, one of the drawbacks of the DDPG algorithm is the overestimation of Q-value which can result in breaking of policy. The TD3 algorithm uses three main improvements to reduce the effect of the overestimation bias mentioned above[2]:

- **Clipped Double Q-learning**

  The first improvement to address the overestimation bias in DDPG is done by introducing an additional critic or value function network and the target is optimized over the minimum of the two Q-values,

  $$Y_t = r\left(s_t, a_t\right) + \gamma \min_{i=1,2} Q_{\theta_i}\left(s_{t+1}, \mu_{\theta_i'}\left(s_{t+1}\right)\right) \tag{2-19}$$

  By choosing the minimum of the two values it is harder for the value function to overestimate the Q-value.

- **Delayed Policy Network Updates**

  By reducing the frequency of updates for the policy network compared to the value network, it prevents the convergence of the value function in failure where overestimation of actions often occurs. Overestimated output of a poor policy by the value function can be overcome using additional updates of the value network using the same policy.

- **Target Policy Smoothing**

  The third improvement is adding noise to the output of the target policy network. This is done by modifying an action produced by the target policy with added noise by clipping it into an interval. This covers a clipped area in the action space instead of predicting a deterministic action. Noise can act as a regularizer for incorrect Q-values produced by the value function for an action, thus getting smoothed.

All the above mentioned improvements provide added stability for approximations of optimal policy [8]. The pseudocode for the TD3 algorithm is given in A-2.

2-1-5 Soft Actor-Critic

The Soft Actor-Critic algorithm applies the maximum entropy framework to augment the standard reinforcement learning reward with an entropy maximization term. The use of maximum entropy within the confines of reinforcement earning is not a novel idea but in previous literature it is observed that the term is used as a regularizer rather than maximizing the entropy. SAC uses three main components: an actor-critic architecture with separate policy and value function networks, an off-policy formulation that enables reuse of previously collected data for efficiency, and entropy maximization to enable stability and exploration[11]. This method further combines off-policy actor-critic training with a stochastic actor to maximize the entropy of this actor with an entropy maximization objective. The SAC algorithm is a far more stable and scalable algorithm and exceeds in both efficiency and performance compared to DDPG and similar algorithms when it comes to complex tasks with high dimensional sample sets. The SAC algorithm compensates for, to a large extent, the drawbacks faced by model free reinforcement learning approaches namely, high sample complexity and brittle convergence properties or hyperparameter sensitivity [10]. SAC considers a more general maximum entropy objective in place of the standard reward function maximization, that favours stochastic policies with the addition of expected entropy over the policy to the objective,

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim d_\pi} \left[r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \alpha \mathcal{H}\left(\pi\left(\cdot \mid \mathbf{s}_t\right)\right) \right] \tag{2-20}$$

The temperature parameter α determines the relative importance of the entropy term against the reward, and thus controls the stochasticity of the optimal policy. All other notations used in the equation are common to all reinforcement learning algorithms.

The progression of the soft actor-critic approach starts from the maximum entropy variant of policy iteration - soft policy iteration - which is a general algorithm for learning optimal

entropy policies by alternating policy evaluation and policy improvement within the maximum entropy framework.

Soft-Policy Iteration

In the *policy evaluation* step of soft policy iteration, the value of a policy π is computed according to the maximum entropy objective in Equation 2-20. For a fixed policy, the soft Q-value can be computed iteratively, starting from any function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ and repeatedly applying a modified Bellman backup operator given by $\mathcal{T}^{\pi}$ as shown below:

$$\mathcal{T}^{\pi} Q\left(\mathrm{s}_t, \mathrm{a}_t\right) \triangleq r\left(\mathrm{s}_t, \mathrm{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V\left(\mathrm{s}_{t+1}\right)\right], \tag{2-21}$$

where

$$V\left(\mathrm{s}_t\right) = \mathbb{E}_{\mathbf{a}_t \sim \pi}\left[Q\left(\mathrm{s}_t, \mathbf{a}_t\right) - \log \pi\left(\mathbf{a}_t \mid \mathrm{s}_t\right)\right], \tag{2-22}$$

is the soft state value function. We can obtain the soft value function for any policy π by repeatedly applying $\mathcal{T}^{\pi}$ as formalized in Lemma 1.

In the *policy improvement* step, the policy towards the exponential of the new Q-function is updated. This particular choice of update can be guaranteed to result in an improved policy in terms of its soft value. Since in practice policies that are tractable are preferred, the policy is additionally restricted to some set of policies Π, which can correspond, for example, to a parameterized family of distributions such as Gaussians. To account for the constraint that $\pi \in \Pi$, the improved policy projected into the desired set of policies. For this projection, it can be shown that the new, projected policy has a higher value than the old policy with respect to the objective in Equation 2-20 which is formalized in Lemma 2.

The full *soft policy iteration* algorithm alternates between the soft policy evaluation and the soft policy improvement steps, and it will provably converge to the optimal maximum entropy policy among the policies in Π (Theorem 1). Although this algorithm will provably find the optimal solution, it can perform in its exact form only in the tabular case. Therefore, to approximate the algorithm for continuous domains, it needs to rely on a function approximator to represent the Q-values, and running the two steps until convergence would be computationally too expensive. The approximation gives rise to a new practical algorithm, called soft actor-critic.

Realizing Soft-Actor Critic

As discussed above, large continuous domains require the derivation of a practical approximation to soft policy iteration. To that end, function approximators for both the Q-function and the policy are used, and instead of running evaluation and improvement to convergence, alternate between optimizing both networks with stochastic gradient descent. A parameterized state value function $V_{\psi}(s_t)$, soft Q-function $Q_{\theta}(s_t|a_t$, and a tractable policy $\pi_{\phi}(a_t|s_t)$ are considered. The parameters of these networks are ψ, θ and ϕ.

The state value function approximates the soft value. This quantity can be estimated from a single action sample from the current policy without introducing a bias, although in practice, including a separate function approximator for the soft value can stabilize training and is

convenient to train simultaneously with the other networks. The soft value function is trained to minimize the squared residual error;

$$J_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi \left(s_t \right) - \mathbb{E} \left[Q_w \left(s_t, a_t \right) - \log \pi_\theta \left(a_t \mid s_t \right) \right] \right)^2 \right], \tag{2-23}$$

with gradient:

$$\nabla_\psi J_V(\psi) = \nabla_\psi V_\psi \left(s_t \right) \left(V_\psi \left(s_t \right) - Q_w \left(s_t, a_t \right) + \log \pi_\theta \left(a_t \mid s_t \right) \right). \tag{2-24}$$

The soft Q-function parameters can be trained to minimize the soft Bellman residual,

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta \left(\mathbf{s}_t, \mathbf{a}_t \right) - \hat{Q} \left(\mathbf{s}_t, \mathbf{a}_t \right) \right)^2 \right], \tag{2-25}$$

with gradient,

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta \left(\mathbf{a}_t, \mathbf{s}_t \right) \left(Q_\theta \left(\mathbf{s}_t, \mathbf{a}_t \right) - r \left(\mathbf{s}_t, \mathbf{a}_t \right) - \gamma V_{\bar{\psi}} \left(\mathbf{s}_{t+1} \right) \right), \tag{2-26}$$

Finally, the policy parameters can be learned by directly minimizing the expected KL-divergence given by:

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathrm{D}_{\mathrm{KL}} \left(\pi_\phi \left(\cdot \mid \mathrm{s}_t \right) \| \frac{\exp \left(Q_\theta \left(\mathbf{s}_t, \cdot \right) \right)}{Z_\theta \left(\mathbf{s}_t \right)} \right) \right] \tag{2-27}$$

The above equations form the basis of the SAC algorithm. The pseudocode for the SAC algorithm is given in A-3. For further reference, readers are referred to [10, 11] which provide a comprehensive study on the SAC algorithm, simulations and applications.

2-2 Effects of Hyperparameters on DDPG and SAC

Deep Deterministic Policy Gradient (DDPG) and Soft Actor-Critic (SAC) are two widely used reinforcement learning (RL) algorithms, particularly in continuous action spaces. Both algorithms rely heavily on the tuning of hyperparameters, which can significantly impact their performance. Hyperparameters such as learning rates, discount factors, and the structure of neural networks influence the stability, convergence speed, and overall success of the training process.

This section examines the effects of key hyperparameters on the training of DDPG and SAC. This section serves to provide information on the changes that might be observed by users due to hyperparamters and how to interpret and correct unwanted behaviour during the training process. Furthermore, the effect of hyperparameters based on the environment is also outlined to highlight the intricacies of working with reinforcement learning algorithms.

2-2-1 Brief Overview of DDPG and SAC

DDPG (Deep Deterministic Policy Gradient)

DDPG is an off-policy, model-free RL algorithm that combines the strengths of DQN (Deep Q-Network) and the Actor-Critic framework [21]. It is particularly suited for continuous action spaces and operates using two neural networks: an actor network that learns the policy and a critic network that evaluates the action-value function.

SAC (Soft Actor-Critic)

SAC is a more recent off-policy, model-free RL algorithm designed to improve stability and efficiency in training by introducing entropy regularization into the policy update [10]. The objective of SAC is to maximize both the expected return and the entropy of the policy, promoting exploration and preventing premature convergence to suboptimal policies.

2-2-2 Key Hyperparameters and Their Effects on DDPG Training

The parameters listed in this section, in general, affect the learning process for DDPG more significantly than other hyperparameters.

Learning Rate

The learning rate controls the step size during the update of network weights. In DDPG, choosing the appropriate learning rate is critical. If the learning rate is too high, the updates can be unstable, leading to divergent behaviour where the policy fails to converge. Conversely, if the learning rate is too low, training can become excessively slow, potentially resulting in poor performance as the agent may not explore the action space effectively.

Discount Factor (γ)

The discount factor determines the importance of future rewards. A lower discount factor makes the agent more short-sighted, focusing on immediate rewards, which can be beneficial in environments where long-term rewards are uncertain or unreliable. A higher discount factor, on the other hand, encourages the agent to consider long-term rewards, which can be crucial in environments where planning is necessary. However, setting the discount factor too high may slow down the learning process or lead to suboptimal policies in environments with delayed rewards.

Batch Size

Batch size refers to the number of samples used in each update of the neural networks. In DDPG, a larger batch size generally leads to more stable gradient estimates, which can improve convergence. However, larger batch sizes also require more computational resources and can slow down the learning per iteration. Smaller batch sizes, while faster per iteration, might introduce more noise in the updates, potentially destabilizing the learning process.

Exploration Noise

DDPG uses an exploration strategy based on adding noise to the deterministic policy (often Gaussian or Ornstein-Uhlenbeck noise). The magnitude of this noise significantly affects exploration. Too much noise can cause erratic behaviour, making it difficult for the agent to learn a coherent policy. Too little noise can lead to insufficient exploration, where the agent might prematurely converge to sub-optimal policies.

Target Network Update Rate (τ)

DDPG employs a target network that is slowly updated towards the current network, controlled by the parameter τ. A smaller τ results in slower updates, which can stabilize training by smoothing the changes in the policy and value estimates. However, if τ is too small, learning can be very slow. Conversely, a larger τ can speed up learning but may destabilize training by introducing excessive variance into the updates.

2-2-3 Key Hyperparameters and Their Effects on SAC Training

Learning Rate

Similar to DDPG, the learning rate in SAC must be carefully tuned. SAC generally exhibits better stability than DDPG due to its entropy regularization, but an inappropriate learning rate can still lead to instability. SAC typically requires a balance between the learning rates of the actor and critic networks. Discrepancies in these learning rates can lead to one network lagging behind the other, which can degrade performance.

Entropy Coefficient (α)

One of the most distinctive features of SAC is its use of an entropy coefficient (α), which controls the trade-off between exploration and exploitation. A higher α encourages more exploration by favouring higher-entropy policies. However, setting α too high can lead to excessive exploration, making it difficult for the agent to exploit known good policies. Conversely, a lower α reduces exploration, which might cause the agent to converge prematurely to a suboptimal policy. Most SAC implementations favour an automatic entropy co-efficient policy but tend to settle on values below 0.5 thus favouring exploitation over exploration. It is important to know the characteristics of the user environment before making a change to the α value as it is an important factor in the learning process for SAC.

Target Network Update Rate (τ)

In SAC, the target networks (both for the value function and for the policy) are updated using a similar technique as in DDPG. The target update rate τ affects how quickly these networks follow the main networks. As with DDPG, smaller values of τ tend to stabilize training by providing smoother updates, whereas larger values can lead to faster but potentially more unstable learning.

Discount Factor (γ)

The discount factor in SAC operates similarly to DDPG, influencing the agent's consideration of future rewards. Given SAC's entropy maximization, the discount factor also interacts with the entropy term, where a high γ with high entropy might overly diffuse the policy, while a low γ might limit the ability to learn long-term strategies effectively. As mentioned in Section 2-2-3, it is important to consider the interdependency of the entropy term while setting values for the related hyperparameters.

Replay Buffer Size

Both DDPG and SAC utilize a replay buffer to store and sample experiences during training. The size of the replay buffer in SAC can affect the diversity of the training samples. A larger buffer size allows for greater diversity in experiences, which can improve learning stability. However, if the buffer is too large, older, less relevant experiences might persist, potentially slowing down learning. A smaller buffer might lead to over-fitting to recent experiences, reducing the robustness of the learned policy.

2-3 Impact of Hyperparameters Based on RL Environment

The effect of hyperparameters on the performance of DDPG and SAC is not only dependent on the agent itself but also specific RL environment. Different environments present unique challenges, such as varying degrees of reward sparsity, dynamic complexity, and state-space dimensionality. The following points highlight how these factors may influence the importance of certain hyperparameters:

2-3-1 Environment Dynamics

In highly dynamic environments where state transitions are volatile, such as in robotics or autonomous driving simulations, careful tuning of the learning rate and target network update rate becomes crucial. For example, in a fast-paced environment, a smaller τ can prevent the agent from making overly aggressive updates that could destabilize learning.

2-3-2 Reward Structure

Environments with sparse rewards, where rewards are only received after completing a task, often require a higher discount factor and a larger replay buffer to ensure the agent can propagate the value of achieving long-term goals. In such cases, the entropy coefficient in SAC should also be tuned carefully to ensure sufficient exploration.

2-3-3 State and Action Space Complexity

Environments with high-dimensional state or action spaces require larger neural networks to adequately represent the policy and value functions, which in turn necessitates careful tuning

of the learning rate and batch size. In DDPG, adding appropriate noise to actions becomes even more critical in such environments to ensure adequate exploration of the action space.

2-3-4 Stochastic vs. Deterministic Environments

In stochastic environments, where the same action can lead to different outcomes, SAC's entropy regularization is particularly beneficial. The entropy coefficient should be adjusted to encourage enough exploration to handle the uncertainty. In deterministic environments, DDPG's deterministic policy approach often requires less aggressive exploration noise.

Hyperparameter tuning is vital for the successful application of DDPG and SAC in reinforcement learning tasks. The effects of these hyperparameters are complex and environment-dependent, making it essential to tailor them to the specific challenges presented by each RL environment. The interplay between learning rate, discount factor, batch size, exploration strategies, and entropy regularization must be carefully managed to achieve optimal performance. As RL continues to evolve, understanding and improving hyperparameter optimization processes will be crucial for advancing the effectiveness of these algorithms.

2-4 Reinforcement Learning in Freeway Traffic

Traffic models are the basis for the design of traffic controllers, both to be inserted in control algorithms in case of model-based controllers and to be used for testing and simulation purposes. As seen in literature, the three basic traffic models were discussed and in this section will discuss in detail the two most relevant traffic models for freeway traffic control.

Continuous Macroscopic Traffic Models

Macroscopic traffic models of continuous type consider flow, speed and density as aggregate variables and represent space and time in continuous domains by means of partial differential equations (PDEs).

Referring to a generic location x and time t, the aggregate variables used in continuous macroscopic traffic models are denoted as follows:

- $\varrho(x,t)$ is the traffic density [veh/km];
- $v(x,t)$ is the mean speed [km/h];
- $q(x,t)$ is the traffic flow [veh/h]

The following two equations represent the hydrodynamic and continuity equations for the continuous flow model:

$$q(x,t) = \varrho(x,t)v(x,t) \tag{2-28}$$

$$\frac{\partial \varrho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \tag{2-29}$$

The Fundamental Diagram denoted as $Q(\varrho(x,t))$ correlates the density and flow in steady state conditions and must satisfy the following conditions:

$$Q(0) = 0, \quad Q\left(\varrho^{\max}\right) = 0, \quad \left.\frac{dQ(\varrho)}{d\varrho}\right|_{\varrho=e^{\mathrm{cr}}} = 0 \tag{2-30}$$

Eq.2-30 shows that the flow must be equal to zero when there is no density or when the density is equal to its maximum, called jam density, and that the flow reaches its maximum value, also known as capacity, when the density is equal to the critical density.

Analogously, the steady-state relation between mean traffic speed and density is denoted with $V(\varrho(x,t))$ and its shape must be such that:

$$V(0) = v^{\mathrm{f}}, \quad V\left(\varrho^{\max}\right) = 0, \quad \frac{dV(\varrho)}{d\varrho} \leq 0, \tag{2-31}$$

where v^{f} indicates the free-flow speed, i.e. the speed of vehicles in absence of traffic. Moreover, the following relations must hold:

$$Q(\varrho(x,t)) = \varrho(x,t)V(\varrho(x,t)) \tag{2-32}$$

The above equations form the basis of the continuous macroscopic traffic model and are augmented with additional equations and conditions as required by the traffic control problem. The discretized form of these equations also form the basis for discrete macroscopic modelling that is discussed in the following section. The continuous model is not used that often in RL applications and thus the discussion will be limited to the above information that is relevant to provide some background for the following sections. For further reading, readers may refer to [28] which gives detailed mathematical formulation.

Discrete Macroscopic Traffic Models

Discrete macroscopic traffic models consider both space and time as discrete variables and are obtained by discretizing continuous macroscopic traffic models.

In discrete macroscopic traffic models, space(freeway link) is divided into N segments of length L (km) and time is discretized into K intervals of duration T. The index used to indicate segment length is $i = 1, \ldots, N$ and for the time duration is $k = 1, \ldots, K$. It should be noted that in some cases the discretization of freeway links might not be equal. When modelling for multiple links, each link maybe indexed using m, for example, $L_{m,i}$ which represents the link number and the segment of that particular link. The above convention will be used in all the following equations in order to encapsulate single and multi-link modelling.

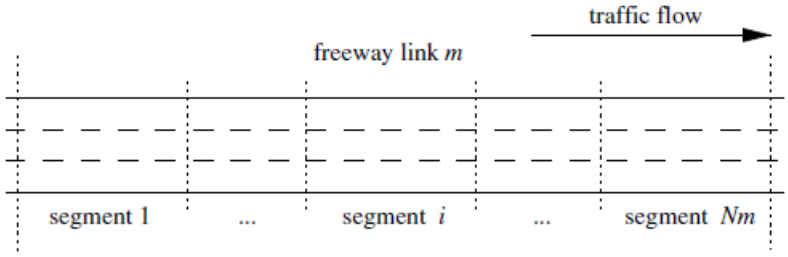


Figure 2-2: Freeway Link

Referring to a generic portion i of a freeway link m at time step k, the discrete macroscopic variables are as follows:

- $\rho_{m,i}(k)$ is the traffic density at time kT;
- $v_{m,1}(k)$ is the mean speed at time kT;
- $q_{m,i}(k)$ is the outflow or traffic volume during interval $[kT, (k+1)T]$.

The hydrodynamic and continuity equations, in the discrete case, that describe the evolution of the network over time are as follows:

The outflow of each segment is equal to the product of the density, mean speed and (if applicable) the number of lanes (λ_m):

$$q_{m,i}(k) = \rho_{m,i}(k) v_{m,i}(k) \lambda_m \tag{2-33}$$

The density of the segment is equal to the sum of the previous density and the difference between the inflow of the upstream segment and the outflow of the segment itself:

$$\rho_{m,i}(k+1) = \rho_{m,i}(k) + \frac{T}{L_m \lambda_m} \left(q_{m,i-1}(k) - q_{m,i}(k) \right) \tag{2-34}$$

Equations 2-33 and 2-34 are based on physical principles (hydrodynamics and continuity) and exact, the equations describing the speed dynamics and density-desired speed relationship are heuristic. As seen in the following Eq.2-35 the mean speed at time instant $k+1$ is dependent on the mean speed at k, a relaxation term that describes drivers trying to achieve the desired speed, a convection term that describes change in speed due to inflow of vehicles and an anticipation term that describes the change in speed due to changes in density,

$$\begin{aligned} v_{m,i}(k+1) = v_{m,i}(k) + \frac{T}{\tau} \left(V\left(\rho_{m,i}(k)\right) - v_{m,i}(k) \right) + \frac{T}{L_m} v_{m,i}(k) \left(v_{m,i-1}(k) - v_{m,i}(k) \right) \\ - \frac{\vartheta T}{\tau L_m} \frac{\rho_{m,i+1}(k) - \rho_{m,i}(k)}{\rho_{m,i}(k) + \kappa}, \end{aligned} \tag{2-35}$$

where τ,ϑ and κ are model parameters with,

$$V\left(\rho_{m,i}(k)\right) = v_{\text{free},m} \cdot \exp\left[-\frac{1}{a_m}\left(\frac{\rho_{m,i}(k)}{\rho_{\text{crit},m}}\right)^{a_m}\right], \tag{2-36}$$

with a_m a model parameter, and where the free-flow speed $v_{\text{free},m}$ is the average speed that drivers assume if traffic is freely flowing, and the critical density $\rho_{\text{crit},m}$ is the density at which the traffic flow is maximal.

A simple queue model is used model the origins:

$$w_o(k+1) = w_o(k) + T\left(d_o(k) - q_o(k)\right), \tag{2-37}$$

where $w_o(k+1)$ and $w_o(k)$ are the queue lengths at $k+1$ and k, $d_o(k)$ is the demand considered to be independent of any control actions, and $q_o(k)$ is the outflow.

The outflow $q_o(k)$ of the origin depends on the traffic conditions on the mainstream, and on the ramp metering rate $r_o(k)$, for the metered on ramp. The following Eq.2-38 shows that the outflow is a minimum of three compared quantities: the available traffic in time period k, the maximal flow that could enter the freeway because of the main-stream conditions, and the maximal flow allowed by the metering rate.

$$q_o(k) = \min\left[d_o(k) + \frac{w_o(k)}{T}, Q_o \cdot r_o(k), Q_o\left(\frac{\rho_{\max,m} - \rho_{m,1}(k)}{\rho_{\max,m} - \rho_{\text{crit},m}}\right)\right], \tag{2-38}$$

where Q_o is the on-ramp capacity under free-flow conditions and $\rho_{\max,m}$ is the maximum density of link m.

The equations mentioned above form the basis of a discrete second order macroscopic METANET model which is most commonly used to model freeway traffic [15, 17]. In subsequent papers, many augmentations and extensions to the METANET model have been made to further improve traffic modelling [13] and to adjust to specific control strategies and traffic scenarios. Additionally, the above first order discrete model is extended to a second order model which considers the influence of on-ramp and off-ramp flows on mainstream behaviour which is then extended to freeway networks [16]. For further reference, readers are directed to the papers [15, 17, 13, 12, 28]. In literature [28], the CTM is also described in detail as a first order discrete model for a one way road without any entrances or exits and then extended to traffic networks in considering triangular or trapezoidal shapes for steady state conditions. The aim of introducing these models in brief is to give a clearer picture regarding the literature shown in table 2-4-2.

2-4-1 Freeway Traffic Model as Markov Decision Process

The freeway traffic problem will be defined as an MDP in this section.

The states of the freeway traffic model can be defined by any of the given variables:

- S = density, mean speed, queue length, demand,

depending on the control strategy and requirement.

The action space of the freeway traffic model is created based on the control strategy:

- A = ramp metering rate, speed limits, signal phases,

The rewards used are also dependent on the control strategy adopted as well as any additional characteristics that the agent should optimize:

- R = Outflow, queue length, TTT/ TTS(total travel time/ total time spent), deviation from critical density, total waiting time.

The above state and action sets along with appropriate rewards from the given options both singular or combinations can be applied to the freeway traffic models mentions above in order to create the required traffic conditions and scenarios for any particular control tasks in both continuous and discrete time. It should be noted that TTT and TTS are often used interchangeably in different literature.

2-4-2 Review of Related Literature

In this section, a comprehensive overview of papers related to reinforcement learning techniques applied to freeway traffic control is carried out. The papers are listed in chronological order in the following to provide some insight into the progress of techniques within the subject.

Table 2-1: Summary of Reinforcement Learning papers for Freeway Traffic Control

Reference	RL Algorithm	Control Strategy	State	Action	Reward	Simulation Model and Result
Davarynejad *et al.*[4]	Q-learning	RM	Discrete; density,queue length,demand and metering rate	Discrete; ramp metering rate	Outflow and queue length	METANET; Flow capacity is maintained
Zhu *et al.*[38]	R-MART	VSL	Discrete; densities	Discrete; speed limits	Total Travel Time	DNL; Reduce TTT by 18%

Continued on next page

Table 2-1: Summary of Reinforcement Learning papers for Freeway Traffic Control (Continued)

Fares *et al.*[5]	Q-learning	RM	Discrete; density and signal at one step before	Discrete; red and green phase	Deviation from critical density	FTFM(proposed); Maintains optimal density and light phases
Fares *et al.*[6]	Multi-agent Q-learning (independent, coordinated, centralized)	RM	Discrete; density	Discrete; red and green phase	Deviation from critical density	VISSIM; Reduce TTS by 6.5% and improve avg speed by 7%
Schmidt-Dumont *et al.*[26]	Multi agent Q-learning with function approximator	RM-VSL	Continuous; down/up-stream densities and queue length	Discrete; red phase durations(RM) and speed limits(VSL)	Deviation from desired density	METANET macroscopic model Reduction in TTS
Walraven *et al.*[32]	Q-learning	VSL	Continuous; densities and speed, previous and current speed limits	Discrete; speed limits	Total time spent	METANET; Reduce TTS

Continued on next page

Table 2-1: Summary of Reinforcement Learning papers for Freeway Traffic Control (Continued)

Lu *et al.*[23]	Q-learning	RM	Discrete; queue length and flow	Discrete; Ramp metering rate	Total Time Spent, deviation in Total Waiting Time	ACTM; Reduce TTS by 18.5%
Li *et al.*[20]	Q-learning	VSL	Discrete; densities of mainline and ramp	Discrete; speed limits	Deviation from critical density	Modified CTM; Reduce TTS by 49.34%
Wang *et al.*[34]	Distributed Q-learning	VSL	Continuous; densities and speed	Discrete; speed limit	Time to collision	MOTUS; Reduce TTT by 51%
Zhou *et al.*[37]	Q-learning with value approximation	RM	Continuous; densities and demand	Discrete; ramp metering rates	Deviation from density	CTM; desired density value maintained
Greguric *et al.*[9]	Deep Q-learning	VSL	Continuous; densities and previous limits	Discrete; speed limits	Deviation from critical density, difference in limits, speed oscillations	VISSIM; Improved average speed by 13% and reduced density by 12%

Continued on next page

Table 2-1: Summary of Reinforcement Learning papers for Freeway Traffic Control (Continued)

Wu *et al.*[36]	DDPG (Actor-Critic)	VSL (differential)	Discrete; occupancy rates	Discrete; speed limits	TTT, average velocity, emergency deceleration and emission	SUMO; improved congestion alleviation, accident and emission reductions
Li *et al.*[19]	DDQN	VSL	Continuous; demand and density Discrete; one step before speed limit	Discrete; speed limit	Deviation in density	CTM(macro); TTS reduced by 65%
Wang *et al.*[33]	Deep Actor-Critic (DDPG and TD3)	RM-VSL	Continuous; flow, density and speed	Discrete; speed limit, outflow and phase	Avg. speed, queue length and delay	SUMO-TCI; TTT reduced by 35%

In table 2-4-2, all papers describe the freeway systems as Markov Decision Processes, on which the RL algorithms are implemented. Thus the main RL components in each paper are described as such to conform to this nomenclature, namely, as states, actions and rewards, so on.

From the table, it is observed that the choices of state space are the states of freeway systems (i.e., density, speed, queue length, and demands), while the choice for action space is the control action that corresponds to the control strategy used in the referenced literature. The definition of the reward function is done in conjunction with the control objective adopted which mainly revolves around minimizing the total time spent (TTS) by the vehicles or regulating the densities to maximize the outflow. Both macroscopic and microscopic(rarely) simulators are considered to test the proposed algorithms. The results mentioned in the table take into account the best improvements over the benchmark cases presented in each paper.

As for the reinforcement learning algorithms used in literature, from table 2-4-2 we see that

Q-learning is the most used algorithm due to its relative simplicity and success in many applications. The drawback of Q-learning is its finite dimension space which makes it perform poorly when dimensions of state and action spaces increase. Due to this in most studies only a small section of the freeway is considered and it also makes integration and coordination of RM and VSL more challenging. Some papers try to mitigate this drawback by using multi-agent Q-learning and function approximators for Q-values to handle continuous state spaces. But this inturn increases the computational complexity of the algorithm and makes it more challenging to implement.

The last three entries in table 2-4-2, showcase the usage of DRL techniques in freeway traffic which can potentially deal with large state and action spaces and coordinate RM-VSL for large scale freeway network. DRL techniques are divided into two categories, namely, value-based and policy-based. Value-based methods inherit the idea of Q-learning to approximate the value functions for state-action pairs, one most well-known method of which is Deep Q-Network. The main limitation of DQN [9, 19] is that it can only deal with discrete action space. Policy-based methods search for an optimal policy directly that maximizes the expected accumulative long-term reward. The actor-critic algorithms exploit the strengths of both value-based and policy-based methods, where an actor determines how the agent behave and a critic evaluates the chosen action.

As far as the application of reinforcement learning to integrate RM-VSL in freeway traffic control goes presently the only literature that explicitly does this is presented in [33] which uses both DDPG and TD3 algorithms. It is also shown that a centralized DRL agent can handle a large freeway network with multiple and VSL-RM hybrid controllers. A comprehensive review of related literature is also done in [33] which outlines the different DRL algorithms(TRPO, A2C, 3DQN) used in earlier literature as reference.

The aim of listing both urban and freeway traffic control is to show the difference in approaches to solving both problems and some similarities in the methodologies as well. Both problems are described as Markov decision processes but the formulation differs. For urban traffic, signal intersections are considered; whereas, for freeway traffic, sections of freeways with ramps and variable message boards are considered. This creates an interesting overlap between urban and freeway traffic which is the dependence on signal phases; at intersections for urban traffic and at ramps for freeway traffic. Although the aim of both traffic control schemes is to improve traffic flow, the methodology adopted differs due to the different conditions under which traffic operates in urban and freeway settings. This also brings into the fold the topic of interdependence of the two traffic control methods. The inclusion of both urban and freeway traffic control is to highlight these nuances.

From the literature listed in the above table it is observed that none of the papers take into account applying RL algorithms to deal with multiple traffic scenarios. In terms of algorithms, it is seen that none of the literature applies SAC for both urban and freeway traffic control although it has been shown to improve performance over Deep Q-learning, DDPG and TD3. One of the drawbacks of the studies listed above is that the studies did not train the RL agent in with continuous actions.

Chapter 3

Combined RM-VSL using Reinforcement Learning

3-1 Outline of Problem Statement

In line with objectives listed in Section 1-3, the issue being tackled in this thesis is the application of continuous actions to a simple freeway network created in Eclipse SUMO software environment in order to control ramp metering and variable speed limits to improve the overall state of the traffic network. Additionally, to aid in this improvement the Soft Actor Critic algorithm will be applied for learning using Python which is an ideal choice due to its stochastic nature which inherently promotes randomness in exploration, along with its entropy regularization function which balances the trade-off between exploration and exploitation.

3-2 Proposed Methodology

The traffic network used for the simulations in the thesis was created in SUMO (Simulation for Urban Mobility). The network created is a 2.5 km stretch of straight highway with a single on-ramp. The aim was to keep the base network as simple as possible in order to accommodate any adjustments needed down the line as the simulations were performed. The reason for using SUMO was the simplicity in creating different types of networks and the availability of better reference materials. The network created was influenced by the network described in [33].

3-2-1 Network Description

The network is divided into 5 segments of 500 metres each with each segment consisting of two lanes. The on ramp consists of 2 segments with one lane each of length 490 meters and 10 metres respectively. The traffic signal for the on-ramp is placed at the end of the first

segment of the ramp, with the smaller segment acting as a buffer. Each lane on each segment has lane area detectors for gathering lane information. Induction loop detectors are placed at the origin, destination and upstream and downstream of the junction to measure inflow and outflow for the network. The network also includes multi-entry/exit detectors at the origins of the mainstream and the ramp and the destinations to monitor the number of vehicles along the network.

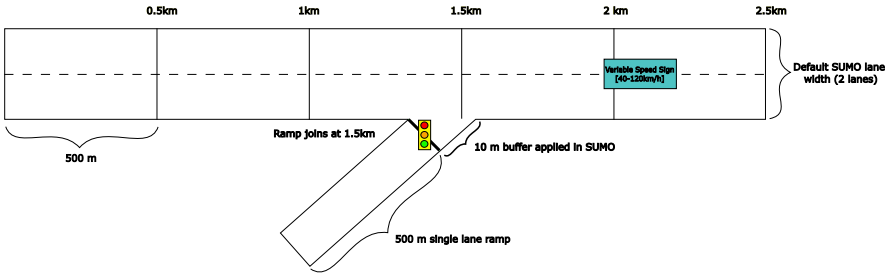


Figure 3-1: Full Network

The cycle time for the traffic signal is set to 60 seconds per cycle by default, which means that one change of the signal phase between red and green happens every 60 seconds. The cycle time can be changed according to the requirements of the simulation. The traffic light cycle time is of importance to set the initial traffic light logic within the network and to test non-RL based control methods for efficacy.

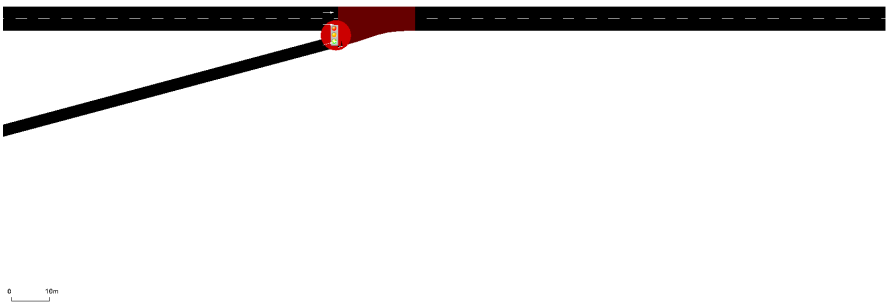


Figure 3-2: Junction

The network also includes a variable speed limit sign on the segment immediately upstream from the on-ramp. A fixed variable speed limit is placed on the last segment of the mainstream which varies between 40 km/h and free-flow speed for a specific duration to mimic congestion.

3-2-2 Demand Profile

The demand profile is an extremely important part of the traffic network creation process. This is due to the fact that the demand along the highway directly determines the effectiveness

of learning for any reinforcement learning agent that is applied to control the traffic. If the demand is too lenient, there would be no reason to apply any control strategies especially complex ones like reinforcement learning as there may be nothing to learn. Conversely, if the demand is too harsh, regardless of the applied control strategy there may be no good solution to the problem.

Two demand profiles have been used in this report to showcase some of the obstacles encountered while demand modelling when trying to apply different control algorithms. One of the demand profiles for the network used in the thesis was derived from the values described in [33] and [30] while the other has a reduction in ramp demand compared to the former. The demand profiles used for the simulations over 5000 seconds is:

Duration	0-500 s	500-1500 s	3500-5000 s
Mainstream Demand 1	4500 veh/hr	4500 veh/hr	1000 veh/hr
Ramp Demand 1	500 veh/hr	2500 veh/hr	500 veh/hr
Mainstream Demand 2	4500 veh/hr	4500 veh/hr	1000 veh/hr
Ramp Demand 2	500 veh/hr	2000 veh/hr	500 veh/hr

Table 3-1: Demand Profile

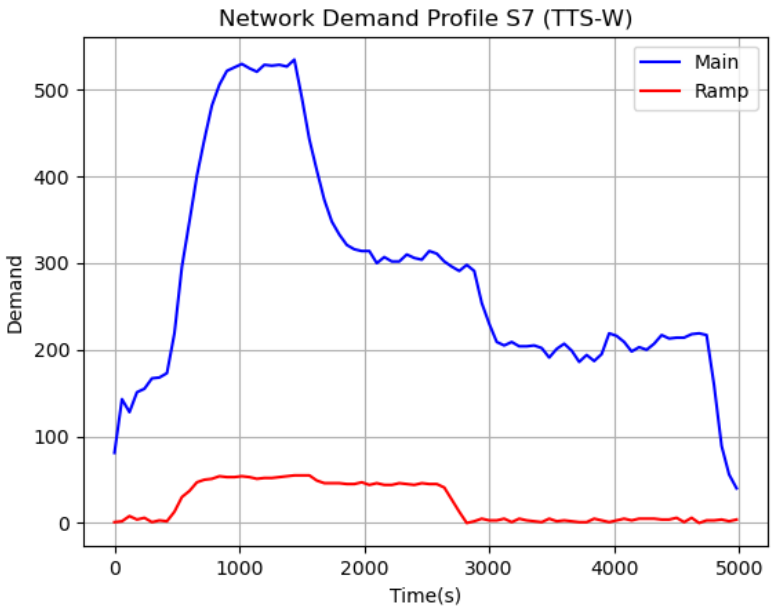


Figure 3-3: Demand Profile in SUMO

The demand profile for the mainstream is modelled in such a way that the demand remains

constant until 3500 seconds and drops off slowly until 500 seconds. For the ramp, the demand stays low until 500 seconds and then increases to a higher demand to cause some queues on the ramp and congestion on the highway and then drop down low again after 1500 seconds. This increase in on-ramp demand coincides with the decrease in speed limit sign on the last segment in the network as mentioned in 3-2-1 to create congestion. It is also important to mention that in this case the only vehicles included in this demand profile are regular passenger vehicles,

3-3 Environment Setup

The environment setup for the traffic network was done by interfacing SUMO with Python using TraCI. The basic setup of the reinforcement learning environment for the traffic network was created with reference to the environment created in [1].The SUMO-RL[1] framework provided a basis for implementing the proper interaction between SUMO and Python while the environment characteristics were coded as per the requirements of the network and the thesis.

3-3-1 Observation Space

The traffic environment created has an observation space with 22 states. They represent the densities of each segment on the mainstream and ramp excluding the buffer segment (6 states), the average speed of vehicles on all segments of the mainstream (5 states), the waiting times/delay on each segment of the mainstream (5 states), the queue length on the ramp, the waiting time on the ramp, total delay/waiting time on the network, the number of vehicles in the queue and the inflow and outflow at the upstream and downstream of the ramp respectively. The most important part of defining the observation space is to include as much useful information as possible for the reinforcement learning agent. The definition of useful information will vary based on the environment and with is the size of the observation space.

3-3-2 Action Space

The action space for the environment in this case is a continuous action space with values ranging from $[0, 1]$. The size of the action space varies depending on the control method. The size of the action space is just 1 action for only ramp metering, where the action value represents the ramp metering rate for controlling the timings of each phase. To apply both ramp metering and adjust the speed limits on one segment the action space size is 2 different actions for ramp metering and variable speed limits with one action corresponding to ramp metering rate and the speed limit adjustment on the two segments each.

3-3-3 Reward

The most important part of creating an environment for reinforcement learning is defining the reward. The modelling of reward is not only important from the perspective of the

environment but also for the reinforcement learning agents used in training. In this instance, for both DDPG and SAC a common reward has been modelled to better understand and compare performance of each agent.

The reward is modelled as a hybrid weighted sum of three different values, namely, the total time spent on the network, the number of vehicles in the queue on ramp in metres and the waiting time of the vehicles on the mainstream or delay. The values are calculated at each step in the environment and assigned weights given to each value based on their importance. The hybrid reward and weights assigned to each value are taken not only from literature [33], but also from multiple simulations done on this particular environment.

For this environment, the combination of TTS, number of vehicles in queue and delay on mainstream gave the best results out of all the tested rewards. The reason for that is that the TTS gives all-round information regarding the traffic network but might ignore certain behaviour. To help correct such behaviour, the delay on mainstream is included such that the agent has information that while reducing the TTS, the focus must also be placed on reducing the congestion on mainstream thereby, reducing delays. Then, the number of vehicles in queue was included in order to penalise poor agent behaviour on the ramp causing excessive queues and jams. From the simulations, the highest weight was assigned to the delay and equal weights were assigned to the TTS and number of vehicles in queue which gave the best results along with all values being assigned the negative sign to minimize them.

The reward for DDPG and SAC is modelled as follows:

$$r = (w_1 \times -TTS) + (w_2 \times -Q) + (w_3 \times -W) \tag{3-1}$$

where, $w_1 = 0.4, w_2 = 0.3 \quad \& \quad w_3 = 0.3$ and Q represents no. of vehicles in queue, W represents waiting vehicles on mainstream and TTS represents total time spent by all vehicles on the network.

3-3-4 Important Features in SUMO Traffic Environment

There are some important characteristics of the traffic environment that are important to define correctly for the simulations to work as intended.

Action Time

The action time, in this case, *delta time* represents the frequency with which an action is taken. In this traffic environment, the action time is set to 60 seconds for the ramp metering action as it is directly dependent on the action time in order to change the signal phases within a given cycle.

The variable speed limit actions are ideally independent of the action time and modelled as such in this environment as they need to be adjusted w.r.t the traffic conditions on the mainstream that are influenced by the demand, ramp conditions and actions performed on the ramp. In such a case, the frequency of the variable speed limit input will still be the same as the ramp metering input but can change independent of the 60 second cycle.

`reset` Function

The formulation of the `reset` function is important to the environment in order to set the initial states for each simulation that is performed. Additionally, this function outputs auxiliary information that is analogous to the information output by the `step` function described in the following subsection.

It is essential to define the function correctly, as poor definitions can cause logical errors, inconsistent object states, or unexpected behaviour due to wrong initial states and observations for the environment.

`step` Function

Defining the `step` function correctly for the user environment is important as this controls the changes taking place in the environment in each step of the simulation. Although the application is relatively straightforward, the definition depends on the environment that is created along with how and when the actions are supposed to be taken. The information that needs to be extracted for plotting figures or to display the end results are also described within this function.

In this thesis, for the traffic environment, the actions first needed to be brought into values usable for the SUMO network from another function and separated based on time of application. The ramp metering action takes place once every 60 seconds as shown in Section 3-3-4 since for this particular environment the sum of the red and green phases is 60. The actions for variable speed limits are independent of the action time and act as required by the changing states within the environment. Once these are established the calculation of the reward, collection of general information related to the environment, setting the termination condition and collecting specific simulation and result oriented information are described as per user requirements within the `step` function. In the following section, the important formulae and commands used to run the simulation will be described for ease of interpretation and understanding.

3-3-5 Commands and Functions

In this section, the various commands used to extract information from SUMO using the TraCI interface will be outlined along with the relevant formulae used to calculate traffic metrics along with their units.

The table 3-2 describes the formulae used to calculate relevant metrics for the reward as well as observations.

Metric	**Formula**	**Unit**
Density	$\frac{N_{veh}}{l_e}$	veh/m
Queue Vehicles	No. of vehicles in queue on the ramp	veh
Waiting Vehicles on Mainstream(Reward)	No.of vehicles waiting/delayed on the mainstream	veh
TTS	$\frac{[\sum_{i=0}^{4}(\rho_i \times l_e \times t_c) + (\rho_r \times l_r \times t_c)]}{3600}$	hours

Table 3-2: Formulae

For the TTS formula, ρ_i & ρ_r represents the density on each segment in the mainstream and the ramp respectively, l_c and l_r represents the length of the mainstream and ramp segments, and t_c represents the cycle time. The TTS is calculated at every cycle and given to the reward signal as a summation across all cycle times.

The table 3-3 details the various commands used to extract data from the simulation environment.

Value	TraCI Command	Detail
No. Of Vehicles	`getLastStepVehicleNumber`	Returns the number of vehicles on the edge or lane
Halted Vehicles	`getLastStepHaltingNumber`	Returns the number of vehicles halted on lane or edge
Waiting Time	`getWaitingTime`	Returns the time each vehicle waits on a lane or edge
Mean Speed	`getLastStepMeanSpeed`	Returns the mean speed of all vehicles on an edge or lane
Accumulated Waiting Time	`getAccumulatedWaitingTime`	Returns the total waiting time for all vehicles in the network
Occupancy	`getLastStepOccupancy`	Returns the percentage of space occupied by a vehicle on the detector

Table 3-3: TraCI Value Retrieval

3-4 Reinforcement Learning Agent Setup

In this thesis, two reinforcement learning agents are tested to improve traffic conditions: DDPG and SAC. There have already been several papers that have implemented DDPG and TD3 for traffic control problems but there has not been much literature in regards to SAC.

3-4-1 RL Agents

The implementation that is used for the purpose of this thesis is taken from the `stable-baselines3` package. The implementation applied for DDPG is derived from [21] and for SAC is derived from [10]. The `sb3` agents are used due to the ease of application to various problems and clear documentation on recording of agent information. The tuning of hyperparameters is also relatively straightforward as it can be done directly and does not need to alter anything internally within the implementation. The way the agents are implemented also leaves room to directly alter more complex and intricate hyperparameters such as neural network size, activation functions, optimizers, etc.

Chapter 4

Simulations and Results

The training and simulations for this thesis were all done on the TU Delft High Performance Computing centre's supercomputer, DelftBlue. The results and plots in this section will show the baselines used for comparison of the models, the relevant information regarding the each simulated agent environment, and comparison metrics across the different untrained and trained environments. Sections 4-1, 4-2, and 4-3 detail the individual plots for the various control methods. It should be noted that the above plots and metrics are using the default seed of SUMO and are singular simulation values. The last section 4-4 will detail the various metrics compared for all control methods and the final values for all methods.

4-1 Baseline

Comparing the results obtained from different control methods is important in order to gauge the improvement and feasibility of newer and/or more complex methods. Without such baselines, it would not be possible to determine if newer methods are sustainable in the long term. To that extent, in literature there seems to be a divide among what is considered baseline for traffic simulations.

4-1-1 No Control vs Fixed Time Control

In general, the baseline values would be taken from the no control case, i.e., the case where the ramp traffic lights are always green and there are no limits on the speed on the highway. This can be considered standard for most traffic scenarios like those modelled using METANET[16] or Flow[35] but when it comes to using external software to model traffic scenarios like SUMO[22] there can be certain exceptions.

Taking the case of [33] and [36] which both use SUMO to simulate traffic scenarios, the baselines used in the two papers differ, such that in the former fixed time control is used as a baseline and in the later no control is used as the baseline. It is to be noted that in

[33], the control method is combined RM-VSL whereas in [36] the control method is just VSL. Although no clear argument is given especially in the former regarding the choice of the baselines, after using and running multiple simulations for this thesis, the reason is the inherent behaviour of the software itself.

In case of any control method which includes ramp metering, no control seems to give the best results due to the inherent nature of the software to give priority to the vehicles at the ramp junction more than the vehicles upstream on the network. This sacrifices the average speed and certain amount of congestion to seemingly improve the overall throughput giving better TTS values. This means that regardless of the control method applied as soon as there are changes to the signal phases at the ramp the TTS values will increase but the overall network condition may be much smoother which may not be reflected in the TTS value. On the other hand, when it comes to purely VSL without any direct influence from ramp traffic signals, no control seems to be an appropriate choice since there is no priority given to specific vehicles on the mainstream of the highway. Thus, any adjustments to speed limits using any control method at appropriate segments of the highway will only result in better results overall since speed changes for the vehicles in the no control case will be more extreme comparatively. Taking into account these issues, the baseline for comparing combined RM-VSL control methods is chosen to be fixed time control wherein, the phase change for the signals is a constant 30 seconds of red and green in a complete 60 second cycle. But the values for no control case are also shown to elicit performance related analysis for the RL algorithms. The values and plot for no control and fixed time control are shown below:

Metrics	Values
Queue Length (m)	1654
Delay on Ramp (s)	549
Average Speed (m/s)	13.48
Total Delay on Mainstream (s)	505848
TTS (h)	425

Table 4-1: Values No Control

Metrics	Values
Queue Length (m)	14773.5
Delay on Ramp (s)	47850
Average Speed (m/s)	12.75
Total Delay on Mainstream (s)	605629
TTS (h)	460

Table 4-2: Values FTC

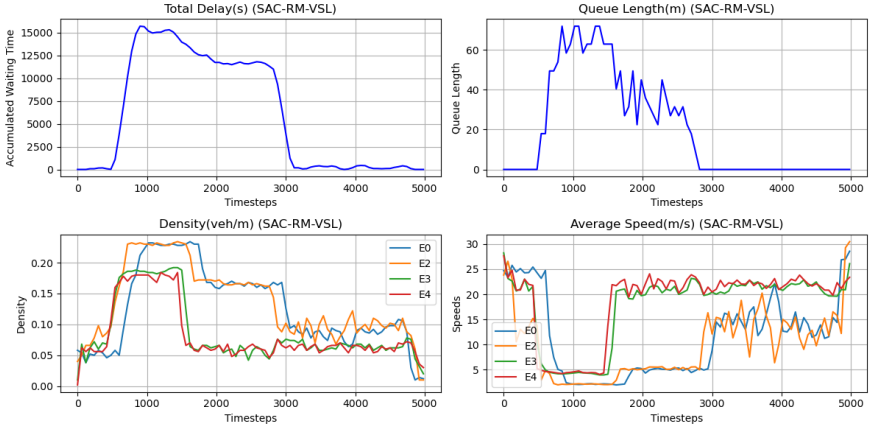


Figure 4-1: Metrics for No Control

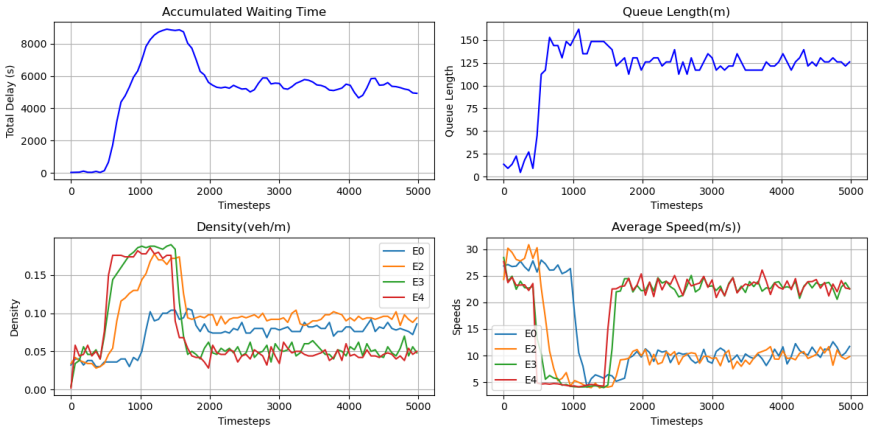


Figure 4-2: Metrics for Fixed Time Control

From the above table and figure we see that fixed time control is not a great control method for this scenario. Although it provides some form of control the overall metrics and behaviour of the network are not ideal. This is taken as the baseline for comparison with other algorithms.

4-2 ALINEA

In order to evaluate the effectiveness of the reinforcement learning algorithms, ALINEA [25] was deployed on the same environment as a benchmark for a good control method for ramp metering and to also test the correctness of the applied demand profile. This verification is an important step as it can determine whether the demand profile set for the network created is either too harsh to control, to lax to need any control or just enough for a basic algorithm like ALINEA to show some improvement.

In this scenario, the ALINEA algorithm controls the ramp metering rate of the ramp traffic signal instead of the flow across the ramp. The equation used for ramp metering rate control is shown below:

$$m_r = 1 - K(\rho_d - \rho_u) \tag{4-1}$$

where K is the gain, ρ_d is the desired density and ρ_u is the density of the segment upstream of the ramp. The ramp metering rate r is then converted into the signal phase timings by linear approximation as shown below:

$$t_g = m_r \times t_c \tag{4-2}$$

where t_g is the green time of the signal and t_c is the cycle time of 60 seconds. The gain K is set 100 to accommodate the density units which is veh/m in this case compared to the general veh/km.

The values and plots related to ALINEA control for two different demands, one with higher ramp demand and one with lower, are shown below:

Metrics	Values
Queue Length (m)	19174
Delay on Ramp (s)	598003
Average Speed (m/s)	18.36
Total Delay on Mainstream (s)	434328
TTS (h)	345

Table 4-3: Values ALINEA

Metrics	Values
Queue Length (m)	15669
Delay on Ramp (s)	392061
Average Speed (m/s)	22.69
Total Delay on Mainstream (s)	345462
TTS (h)	222

Table 4-4: Values ALINEA for Lower Demand

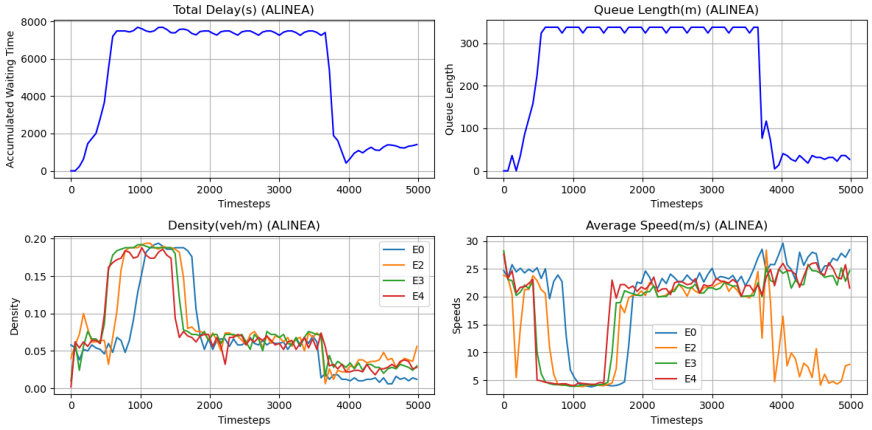


Figure 4-3: Metrics for ALINEA Control with High Demand

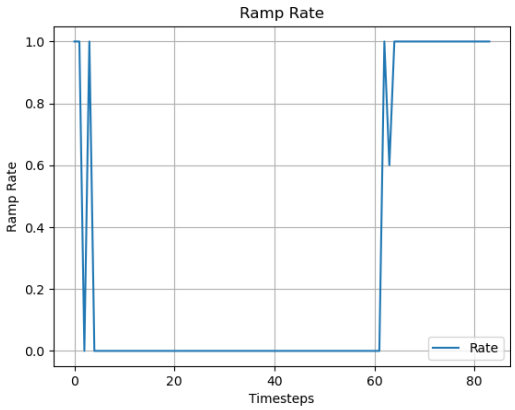


Figure 4-4: ALINEA Ramp Rate for High Demand

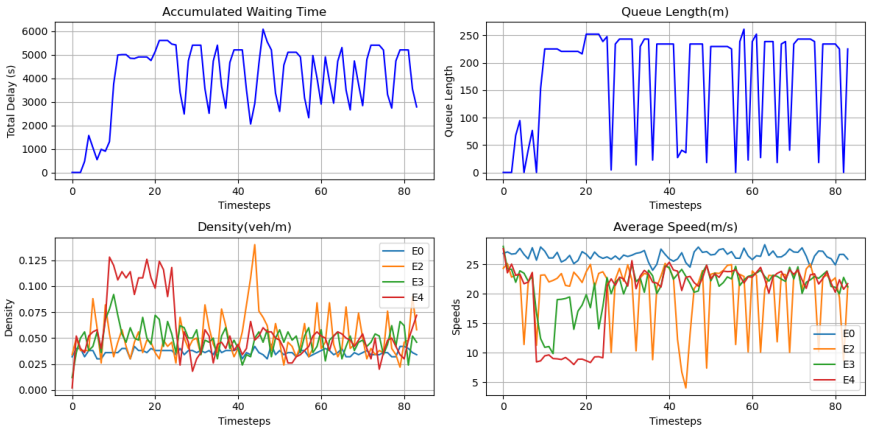


Figure 4-5: Metrics for ALINEA Control for Lower Demand

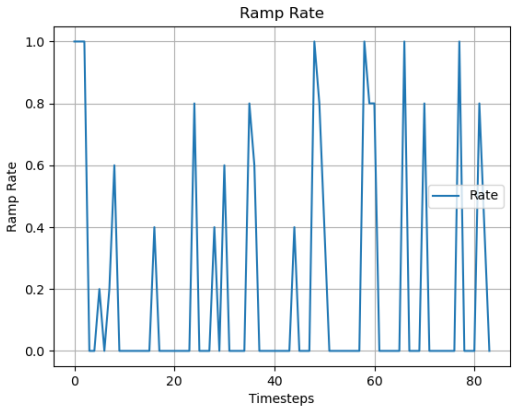


Figure 4-6: ALINEA Ramp Rate for Lower Demand

From the figures we observe that with a small change to the ramp demand, i.e., a reduction of 500 veh/hr during congestion from 2500 veh/hr to 2000 veh/hr, there is a marked difference in control when using ALINEA. The ALINEA algorithm works well for the latter but not the former with the same gain $K = 100$ and desired density $\rho_{des} = 0.04$. Here the density values are in veh/m and not veh/km to match the output from the environment. But the RL algorithms in the following sections have been applied to the scenario with higher demand as there was no improvement for the RL algorithms observed with the lower demand even after multiple simulations with various hyperparameters. The choice of demand profile in traffic control will also be discussed in section 5.

4-3 Reinforcement Learning

When applying reinforcement learning, in case of ramp metering, the action between $[0, 1]$ is converted to the green time for the phase using simple linear approximation as shown below:

$$t_{\mathrm{g}} = a_{\mathrm{rm}} \times (t_{\mathrm{c}} - t_{\mathrm{min}}) + t_{\mathrm{min}} \tag{4-3}$$

where t_g is the green time of the signal, t_c is the cycle time of 60 seconds, $t_{min} \leq 5$ is a minimum integer value given to the signal to avoid issues with having 0 seconds as the green time during simulation and a is the action taken by the model.

For the combined RM-VSL control in addition to the above shown ramp metering conversion, the action controlling variable speed limits is also similarly converted to the required value using linear approximation as shown below:

$$v_{\mathrm{lim}} = a_{\mathrm{vsl}} \times v_{\mathrm{d}} + v_{\mathrm{min}} \tag{4-4}$$

where v_{lim} is the speed limit, $v_{min} \geq 11.11m/s$ is the minimum speed that should be maintained along the highway, v_d is the difference between the $v_{max} = 33.33m/s$(in this scenario) and the v_{min} and a is the action taken by the agent. This precaution is again taken so as to prevent a situation of setting 0 speed limit for any duration of time to prevent unwanted congestions. This is a more pro-active method.

4-3-1 DDPG

In this section the metrics related to the DDPG control for both RM and combined RM-VSL are presented. The figures and values for RM with DDPG for single simulation with default SUMO settings are given below:

RM

The values and plots related to RM control using DDPG are shown below:

Metrics	Values
Queue Length (m)	1638
Delay on Ramp (s)	555
Average Speed (m/s)	13.17
Total Delay on Mainstream (s)	516248
TTS (h)	431

Table 4-5: Results DDPG (RM)

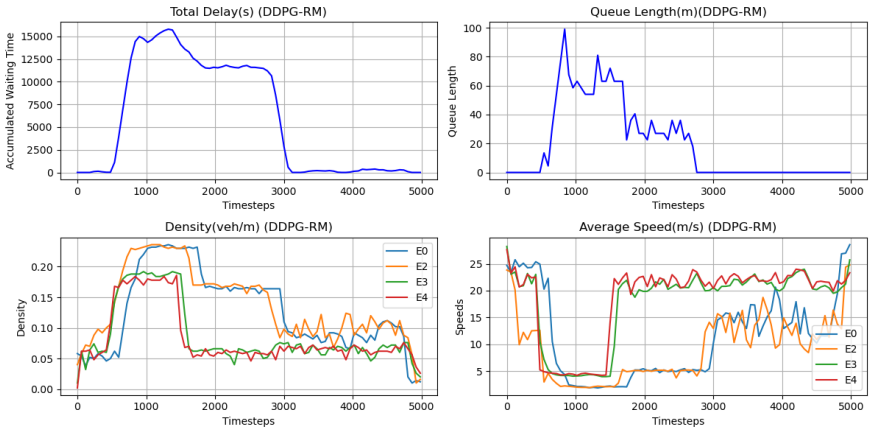


Figure 4-7: Metrics for DDPG (RM)

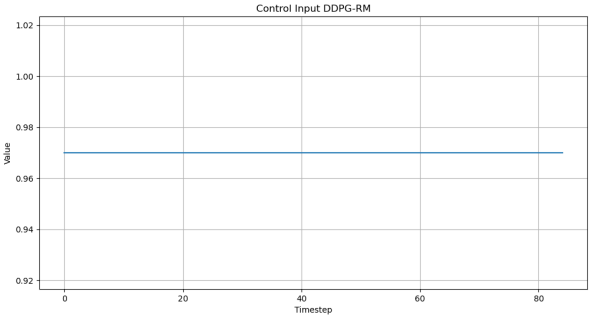


Figure 4-8: Control Input for DDPG (RM)

From the above, table and figures we see an improvement in queue length and ramp delay compared to FTC and ALINEA as expected from an algorithm focused on controlling the ramp metering rate. There is also an improvement in the TTS compared to FTC and ALINEA.

Combined RM-VSL

The values and plots related to combined RM-VSL control using DDPG are shown below:

Metrics	Values
Queue Length	1674
Delay on Ramp (s)	559
Average Speed (m/s)	13.28
Total Delay on Mainstream (s)	507748
TTS (h)	427

Table 4-6: Results DDPG (RM-VSL)

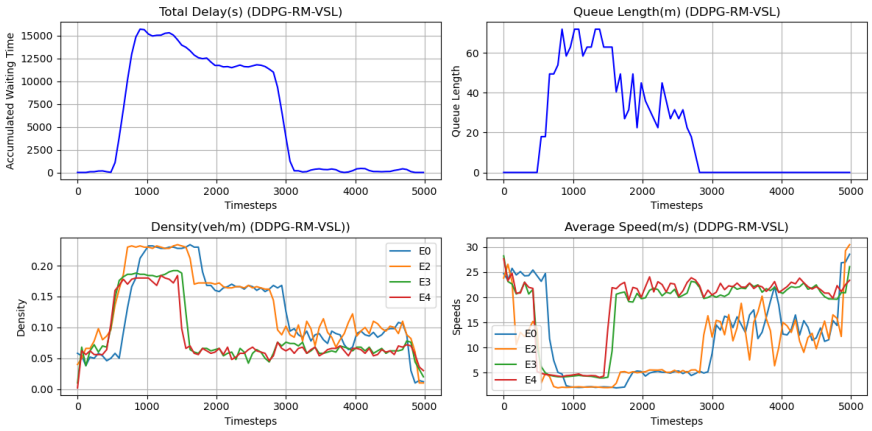


Figure 4-9: Metrics for DDPG (RM-VSL)

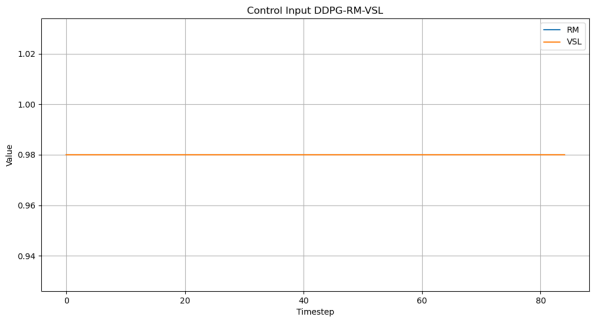


Figure 4-10: Control Input for DDPG (RM-VSL)

The effect of adding VSL is seen in the reduction in delay on the mainstream with a slight improvement to the average speed of vehicles on the network as seen from tables 4-6 and 4-5.

From fig. 4-8, the control input is similar to the no-control case with the value being constant but slightly below the all green value of 1. Similarly, in fig. 4-10, the control inputs actually overlap each other for both actions and are held at a constant value through the episode. This shows that the learned policy is quite poor such that it is unable to handle even two different actions in a way that they can be even controlled independently and is unable to match the no control results.

4-3-2 SAC

RM

The values and plots related to RM control using SAC are shown in table 4-7 and fig. 4-11:

Metrics	Values
Queue Length	1699
Delay on Ramp (s)	619
Average Speed (m/s)	13.25
Total Delay on Mainstream (s)	511162
TTS (h)	424

Table 4-7: Results SAC(RM)

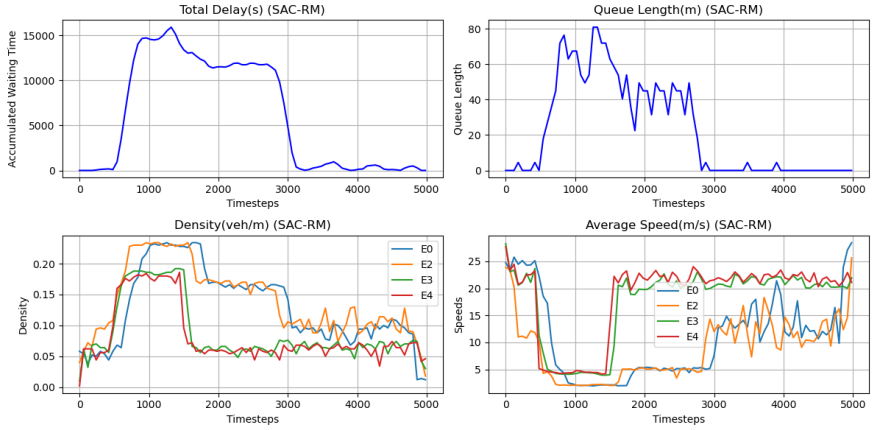


Figure 4-11: Metrics for SAC (RM)

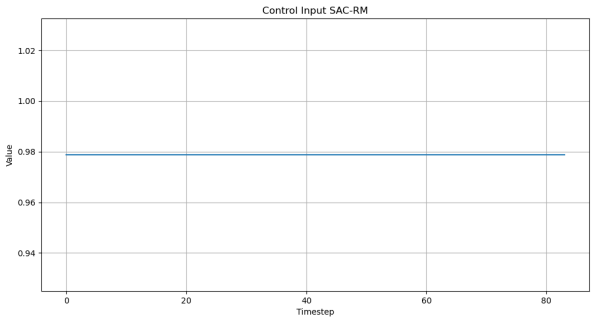


Figure 4-12: Control Inputs for SAC (RM)

The figure 4-12 shows a constant ramp metering rate across one entire episode slightly below the value of 1 which would be the no control case. Thus the policy is unable to learn proper values for control input according to varying traffic conditions and performs overall slightly worse than the no control case. The results and figure for ramp metering control using SAC show marginal improvements when compared to the ramp metering control using DDPG and is comparable to the performance of combined control using DDPG. As expected the queue length metric is slightly better due to the focus on ramp metering.

Combined RM-VSL

The values and plots related to combined RM-VSL control using SAC are shown in table 4-8 and fig 4-13:

Metrics	Values
Queue Length	1692
Delay on Ramp (s)	543
Average Speed (m/s)	13.57
Total Delay on Mainstream (s)	500421
TTS (h)	420

Table 4-8: Results SAC (RM-VSL)

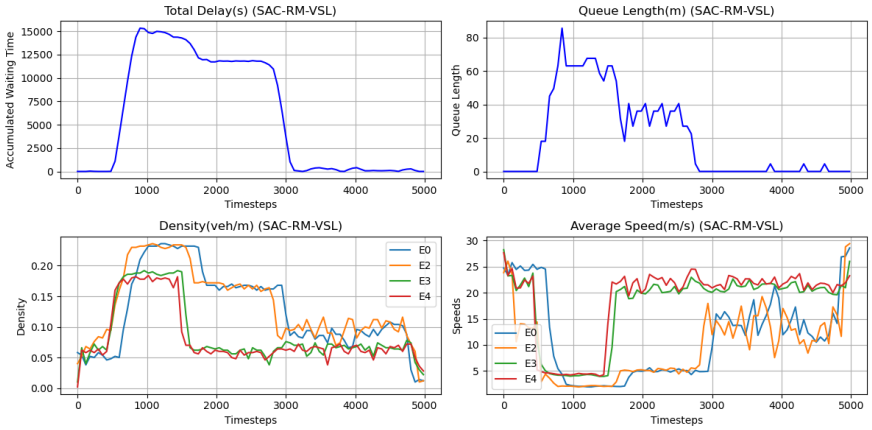


Figure 4-13: Metrics for SAC (RM-VSL)

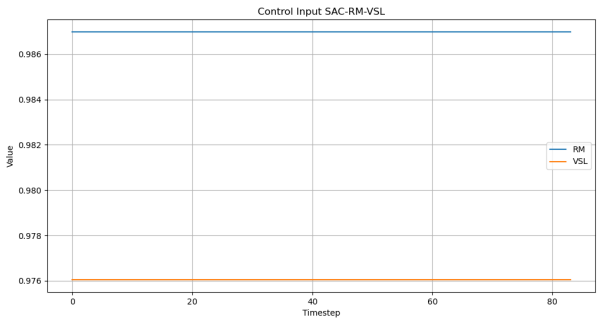


Figure 4-14: Control Inputs for SAC (RM-VSL)

From figure 4-14, we see that the policy learnt by the agent holds both control inputs at a constant value through the entire episode and achieves a balance of reward metrics with such a strategy for this environment. This is analogous to the no-control case (all actions

are set to 1) which is not a good learned policy. Combined RM-VSL control can be seen to show marginal improvements over all previous control methods with slight deviations and the overall metrics of the network have been improved across the board including the TTS. But from the control inputs obtained along with the fact that it is unable to significantly outperform the no control case, we can conclude that the algorithm does not perform too well in this scenario.

4-4 Comparison and Final Results

In this section, some of the metrics will be compared across the different control methods are shown. The first set of plots shown are the learning curves for each of the different agents.

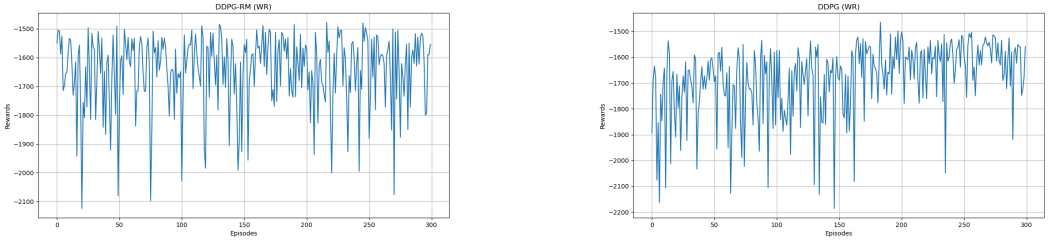
The following table shows the basic hyperparameters that were used to get the results shown in Section 4-3 and Section 4-4:

Hyperparamters	DDPG (RM)	DDPG (RM-VSL)	SAC (RM)	SAC (RM-VSL)
Learning Rate	0.0005	0.0005	0.003	0.003
γ	0.999	0.999	0.99	0.99
τ	0.00099	0.00099	0.005	0.005
Buffer Size	10^6	10^6	10^6	10^6
Batch Size	336	336	256	256
Action Noise	$\mathcal{OU}(0, 0.3)$	$\mathcal{OU}(0, 0.3)$	$\mathcal{OU}(0, 0.3)$	$\mathcal{OU}(0, 0.3)$
Entropy Coefficient	N/A	N/A	Auto	Auto

Table 4-9: Hyper-parameters for RL Agent Training

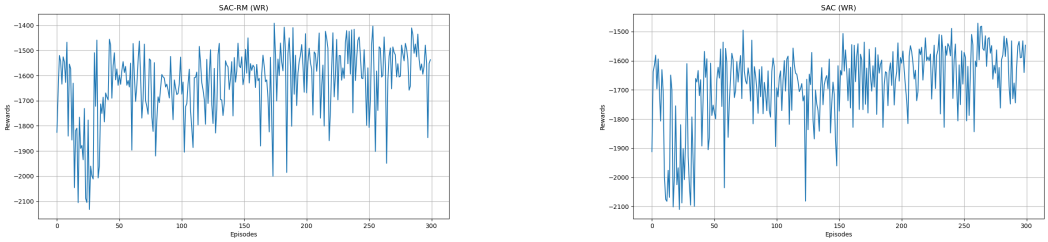
It is to be noted that the action noise used to promote exploration of policies in both agents is the Ornstein-Uhlenbeck Noise with $\mu = 0$ and $\sigma = 0.3$. These values were set after checking the performance of the agents with lower values of σ which gave poorer performance and higher values of σ which showed no particular improvement either in stability or learning performance.

Plots with learning curve over 300 episodes are depicted in figs. 4-15a- 4-16b. The SUMO seed was set to random for each episode of the training in order to show robustness of RL algorithms. From the plots, only figs. 4-15b, 4-16a, 4-16b show some robustness with improvements in training over 300 episodes.



(a) RM **(b)** RM-VSL

Figure 4-15: DDPG Learning Curves



(a) RM **(b)** RM-VSL

Figure 4-16: SAC Learning Curves

From figure 4-15a, we observe that there is no upward trend or improvement in the overall learning curve. This is possibly due to the algorithm finding a certain policy deemed to be adequate in the first few episodes and then applying the policy on future episodes that have the same demand but small changes in vehicle generation and position values caused by SUMO seed set to random.

As noted above, the bottom three figures show a gradual trend of improvement with all the final values for the reward being very close. From the tables and figures in section 4-3 we observe that the metrics are also consequently close for a single simulation run of the trained agents. The figures 4-17, 4-18, 4-19, and 4-20 depict the values represented as box-plots over 300 iterations for random seeds for the SUMO software different from training. The seed values in the SUMO software control the randomness and spread of the vehicles entering the traffic network without changing the demand.

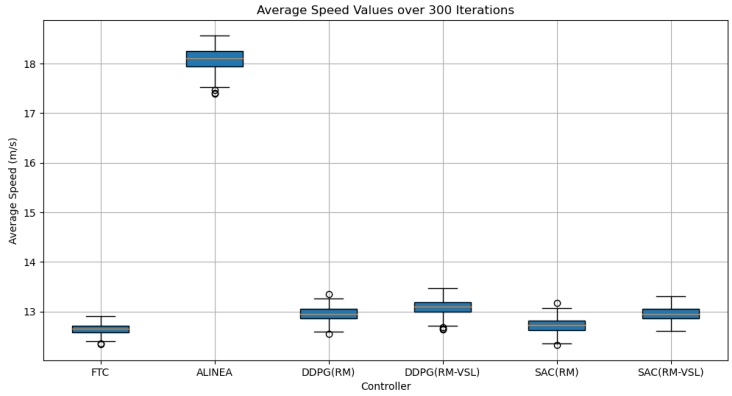


Figure 4-17: Average Speed

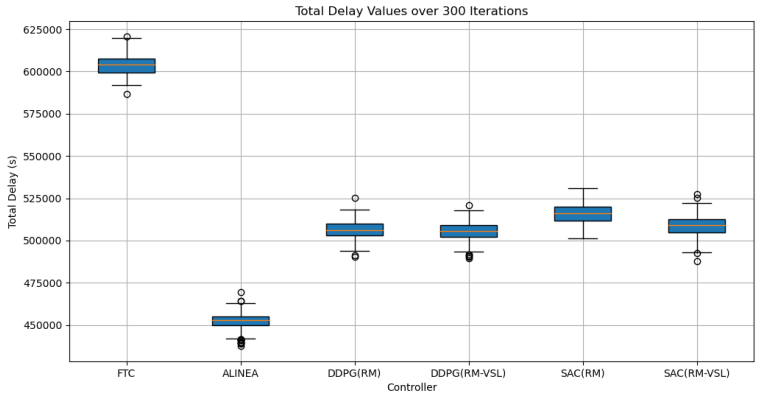


Figure 4-18: Delay on Mainstream

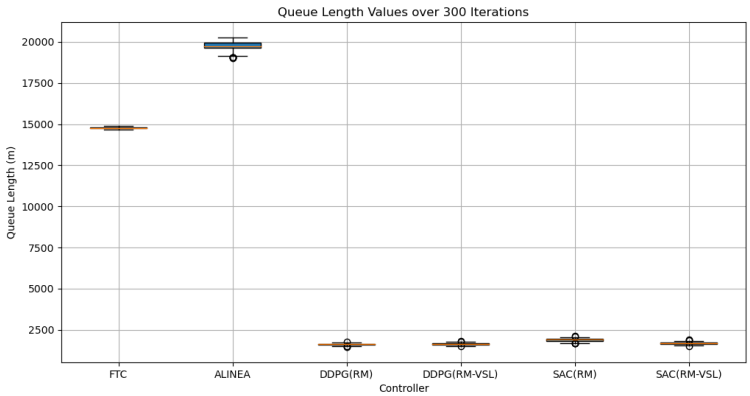


Figure 4-19: Queue Length

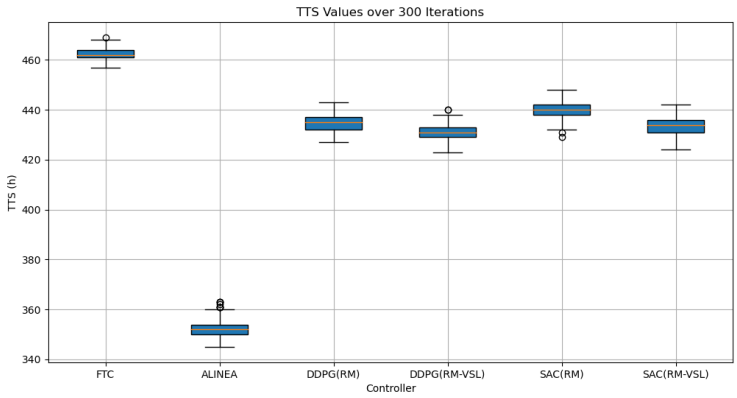


Figure 4-20: Total Time Spent

The following table shows are the representative median values for the above iterations with learned agents along with the average values for the no control case simulated separately:

Metrics / Model	TTS (h)	Total Delay Mainstream (s)	Avg. Speed (m/s)	Queue Length (m)	Reward (Single)
NC	429	506413	13.18	1664	-1491
FTC	462	635080	12.65	14763	-2156
ALINEA	352	452268	18.08	19743	-1670
DDPG(RM)	437	516265	13.15	1633	-1469
DDPG(RM-VSL)	433	515381	13.45	1638	-1475
SAC(RM)	440	515874	13.17	1635	-1490
SAC(RM-VSL)	433	508926	13.35	1640	-1481

Table 4-10: Final Values over 300 Iterations

The values in table 4-10 represent the average values of the metrics with the learned applied to the environment shown in figs. 4-17 - 4-20 along with the reward value for the simulation run depicted by figs. 4-2 - 4-13.

The average weighted reward values for the learned algorithm on testing have also been included in the table. These were added in order to show that with weighted rewards the

performance of the algorithm depends on the different values used in the reward. The improvement of one value does not necessarily guarantee an overall improvement in the environment simulation as is the case in the results marked with DDPG-U. In fact, although there is a significant reduction in the total time spent for those two simulations the queue length blows up and that is reflected in the final rewards values which are closer to the baselines with very little improvement.

In case of the DDPG and SAC simulations, it is observed that optimizing or improving the queue length is very slightly favoured in the purely RM case compared to the combined RM-VSL case. This is expected since the actions taken on the RM case only directly control the ramp signal timings and only indirectly affect the mainstream which in turn favours ramp related metrics. In case of RM-VSL case, while sacrificing a little improvement in queue length, the average speed, total delay and TTS are simultaneously improved compared to the used baselines but still behind the no control scenario.

Additionally, it is observed in this case that the SAC algorithm shows reduced returns in certain cases for the traffic network. This is attributed to the fact that in the case of this work the overall complexity of the traffic network is too low for SAC to learn. But this in turn shows promise that even with limiting factors the learning was comparable to simpler algorithms like DDPG and holds much more promise for higher dimensional action spaces with more ramps and speed limits along with more nuanced applications like autonomous vehicles in traffic.

The following figures show the single simulation comparisons of metrics. These simulations were done using the same method mentioned in Section 4-3.

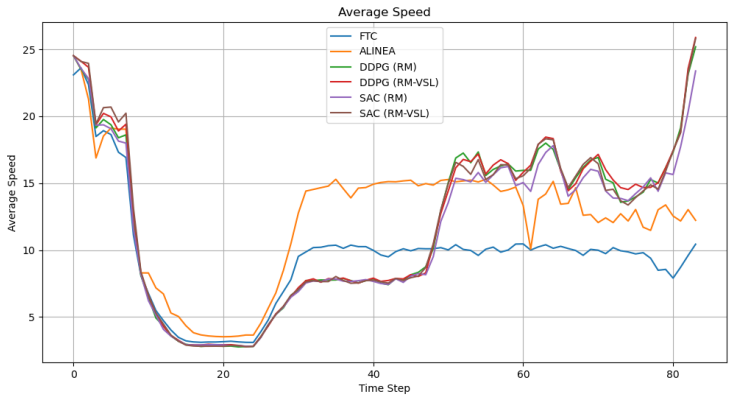


Figure 4-21: Average Speed across the Network

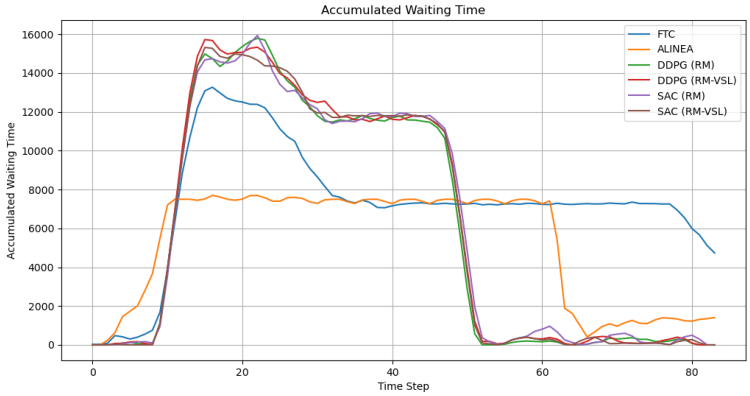


Figure 4-22: Delay on Mainstream

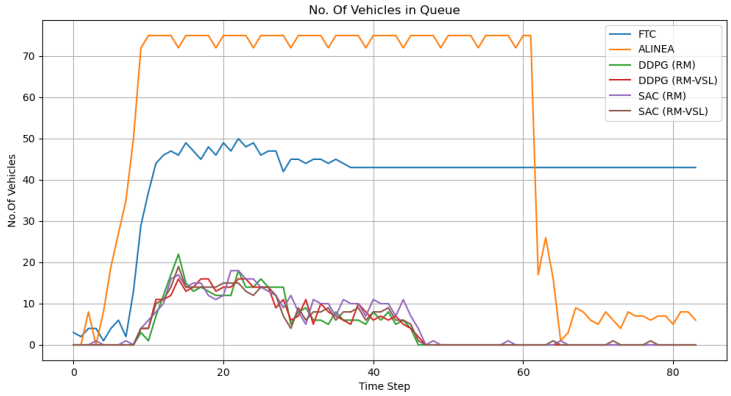


Figure 4-23: Queue Vehicles on the Ramp

The above figures show the single simulation progression in metrics for each of the control methods for one episode. A critical analysis of the merits and failures of the applied methods will be presented in Section 5.

Chapter 5

Conclusion

In this thesis, a simple framework for testing the Soft Actor Critic algorithm on traffic networks is proposed. Multiple simulations are performed using different control methods and algorithms on the given network and attempted to check the efficacy of the SAC algorithm in traffic control applications. The results show that in general SAC performs better than other control algorithms and when compared to DDPG for continuous actions SAC indeed is much better. However, certain observations have been made which will be detailed in Section 5-1.

5-1 Summary of Results

From the simulation results, it is observed that for TTS, SAC with combined RM-VSL achieves a 6.5% improvement over Fixed Time Control, a 9.8% improvement over ALINEA, and a 2% improvement over DDPG(RM), and SAC(RM) while matching the auto optimized results of DDPG(RM-VSL). It is to be noted that the improvement in ALINEA is calculated using the weighted reward since the other values are too extreme and show poor control for the given demand.

An observation regarding the use of hybrid sum of weighted values as the reward is that in this thesis having a better learning curve did not always equate to better performance on applying the model for testing. This observation was made when applying certain learned agents to the environment for testing and extracting the metrics from the simulations. Such curves might favour certain values within the reward function and tend to minimize those while neglecting the other values which might show up as good learning but lead to poor performance.

Another observation to note are the metrics and plots marked with suffix U for DDPG are simulated with the same hyperparameters as SAC to highlight the ease of manual hyperparameter selection for SAC compared to DDPG in case of a traffic scenario with continuous action space. The optimized version of DDPG was run over several days with quite narrow ranges for each hyperparameter in order to elicit the results displayed in the thesis. The

effectiveness of SAC will increase with an increase in complexity of the environment and of action spaces such that diverse and multiple actions can be accommodated.

In traffic simulations using SUMO with control using Python, there are two cases of randomness that need to be taken into account. One is inherently from the SUMO software which can be overcome by doing all simulations on a single seed value. The second comes from the RL algorithms within Python. In the case of this thesis, the simulations were done only using a single seed for RL algorithms, due to length of simulations, multiple algorithms to be tested and availability of slots in the TU Delft Supercomputer. The difference in seed values or randomness in the RL algorithms manifests in the way of differing starting points for policy selection and action selection and can affect exploration inturn affecting the training process.

The constant control input values even after sensitivity analysis/hyperparameter tuning show that the policies being learnt by the RL agents are being limited by the environment definition and the traffic scenario. The policy being learnt is similar to the no control case and although this shows that the agent learns from the trend in the traffic scenario, it also shows that the agent is unable to provide any significant improvement from the no control case. A detailed analysis of the possible causes is discussed in section 5-2.

5-2 Analysis of Results

5-2-1 Reward Function

The figures 4-15a- 4-16b in section 4-4 show that the learning for DDPG and SAC is quite unstable with reward values fluctuating heavily throughout the learning duration. SAC still is slightly better with an upward trend after 100 episodes but still not smooth. This is due to the flawed definition of the reward function. The reward function described in equation 3-1 uses the total time spent, vehicles in queue and in mainstream. Although the reward specifically penalizes an overall inefficiency in the network it may be too delayed and sparse causing training instability since:

- Total time spent is a delayed reward as it can be accurately predicted only at the end of each episode. This means that actions taken early on in the simulation, when the values are collected over each cycle time, may not have any significant feedback.
- Conversely, waiting vehicles on mainstream and queue vehicles are more immediate rewards but have a tendency to be noisy.
- The difference in reward timings can cause reward sparsity which could in turn cause the degradation in gradients while learning causing it to stall.

5-2-2 Continuous Control

The action space defined is continuous, i.e., it can take on any values within a given range ([0, 1] in the case of this environment). It is in the nature of such an action space to be inherently noisy which can cause erratic signals for both variable speed limits and ramp metering in turn leading to poor selection of actions even for learned agents leading to:

1. Shockwaves
2. Ramp overloading
3. Destabilize mainline flow

But the lack of action noise can lead to a standstill in learning for the RL agents since it might prevent proper exploration.

The network structure and the consequent information provided to the observation space in this work for such a complex and noisy environment might have been lacking even though the metrics for all segments were input individually as well as for the network as a whole, i.e., to have both local and global network information to process states and actions.

5-2-3 Conflict between Objectives

The use of a single RL agent to control two objectives could have lead to conflict between the ramp metering and variable speed limit and ramp metering objectives resulting in an inability to learn coordinated policies. From the results, it can be deduced that in many cases the objectives are indeed clashing with each other causing not only the unstable learning curves with volatile rewards but also a lack in hierarchy while learning with the RL agents.

5-2-4 Demand Modelling

As seen in section 4-2, demand modelling is also an important aspect of traffic control. When working with virtual scenarios, it is important to create a demand sufficiently harsh to justify the needs for control algorithms but also not unrealistically harsh that nothing may work. The section 4-2 perfectly shows that in certain cases just simple control like ALINEA is more than sufficient without the need for complex RL control methods but in some cases ALINEA is not able to adapt at all with a slight increase in demand.

5-2-5 RL Agent and SUMO Limitations

The use of predefined RL agents from Python libraries also played a part in poor training performance as the modularity was much lower compared to self defined RL agents. This lead to a set path of defining the RL agents and tuning of hyperparameters to in order to improve performance. Some other limitations are as follows:

- DDPG relied on noise based exploration which caused instability in learning as seen in section 4-4
- Poor ability to tune entropy in SAC and rely on the predefined agent decreased the overall performance of the SAC agent
- Vehicle behaviour prediction within SUMO along with randomness in vehicle generation although addressed still caused instability in the simulations especially when assessing ALINEA control.

DDPG and SAC perform marginally better than baseline control cases but worse than the no control case due to a combination of the reasons mentioned in section 5-2. Although, model-free agents such as DDPG and SAC are adaptable, the simple application to a complex environment lead to degraded performance in this case.

Keeping in mind the above considerations and looking at the results under these constraints, it can be said that SAC can be a better option compared to DDPG for freeway traffic control especially with continuous actions. The improvements to the models compared to the baselines are lower than expected from the initial idea for the work and still worse than no control scenario, thus it cannot be considered a complete success but it does show that SAC can perform better with continuous actions and has the potential to perform even better with minor adjustments compared to DDPG. This can be used to manage traffic across multiple sections and intersections using a single algorithm with real time data. To that extent, the following chapter 6 lists the adjustments that can be made to the methodology used in this work to improve the results for potential future implementations.

Chapter 6

Further Work

After reviewing the work done in this thesis, there is definitely room for growth and improvement in the direction of application of reinforcement learning algorithms to highway traffic problems. This thesis has already demonstrated the viability of using continuous actions in order to control the ramp traffic signals and variable speed limits.

First, with an improvement in the understanding of the reinforcement learning algorithms, user coded algorithms that mimic DDPG, TD3 and SAC can be deployed with wider fine tuning options for hyperparameters such as:

- Independent learning rates for actor and critic networks to improve learning and stability and handle delayed rewards
- Implementing wider range of combinations of activation functions and optimizers
- Adjusting the size and structure of the neural networks for both actor and critic
- For SAC, create an efficient entropy tuning function for exploration
- Creating provisions to improve data collection and plotting such as episode wise iteration, multiple random seed iterations, displaying model data along with output to log simulations with ease.

Second, prioritize reward shaping by choosing appropriate combination of signals to build hybrid rewards. For rewards with different units, reward normalization should be applied to better equip the RL agents in training. Account for delayed and immediate rewards if they are combined by using a function to accumulate immediate rewards and delay their effect on the reward to prevent reward sparsity.

Third, based on the understanding of the simulation software and user defined environment, the observation space can be expanded to its smallest bits by recording values for every lane including density, inflow, outflow and any information that the user deems relevant to their application.

Fourth, adjusting the network structure to reduce segment lengths to improve the frequency and volume at which data is collected. This can bridge the gap between the actor and critic updates by giving more information to the state to apply better actions.

Lastly, multi agent reinforcement learning to coordinate different objectives can be implemented especially with continuous control to mitigate conflicts between tasks and better handle the noisy actions.

With the right tools and advancing technology in the machine learning field, the above suggestions might only be years away from being put into practical application.

Appendix A

Appendix A

The algorithms in Appendix A correspond to the reinforcement learning models described in 2. Basic algorithms along with some augmented algorithms and variations are also described.

A-1 DDPG

Algorithm 1 DDPG Algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^Q)$ with weights θ^Q and θ^μ
Initialize target network $Q^{'}$ and $\mu^{'}$ with weights $\theta^{Q^{'}} \leftarrow \theta^Q$, $\mu^{Q^{'}} \leftarrow \mu^Q$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize random process $\mathcal{N}$ for process exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu + \mathcal{N}_t$ according to current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q' \left(s_{i+1}, \mu' \left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
    Update the critic by minimizing the loss:$L = \frac{1}{N} \sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^Q\right)\right)^2$
    Update actor policy using sampled policy gradient: $\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q\left(s, a \mid \theta^Q\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu\left(s \mid \theta^\mu\right)\Big|_{s_i}$
    Update the target networks: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$, $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

A-2 TD3

Algorithm 2 TD3 Algorithm

Initialize critic networks Q_{θ_1}, Q_{θ_2} and actor network π_ϕ with random parameters θ_1, θ_2 and ϕ.
Initialize target networks $\theta_1^{'} \leftarrow \theta_1$, $\theta_2^{'} \leftarrow \theta_2$ and $\phi^{'} \leftarrow \phi$.
Initialize replay buffer $\mathcal{B}$.
for t = 1, T **do**
Select action with exploration noise $a \sim \pi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new stat $s^{'}$.
Store transition tuple $(s, a, r, s^{'})$ in $\mathcal{B}$
Sample mini batch of N transitions $(s, a, r, s^{'})$ from $\mathcal{B}$
$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
$y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
if t mod d **then**
Update ϕ by the deterministic policy gradient:

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\Big|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$$

Update target networks:

$$\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$$

end if
end for

A-3 SAC

Algorithm 3 SAC Algorithm

Require: The learning rates λ_π, λ_Q, λ_V for the functions π_θ, Q_w, V_ψ respectively; weighting factor τ for exponential moving average

Initialize parameters θ, w, ψ and $\bar{\psi}$

for each iteration **do**

for each environment **do**

$a_t \sim \pi_\theta(a_t|s_t)$

$s_{t+1} \sim \rho_\pi(s-t+1|s_t, a_t)$

$\mathcal{D} \leftarrow \mathcal{D} \cup s_t, a_t, r(s_t, a_t, s_{t+1}$

for each gradient step update **do**

$\psi \leftarrow \psi - \lambda_V \nabla_\psi J_V(\psi)$

$w \leftarrow w - \lambda_Q \nabla_w J_Q(w)$

$\theta \leftarrow \theta - \lambda_\pi \nabla_\theta J_\pi(\theta)$

$\bar{\psi} \leftarrow \tau\psi - (1-\tau)\bar{\psi}$

end for

end for

end for

Appendix B

Appendix B

B-1 Lemma 1

Lemma 1 (Soft Policy Evaluation). Consider the soft Bellman backup operator $\mathcal{T}^{\pi}$ in Equation 2-21 and a mapping $Q^0 : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ with $|\mathcal{A}| < \infty$, and define $Q^{k+1} = \mathcal{T}^{\pi} Q^k$. Then the sequence Q^k will converge to the soft Q-value of π as $k \rightarrow \infty$.

Proof. Define the entropy augmented reward as $r_{\pi}\left(\mathrm{s}_t, \mathrm{a}_t\right) \triangleq r\left(\mathrm{~s}_t, \mathrm{a}_t\right) + \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[\mathcal{H}\left(\pi\left(\cdot \mid \mathrm{s}_{t+1}\right)\right)\right]$ and rewrite the update rule as,

$$Q\left(\mathbf{s}_t, \mathbf{a}_t\right) \leftarrow r_{\pi}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p, \mathbf{a}_{t+1} \sim \pi}\left[Q\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right] \tag{B-1}$$

and apply the standard convergence results for policy evaluation [31]. The assumption $|\mathcal{A}| < \infty$ is required to guarantee that the entropy augmented reward is bounded.

B-2 Lemma 2

Lemma 2 (Soft Policy Improvement) Let $\pi_{old} \in \Pi$ and let π_{new} be the optimizer of the optimization problem defined in Equation . Then $Q^{\pi_{\text{new}}}\left(\mathrm{s}_t, \mathrm{a}_t\right) \geq Q^{\pi_{\text{old}}}\left(\mathrm{s}_t, \mathrm{a}_t\right)$ for all $\left(\mathrm{s}_t, \mathrm{a}_t\right) \in \mathcal{S} \times \mathcal{A}$ with $|\mathcal{A}| < \infty$.

Proof. Let $\pi_{old} \in \Pi$ and let $Q^{\pi_{old}}$ and $V^{\pi_{old}}$ be the corresponding soft state-action value and soft state value, and let π_{new} be defined as,

$$\begin{aligned} \pi_{\text{new}}\left(\cdot \mid \mathrm{s}_t\right) &= \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'\left(\cdot \mid \mathbf{s}_t\right) \| \exp\left(Q^{\pi_{\text{old}}}\left(\mathrm{s}_t, \cdot\right) - \log Z^{\pi_{\text{old}}}\left(\mathbf{s}_t\right)\right)\right) \\ &= \arg\min_{\pi' \in \Pi} J_{\pi_{\text{old}}}\left(\pi'\left(\cdot \mid \mathbf{s}_t\right)\right) \end{aligned} \tag{B-2}$$

It must be the case that, $J_{\pi_{\text{old}}}\left(\pi_{\text{new}}\left(\cdot \mid \mathbf{s}_t\right)\right) \leq J_{\pi_{\text{old}}}\left(\pi_{\text{old}}\left(\cdot \mid \mathbf{s}_t\right)\right)$, since we can always chose $\pi_{new} = \pi_{old} \in \Pi$. Hence,

$$\mathbb{E}_{\mathbf{a}_t\sim\pi_{new}}[\log \pi_{new}\left(\mathbf{a}_t \mid \mathbf{s}_t\right) - Q^{\pi_{old}}\left(\mathbf{s}_t, \mathbf{a}_t\right) + \log Z^{\pi_{old}}\left(\mathbf{s}_t\right)] \leq \mathbb{E}_{\mathbf{a}_t\sim\pi_{old}}[\log \pi_{old}(\mathbf{a}_t \mid \mathbf{s}_t) - Q^{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t) + \log Z^{\pi_{old}}(\mathbf{s}_t)] \tag{B-3}$$

and since partition function $Z^{\pi_{old}}$ depends only on the state, the inequality reduces to,

$$\mathbb{E}_{\mathbf{a}_t\sim\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) - \log \pi_{\text{new}}\left(\mathbf{a}_t \mid \mathbf{s}_t\right)\right] \geq V^{\pi_{\text{old}}}\left(\mathbf{s}_t\right) \tag{B-4}$$

Next, consider the soft Bellman equation:

$$\begin{aligned} Q^{\pi_{\text{old}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) &= r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1\sim p}}\left[V^{\pi_{\text{old}}}\left(\mathbf{s}_{t+1}\right)\right] \\ &\leq r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma \mathbb{E}_{\mathbf{s}_{t+1\sim p}}\left[\mathbb{E}_{\mathbf{a}_{t+1}\sim\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right) - \log \pi_{\text{new}}\left(\mathbf{a}_{t+1} \mid \mathbf{s}_{t+1}\right)\right]\right] \\ &\vdots \\ &\leq Q^{\pi_{\text{new}}}\left(\mathbf{s}_t, \mathbf{a}_t\right) \end{aligned} \tag{B-5}$$

where we have repeatedly expanded $Q^{\pi_{old}}$ on the RHS by applying the soft Bellman equation and the bound in Equation B-4 . Convergence to $Q^{\pi_{new}}$ follows from Lemma 1.

B-3 Theorem 1

Theorem 1 (Soft Policy Iteration). Repeated application of soft policy evaluation and soft policy improvement to any $\pi in \Pi$ converges to a policy π^* such that$Q^{\pi^*}\left(\mathrm{s}_t, \mathrm{a}_t\right) \geq Q^{\pi}\left(\mathrm{s}_t, \mathrm{a}_t\right)$ for all $\pi \in \Pi$ and $\left(\mathrm{s}_t, \mathbf{a}_t\right) \in \mathcal{S} \times \mathcal{A}$, assuming $|\mathcal{A}| < \infty$.

Proof. Let π_i be the policy at iteration i. By Lemma 2, the sequence Q^{π_i} is monotonically increasing. Since Q^{π} is bounded above for $\pi in \Pi$ (both the reward and entropy are bounded), the sequence converges to some π^*. We still need to show that π^* is indeed optimal. At convergence, it must be case that$J_{\pi^*}\left(\pi^*\left(\cdot \mid \mathbf{s}_t\right)\right) < J_{\pi *}\left(\pi\left(\cdot \mid \mathbf{s}_t\right)\right)$ for all $\pi \in \Pi, \pi \neq \pi^*$. Using the same iterative argument as in the proof of Lemma 2, we get$Q^{\pi^*}\left(\mathbf{s}_t, \mathbf{a}_t\right) > Q^{\pi}\left(\mathrm{s}_t, \mathbf{a}_t\right)$ for all $\left(\mathbf{s}_t, \mathbf{a}_t\right) \in \mathcal{S} \times \mathcal{A}$, that is, the soft value of any other policy in Π is lower than that of the converged policy. Hence π^* is optimal in Π.

Bibliography

[1] Lucas N. Alegre. SUMO-RL. https://github.com/LucasAlegre/sumo-rl, 2019.

[2] Nicolas Bach, Andrew Melnik, Malte Schilling, Timo Korthals, and Helge Ritter. Learn to move through a combination of policy gradient algorithms: Ddpg, d4pg, and td3. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 631–644. Springer, 2020.

[3] Kanok Boriboonsomsin and Matthew Barth. Impacts of freeway high-occupancy vehicle lane configuration on vehicle emissions. *Transportation Research Part D: Transport and Environment*, 13(2):112–125, 2008.

[4] Mohsen Davarynejad, Andreas Hegyi, Jos Vrancken, and Jan van den Berg. Motorway ramp-metering control with queuing consideration using q-learning. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1652–1658. IEEE, 2011.

[5] Ahmed Fares and Walid Gomaa. Freeway ramp-metering control based on reinforcement learning. In *11th IEEE International Conference on Control & Automation (ICCA)*, pages 1226–1231. IEEE, 2014.

[6] Ahmed Fares and Walid Gomaa. Multi-agent reinforcement learning control for ramp metering. In *Progress in Systems Engineering*, pages 167–173. Springer, 2015.

[7] William Fedus, Prajit Ramachandran, Rishabh Agarwal, Yoshua Bengio, Hugo Larochelle, Mark Rowland, and Will Dabney. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pages 3061–3071. PMLR, 2020.

[8] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[9] Martin Gregurić, Krešimir Kušić, Filip Vrbanić, and Edouard Ivanjko. Variable speed limit control based on deep reinforcement learning: A possible implementation. In *2020 International Symposium ELMAR*, pages 67–72. IEEE, 2020.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[11] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[12] Andreas Hegyi, Tom Bellemans, Bart De Schutter, and RA Meyers. Freeway traffic management and control., 2009.

[13] Andreas Hegyi, Bart De Schutter, and Johannes Hellendoorn. Optimal coordination of variable speed limits to suppress shock waves. *IEEE Transactions on intelligent transportation systems*, 6(1):102–112, 2005.

[14] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[15] A Kotsialos, M Papageorgiou, and A Messmer. Optimal coordinated and integrated motorway network traffic control. In *14th International Symposium on Transportation and Traffic TheoryTransportation Research Institute*, 1999.

[16] Apostolos Kotsialos, Markos Papageorgiou, Christina Diakaki, Yannis Pavlis, and Frans Middelham. Traffic flow modeling of large-scale motorway networks using the macroscopic modeling tool metanet. *IEEE Transactions on intelligent transportation systems*, 3(4):282–292, 2002.

[17] Apostolos Kotsialos, Markos Papageorgiou, and Frans Middelham. Optimal coordinated ramp metering with advanced motorway optimal control. *Transportation Research Record*, 1748(1):55–65, 2001.

[18] Phil Lasley. 2023 urban mobility report. *Independent*, 2023.

[19] Shubin Li, Tao Wang, Hualing Ren, Baiying Shi, Xiangke Kong, Jianyong Chai, and Xuejuan Wang. Variable speed limit strategies based on the macro hierarchical control traffic flow model. *Journal of advanced transportation*, 2021, 2021.

[20] Zhibin Li, Pan Liu, Chengcheng Xu, Hui Duan, and Wei Wang. Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks. *IEEE transactions on intelligent transportation systems*, 18(11):3204–3217, 2017.

[21] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[22] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[23] Chao Lu, Jie Huang, Lianbo Deng, and Jianwei Gong. Coordinated ramp metering with equity consideration using reinforcement learning. *Journal of Transportation Engineering, Part A: Systems*, 143(7):04017028, 2017.

[24] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[25] Markos Papageorgiou, Habib Hadj-Salem, Jean-Marc Blosseville, et al. Alinea: A local feedback control law for on-ramp metering. *Transportation research record*, 1320(1):58–67, 1991.

[26] Thorsten Schmidt-Dumont and Jan H van Vuuren. Decentralised reinforcement learning for ramp metering and variable speed limits on highways. *IEEE Transactions on Intelligent Transportation Systems*, 14(8):1, 2015.

[27] David Schrank, Bill Eisele, Tim Lomax, Jim Bak, et al. 2015 urban mobility scorecard. *Independent*, 2015.

[28] Silvia Siri, Cecilia Pasquale, Simona Sacone, and Antonella Ferrara. Freeway traffic control: A survey. *Automatica*, 130:109655, 2021.

[29] Kenneth A Small, Clifford Winston, and Carol A Evans. *Road work: A new highway pricing and investment policy*. Brookings Institution Press, 2012.

[30] Dingshan Sun, Anahita Jamshidnejad, and Bart De Schutter. A novel framework combining mpc and deep reinforcement learning with application to freeway traffic control. *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Erwin Walraven, Matthijs TJ Spaan, and Bram Bakker. Traffic flow optimization: A reinforcement learning approach. *Engineering Applications of Artificial Intelligence*, 52:203–212, 2016.

[33] Chong Wang, Yang Xu, Jian Zhang, and Bin Ran. Integrated traffic control for freeway recurrent bottleneck based on deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[34] Chong Wang, Jian Zhang, Linghui Xu, Linchao Li, and Bin Ran. A new solution for freeway congestion: Cooperative speed limit control using distributed reinforcement learning. *IEEE Access*, 7:41947–41957, 2019.

[35] Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 10, 2017.

[36] Yuankai Wu, Huachun Tan, Lingqiao Qin, and Bin Ran. Differential variable speed limits control for freeway recurrent bottlenecks via deep actor-critic algorithm. *Transportation research part C: emerging technologies*, 117:102649, 2020.

[37] Yue Zhou, Kaan Ozbay, Pushkin Kachroo, and Fan Zuo. Ramp metering for a distant downstream bottleneck using reinforcement learning with value function approximation. *Journal of Advanced Transportation*, 2020, 2020.

[38] Feng Zhu and Satish V Ukkusuri. Accounting for dynamic speed limit control in a stochastic traffic environment: A reinforcement learning approach. *Transportation research part C: emerging technologies*, 41:30–47, 2014.

Glossary

List of Acronyms