Master's Degree in:

ICT Innovation

Master Thesis

Reinforcement Learning in Traffic Control for Connected Automated Vehicles

Supervisor TUBerlin:
Prof. Sahin Albayrak

Supervisor UniTrento:
Prof. Danielle Fontanelli

Student:
Aristeidis Noulis

ACADEMIC YEAR 2019/2020

Abstract

The last years, more and people are concentrating in big cities for reasons of living and working. This effect has already some negative impacts on transportation networks including congestion and inefficiency. Parallel to the centralization, the number of autonomous vehicles on roads is continuing to grow, without completely replacing human driving vehicles. The upcoming mixed autonomy traffic situations will bring more dangers in terms of safety and transportation efficiency. The traditional traffic management solutions may not be able to handle these situations. Machine learning approaches have been already proved efficient in various complex fields. In this dissertation, a sub-field of Machine Learning, the Deep Reinforcement Learning will be investigated for enabling a smooth coexistence of automated, connected, and conventional vehicles. In particular, various reinforcement learning models, with both single and multi agent approaches, will be trained and tested on controlling the traffic flow in a specific mixed autonomy traffic scenario, where a transition from autonomous to human driving mode is needed for the vehicles.

Contents

List of Figures

List of Tables

Introduction

1.1 Motivation

In the coming years, the number of autonomous vehicles on roads will continue to grow. However, the time that conventional, human-driven vehicles will not exist anymore, but only autonomous ones will travel around the transportation network, will be long enough to come. During the times of autonomous and conventional vehicles coexistence, many unexpected incidents, which are already being handled by humans drivers without a special prior training, may be still problematic for the autonomous vehicles.

Mixed autonomy traffic will present problems for both safety and efficiency. In cases where the autonomous vehicles cannot operate normally or handle upcoming traffic situations properly, the vehicles should change to human driving mode. These cases are numerous and vary from a simple emergency stop, an accident avoidance or special driving behaviour because of a road deficiency. This transition cannot be always smooth, fast or perfectly completed on time and could lead to traffic disruptions in the transportation network.

Furthermore, the already available traffic management techniques may not be effective enough to deal with such problems considering aspects like urban sprawl and centralization. Additionally, the traditional traffic management methods should be adjusted and changed for dealing with the new circumstances. This cannot always be done in short period or it may need more resources than the available ones.

In the era of big changes and continues disruptions, a new need for adaptation and handling of these mixed autonomy traffic situations has been borned in order to avoid chaotic or dangerous incidents in transportation networks.

Machine learning techniques and approaches have already proved their success in various fields and sectors, by outperforming on human solutions in most of the cases. The traffic management field is one of the areas where machine learning methods have presented significant results. Especially a subset of machine learning, the deep reinforcement learning, has been used

for experimentation purposes in the complex transportation environments.

For future traffic control in various mixed autonomy traffic scenarios for connected and automated vehicles, an extensive investigation for strategies, which will be based on reinforcement learning approaches, should be implemented. These approaches could produce results with a significant impact on traffic management solutions, disrupting, validating or improving them.

1.2 Related Work

Until today, there are solid and popular projects which are trying to merge reinforcement learning techniques with traffic management scenarios for many and various reasons. Some of them are working on the optimization of the traffic light control in one intersection ore more, for a small part or a whole city. Whereas some others have as project goal to provide an ideal environment for end users, in order to test and improve their custom traffic management scenarios with predefined reinforcement algorithms.

The incorporation of reinforcement learning algorithms with traffic management scenarios has been already happened extensively. However, there are not many implemented researches on autonomy and mixed autonomy traffic situations, as most of the project are concentrated in common and familiar traffic problems.

As the days of having autonomous vehicles in the same transportation network with conventional ones, are becoming an expected reality, there is need for implementing research in this area. This need is behind the main approach in this dissertation, which will use deep reinforcement learning, to train an agent for controlling the traffic flow mixed autonomy traffic scenarios.

1.3 Outline

The rest of this dissertation is structured as follows:

- The second chapter covers the theoretical background of the scientific area of deep reinforcement learning and artificial intelligence in general. It provides a detailed analysis of the mixed autonomy traffic scenario which is being studied and all the important software tools which have been used. Finally, it refers to related traffic management solutions which are incorporating traffic management and reinforcement learning approaches.

- The third chapter focuses on the analysis of the deep reinforcement learning implementation, covering everything from design phase, to experiments and evaluation, describing the problems that arose and the methods of dealing with them.
- The fourth and final chapter summarizes the conclusions of the implementation and the suggested possible future extensions, variations and improvements.

Background

The following sections describe the field of Artificial Intelligence, the software tools and related traffic management projects, that have been used in dealing of the traffic control for Connected Automated Vehicles with Reinforcement Learning. In addition, the mixed autonomy traffic scenario is described in detail with all its specifications.

2.1 Artificial Intelligence

Artificial intelligence (AI), which is also known as Machine Intelligence (MI), is the demonstrated intelligence by the machines. Unlike the Natural Intelligence (NI) displayed by humans and animals, Artificial Intelligence is the ability of computers, robots and digital machines in general, to mimic cognitive functions commonly associated with intelligent beings, such as learning and problem solving.

The branch of computer science, related to Artificial Intelligence, aims to study intelligent rational agents, which perceives and analyze its environment using predetermined rules, search algorithms or pattern recognizing machine learning models. Based on those analyses, these agents take actions and make decisions, which maximize theirs chance of successfully achieving specific goals [1].

2.1.1 Machine Learning

Machine Learning (ML) is a method of data analysis that automates analytical model building. It is seen as a subset of Artificial Intelligence, that provides to systems the ability to automatically learn and improve through gained experience and with minimal human intervention or assistance.

Machine Learning algorithms, built mathematical models based on sample data, which are being used as training data and can be observations, examples, instructions or direct experiences. The algorithms try to identify

patterns and make predictions in the future, based on the provided data [2].

Machine Learning techniques are generally categorized in:

- Supervised Learning which is the task of learning from tagged data and its goal is to generalize.
- Unsupervised Learning which describes the learning from unlabeled data and its goal is to compress.
- Reinforcement Learning, the learning through trial and error and its goal is to act.

2.1.2 Deep Reinforcement Learning

Deep Reinforcement Learning is the combination of Reinforcement Learning and Deep Learning.

Deep Learning (DL) is essentially an autonomous, self-teaching system, that imitates the workings of the human brain in processing data and creating patterns for use in decision and prediction making on new data [3]. Deep learning known architectures such as deep, recurrent and convolutional neural networks, have been applied in many fields (computer vision, speech recognition, image analysis and etc), producing results comparable to and in some cases surpassing human expert performance [4].

Deep learning constitutes a branch of Machine Learning, representing the Unsupervised Learning category as it is capable of learning unsupervised from data that is unstructured or unlabeled. Deep learning algorithms using artificial neural networks which mimic the network of neurons in human brain, can perform various cycles to narrow down patterns and improve the predictions with each cycle.

Reinforcement Learning (RL) is a sub-field of Machine Learning concerned with decision making and studies how an agent ought to take actions in an uncertain environment, having as goal to maximize the notion of cumulative reward [5].

Reinforcement learning is an autonomous, self-teaching system which differs from supervised learning in not needing labelled input and output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge), as essentially learns by trial and error in order to achieve the best outcomes [6].

The agent learns to achieve a goal in an uncertain, potentially complex environment, typically stated in the form of a discrete-time stochastic control

process, known as Markov Decision Process (MDP) [7]. Markov Decision Process provides a mathematical framework, which is suitable for modeling decision making in situations of partly random and partly under control outcomes. The agent's action selection is modeled as a map which is called policy. This policy map gives the probability of taking specific action in a specific state [8].

Although the designer sets the reward policy, the model has no hints or suggestions for how to achieve the highest reward. It tries to figure out how to perform the task by employing trial and error so as to come up with a solution. During the process the models gets either rewards or penalties for the actions it performs [9].

Deep Learning and Reinforcement Learning are both systems that can learn autonomously. However the have one significant difference in the way each system is learning. Deep Learning is learning from a defined data set and apply its knowledge in a new data set, whereas Reinforcement Learning is learning dynamically by adjusting the actions based in continuous feedback with only goal to maximize a reward.

Despite this difference, it is possible to used Deep Learning in Reinforcement Learning systems.

Deep Reinforcement Learning is the combination of Reinforcement Learning and Deep Learning. It is being able to solve a wide range of complex decision-making tasks which widely exist in real-world problems. Applications of Deep Reinforcement Learning can be found in many domains such as healthcare, robotics, autonomous driving, smart grids, finance, traffic management and more [10].

2.2 Software Tools

An detailed overview of the software tools that have been used during this research is going to be presented in the following subsections, divided in the simulation and machine learning parts.

2.2.1 SUMO Simulator

Simulation of Urban MObility (SUMO) is an open source, highly portable, microscopic and continuous multi-modal traffic simulation package designed and mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center [11].

SUMO can be used for investigating numerous traffic management topics, with more related to this research, the simulation of the traffic effects of

the use of autonomous and conventional human-driven vehicles in the same transportation networks.

All the tools for handling tasks such as route finding, visualization, network import, emission calculation and APIs for remote simulation control, are available with SUMO. One of the available APIs is the TraCI, a short term for Traffic Control Interface. TraCI API allows to retrieve values of simulated objects (vehicles in that case) and to manipulate their behavior during the simulation. The pure python version of the TraCI API has been used in this project, for communicating with SUMO.

2.2.1.1 Scenario Structure

The mixed autonomy traffic scenario used in this thesis, presents a traffic situation that the humanity will face in the coming years, where a mix of autonomous and conventional vehicles exist in the transportation networks. The following paragraphs are going to provide a detailed overview of the important parts which compose the scenario.

2.2.1.1.1 Highway Specifications

There is one way highway of 5 km long, it has 2 lanes and it is divided in Autonomous Driving (AD) and Conventional Driving (CD) zones. There is one Road Side Unit (RSU) at specific location in the highway, which is the agent of action in this scenario. It is represented with a icon similar to Wi-Fi netowork and it's role is to notify automated vehicles to change to human driving mode before the end of the AD zone. The length of the Autonomous Driving (AD) Zone is 2.5 km and it is located in the first part of the highway. At the beginning of the AD zone and in each lane there is an *inductionLoop* detector, to detect vehicles entering the AD zone and a *laneAreaDetector* to get the current density in the AD zone.
All these details are configured through various xml files that are supported by SUMO simulator.

A SUMO screenshot of the highway is presented in figure 2.1.

2.2.1.1.2 Vehicle Specifications

The definition of Vehicles instances, Vehicle Types, Routes and Flows (Repeated Vehicles) is happening also through the xml files [12]. When a vehicle initialized in SUMO, it needs some additional information like the Vehicle Type, which describes the vehicle's physical properties and the Route that the vehicle shall take. The defined vehicle types are:

Figure 2.1: A representation of the highway.

- Legacy Vehicle (LV): Vehicles with only human driving mode.
- Connected Vehicles (CV): Vehicles which can communicate bidirectionally with the Road Side Unit and other vehicles.
- Connected Automated Vehicles (CAV): Autonomous Vehicles which can communicate bidirectionally with the Road Side Unit and other vehicles.

In the mixed autonomy scenario of this thesis, there is not specific route for every vehicle, but Flows have been used, so as to define repeated vehicle emissions for the available vehicle types. Three flows have been initialized, one for every type of vehicle that exist in the scenario. The vehicles in each flow use the vehicle parameters of the type that they belong to, except for the departure time which is unique for every vehicle.

By defining the vehicles according to the above way, resulted in the occurrence of randomness about the time of vehicle's insertion to simulation and the preferred vehicle type. The random number generator (RNG) algorithm that SUMO uses for these cases, is the Mersenne Twister [13]. The generation of the uniform pseudorandom numbers is happening based on either a specific seed value or the system time at which the simulation is happening. In this project the seed value has been used with different way during the training of the models and on the evaluation process of them and will be analyzed in the subsections 3.1.4 and 3.1.5.

2.2.1.1.3 Transition of Control (ToC)

The vehicles of the CV and CAV distributions are equipped with the ToC device [14]. This is a SUMO tool that provides facilities to model a transition of control (ToC) in automated vehicles. The transition of control can be either a upward transition (from driver to automation mode) or a downward transition (from automation to driver mode).

In the scenario, only the downward transition has been modeled. This is a more complex process, as the driver's condition right after the take-over event may allow only a reduced driving performance.

This driving transition is divided in two phases by the ToC Device:

- Preparing: The vehicle received a take-over request (ToR), but the control of the vehicle has not still taken back by the driver.
- Recovering: The transition of control (ToC) has been completed and there is a decreased driving performance by the driver.

Furthermore, in case that the driver fails to take back the vehicle control within a specified lead time, the ToC Device provides a a minimum risk maneuver (MRM).

All the above are illustrated in the provided by SUMO documentation figure 2.2.

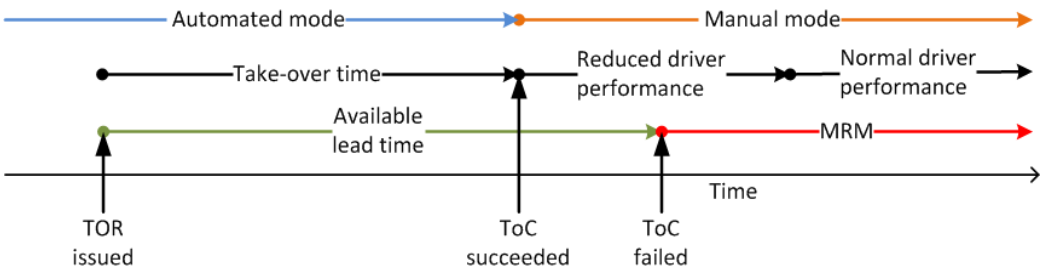

Figure 2.2: ToC process as it modelled by the ToC Device.

2.2.2 Machine Learning Libraries and Frameworks

An short introduction of the machine learning framework and the reinforcement learning libraries is going to be introduced below.

2.2.2.1 TesnorfLow

TensorFlow is an open source platform for machine learning applications, developed by the Google Brain team [15]. It provides a great variety of tools and resources to developers, giving the opportunity to build and train Machine Learning models through the Keras API [16] and visualize and monitor them through the Tensorboard tool.

Tensorflow was essential in this project as it constitutes the base for the reinforcement learning libraries that have been used and are presented in 2.2.2.3 and 2.2.2.4.

2.2.2.2 OpenAI Gym

Gym [17] is a very useful toolkit, provided by an AI research and deployment company, knows as OpenAI. Gym can be used for developing and comparing reinforcement learning algorithms, while it makes no assumptions about the structure of the agent. It is implemented in Python and is fully compatible with TensorFlow. Gym has a big collection of built-in environments, which have a shared interface, allowing the developer to write general reinforcement learning algorithms.

Every Gym environment has some standard specifications. It has an action space (what the agent can do) and an observation space (what the agent can see). It includes 4 important functions, which are the follow:

- Init function : Defines action and observation space and initialize the environment.
- Step function: Executes one time step within the environment.
- Reset function: Resets the state of the environment to an initial state
- Render function: Renders the environment to the screen.

In this project, a custom environment has been developed, following the guidelines and satisfying the requirements that Gym environments have.

2.2.2.3 Stable Baselines

Stable Baselines is a collection of reinforcement learning algorithms [18], which make it easier to build projects without being buried in implementation details. It is an improved version of the OpenAI Baselines [19] collection, in terms of customization, control and variety of the algorithms. Stable baselines supports also the logging of training information through Tensorboard tool, a very crusial to tool for evaluation of the trained models.

2.2.2.4 Ray - Rllib

Ray [20] is a framework for building and running distributed applications, on which there are many libraries for solving problems in machine learning. The useful tool for this thesis is the Rllib [21], a Scalable Reinforcement Learning library. RLlib is an open-source reinforcement learning library, which offers both high scalability and native support for TensorFlow and OpenAI Gym environments. Rllib can be used for experimentation with Multi Agent Reinforcement Learning and for this reason has been used over Stable Baselines, which only supports Single Agent approaches.

2.2.2.5 FLOW

FLOW is a deep reinforcement learning framework for mixed autonomy traffic experimentation, created and actively developed by members of the Mobile Sensing Lab at UC Berkeley [22].

FLOW can be introduced also as a traffic control bench marking framework, as it includes predefined traffic control scenarios, tools for not only creating new scenarios but also integrate them with deep reinforcement learning algorithms. FLOW framework is fully compatible with SUMO.

2.3 Traffic Managment

Traffic Management (TM) is the combination of measures and actions for organization, direction and regulation of all stationary and moving traffic in a road transportation system. Its purpose is not only to ensure that the movement of people and goods is secure, reliable and effective, but also to protect and improve the quality of the transportation facilities [23].

Automated Vehicles can drive properly in different traffic conditions, but there are some cases where the human driver should take back the control of the vehicle. These can be either unexpected events or situations where the vehicles reach the limits of their functional system. The transition of control (ToC) that Automated Vehicles perform, in order to allow the driver to take control of the key driving activities, can affect the whole traffic flow in a negative way, in terms of efficiency and safety. That creates a crucial need for novel traffic management approaches [24].

2.3.1 TransAID

TransAID project tries to deal with situations similar to those mentioned above. More specifically, Traffic Management approaches, including procedures and protocols, have been developed in order to ensure a smooth coexistence of automated, connected, and conventional vehicles, in cases where transition of control (ToC) in Automated Vehicles is needed [25].

The mixed autonomy scenario that has been used in this project, is a scenario created and investigated in TrainsAID. Their demonstrated Traffic Management solution, manages to distribute the take-over requests (ToRs) to Connected and Automated Vehicles (CAVs), in an efficient and safe way, before the end of the automated driving zone (AD zone). More precisely, they try to group C(A)V vehicles in one line and after that they send the ToR messages from the last vehicle in the group and after an approximate

period they move on and send the ToR message to the vehicle in front.

For example, there is a case where the algorithm managed to spot three C(A)V vehicles in a row in one of the lanes, without any conventional vehicle between them. Then, the algorithm will send the first ToR message to the last (the one closer to the start of the highway) of these three vehicles and only to that. The period between two ToR messages depends on a calculated time interval value. This value takes on account the speed of vehicle and the needed free space around it for performing the transition to human driving time and annotates the required time between two successive TOR messages. When this time has passed the algorithm will send the ToR message to second vehicle. The same procedure will be followed for the first vehicle also.

This thesis work, tries to distribute the ToR messages without disrupting the traffic flow and trying to delay as possible the change of the driving mode from automate to conventional in CAV vehicles by applying Reinforcement Learning algorithms, aiming on better results than TransAID.

2.3.2 Reinforcement Learning with SUMO

This dissertation is not the only project which tries to use Reinforcement Learning algorithms, for Traffic Management scenarios in SUMO simulator. There are already available projects which has a goal to optimize the traffic signal control, by setting the correct traffic light phase at specific intersections to maximize traffic efficiency. There are targeting either a single traffic light using single agent approaches or many traffic lights in the same grid using multi agent approaches.

However, a project that is dealing mixed autonomy traffic scenario similar to the one used in this thesis has not been found. Only the FLOW framework, which has been presented in 3.1.1, examines various traffic management scenarios which are under the same principles of mixed autonomy situations and can be consider as highly related work.

The above reasons boosted further the personal interest of the author, on developing an innovative and effective solution for complex mixed autonomy situations which would exist in future urban areas.

Deep Reinforcement Learning Framework

This chapter contains all the crucial information about the proposed reinforcement learning framework that has been developed. It is covering important components of all the phases, from the design to implementation and it going to provide all the details that are important for the evaluation of the system.

3.1 Design

During the design phase, the already available options of deep reinforcement learning frameworks were examined. The first step was to search for available open source deep reinforcement learning framework which will support the SUMO simulator. The thorough investigation resulted in FLOW, which is the special kind of framework that was needed and it is especially designed for mixed autonomy traffic research. As it was supporting custom SUMO scenarios and various deep reinforcement learning agents for training and deployment, it seemed the best decision for starting developing various approaches for optimization of the traffic management in the defined scenario.

After a month of efforts in order to incorporate FLOW with the predefined SUMO scenario, it was decided to drop it out in favor of a custom reinforcement learning framework, which will be analyzed later in this chapter. The main reasons for the rejection of FLOW framework, are analyzed in the subsection 3.1.1.

3.1.1 The FLOW case

For avoiding any misunderstandings, it needs to be said that FLOW is an amazing and very well designed tool. However, its limitations on importing complex SUMO scenarios, as the one of this thesis, and the need for further

customization on deep reinforcement learning agents, created some issues in the development of the proposed solution.

Initially, the predefined SUMO scenario of this research, as already mentioned in 2.2.1.1.2 uses Flows for every type of vehicle distribution. Every vehicle in a distribution has parameters (acceleration, tau, deceleration, etc) that may be different from vehicle to vehicle in the same distribution.

In case of FLOW framework, all these parameters are nicely packed to a class known as *SumoCarFollowingParams*. However, the perception of FLOW framework in connection between flows and vehicle distribution were different that the one used in the scenario. Practically, FLOW will create the flows for every distribution by using the *Vehicle* class, which creates many vehicle instances, with same values of the *SumoCarFollowingParams* class for every instance. This implementation was unable to handle a scenario where practically every vehicle in the simulation will have unique parameters of the *SumoCarFollowingParams* class.

At first, there was the development of a custom python script for reading vehicles parameters as the default code could not read and parse values of the scenario's xml files. Secondly, the *Vehicle* class modified and customized to a more advance version of it. The goal of changing original connection of one *SumoCarFollowingParams* class instance per *Vehicle* class, to one *SumoCarFollowingParams* class instance per *Vehicle* class instance has been successfully achieved.

However, more issues came up. After the localization of the main reasons of conflicts with the new class, a new finding appeared. Some changes in core code of the communication and exchange of information between the flow framework and TraCI API was needed to be done.

The changes that have been introduced, activated a series of sequential errors and problems in how the communication between the TraCI API and FLOW framework was happening. The problem was mainly on updating the positions, speed and values related to vehicles from TraCI API, inserting new vehicles to flow types, updating and managing the *Vehicle* classes.

After a lot of time with trial and error approaches, there was not any successful progress to these important problems. FLOW framework could not support the more sophisticated and more detailed vehicle type distributions that the scenario introduced. That was the final sign that has been taken into account for changing the approach and creating a custom made framework in order to train and test the future models.

3.1.2 The Custom Framework

The custom developed framework is mainly divided in the reinforcement learning part and the SUMO communication part.

Traci Manager

As far as the SUMO communication part is concerned, one main class has been developed. The *Traci Manager* class is responsible for doing all the communication with SUMO through the TraCI API, updating vehicles information, sending ToC requests, calculating average, max, min and percentage and all the kind of values that will be needed during the training or the simulation processes.

Some of the fundamental tasks that this class will have to do during each experiment are the below:

- Set up the TraCI API connection, so as to be able to exchange information with SUMO.
- Initialize and handle a Vehicle class in order to store and use properly to all the vehicle related values (position, speed id, etc).
- Send the actual ToC messages to SUMO ToC device as defined in 2.2.1.1.3 paragraph.
- Handle lists for Legacy Vehicles, Connected Vehicles and Connected Automated Vehicles in order to provide flexibility in various other function used during the experiments.

Apart from the above functionalities, the *Traci Manager* class provides numerous functions that will be helpful in terms of implementation and will be mentioned during the described in sections 3.2 and 3.3, applied approaches.

Gym Environment

For the reinforcement learning part, many software tools have been used. The environment was created according to the specification of a Gym environment[17]. In general, Gym environments are quite widespread for such kind of machine learning applications.

As it is already mentioned in 2.2.2.2, Gym includes a collection of test problems and environments, that give the possibility to work out the preferred reinforcement learning algorithms. However, in this thesis, there was no interest to use any of the available built-in environments of Gym. It was needed to create a custom environment.

The environment has of course an observation and an action space. The observation space is an environment-specific object, representing agent's observation of the environment. It was highly depended on the applied approach of each experiment, but for all of the cases, it was expressed as a matrix. On the other hand, the action space was a discrete list of non-negative integers, where the size was different from approach to approach. Both of these terms, will be defined and expressed analytically for every approach that will be presented in the sections 3.2 and 3.3.

The developed environment also consists of the following functions:

- Init function: It will take all the additional parameters regarding the simulation configuration (simulation steps, xml files locations, etc) and will initialize a the Gym environment.
- Reset function: It will reset to the start state the environment, with all its variables and the *Traci Manager* class instance.
- Compute Observation function: It will update the observation list by calling the needed *Traci Manager* functions (depending on current applied approach).
- Compute Rewards function: It will compute the rewards by calling the right reward function (depending on current applied approach).
- Step function: It will execute an environment step (one SUMO simulation step) and will apply the action in the environment, by calling the run function of the *Traci Manager* class. After that it will calculate observations and rewards using the suitable function and it will check if the termination state has been reached.
- Render function: It will set if the SUMO GUI will also run or not.
- Save function: It will to save csv file useful information of the experiment.

The implementation of the environment was in a way that will be independent from direct SUMO communication and value retrieval in order to be easily handled through various reinforcement learning customizations.

Deep Q Network (DQN)

The reinforcement learning libraries that have been used were the Stable Baselines (2.2.2.3) for the Single Agent and the Rllib (2.2.2.4) for the Multi Agent approach. Both of them are highly depended on Tensorflow framework and for this reason Tensorflow was chosen over other available options.

Being influenced by previous implementations on reinforcement learning implementations on SUMO, DQN was the first choice for this mixed autonomy traffic managment case. Nevertheless, a comparison between DQN and

other reinforcement learning algorithms, like A2C [26] and PPO [8], has been made on the first working version of Single Agent and Multi Agents approaches, getting results in favor of DQN.

The final selected reinforcement learning algorithm is the Deep Q Network (DQN), introduced by DeepMind in 2013 [27]. As far as it concerned the DQN and its most important supplements the Double DQN, the Dueling DQN and the DQN with Prioritized Experience Replay, a short presentation is introduced below:

- DQN: A reinforcement learning algorithm that combines Q-Learning with deep neural networks in order to make reinforcement learning to work for complex and high-dimensional environments (video, robotics and etc).
- Double DQN: The stock DQN algorithm was found to have a tendency of overestimating the action values under certain conditions, so Double DQN introduced in order change that [28].
- Dueling DQN: It uses a new neural network architecture for model-free reinforcement learning. The network is splitted into two parts, one learns to provide a value estimation at each time-step, and the other calculates the advantages of the actions. Both of them are combined for a single action-advantage Q function, generalizing the learning across actions without imposing any change to the underlying reinforcement learning algorithm [29].
- DQN with Prioritized Replay: It extends and prioritizes original DQN's experience replay, by replaying more frequently important transitions and more specifically memories where the real reward significantly diverges from the expected reward. It achieves an efficient learning as the agent is able to adjust itself in response to developing incorrect assumptions [30].

By default, both Stable Baselines and Rllib initialize the DQN class with double q learning, dueling and experience replay extensions enabled. These version have been used as base for each approach. However, they have been parametrized, trained and tested in various configurations.
The available options for configuring the DQN algorithm are numerous. The most important ones which have been manipulated in this thesis are presented below:

- policy: The policy function that DQN algorithm will use.
- gamma: A float discount factor, that states how important future rewards are to the current state. Discount factor is a value between 0 and 1.

- learning_rate: A float number, which determines to what extent newly acquired information overrides old information.
- buffer_size: The size of the replay buffer, where the experiences are saved at each step so as to train the neural network episodically.
- exploration_fraction: The float fraction number of entire training period over which the exploration rate is annealed.
- batch_size: The size of a batched sampled from replay buffer used for training.
- learning_starts: The number of simulation steps where the model used to collect transitions, before the learning process starts.
- target_network_update_freq. Value to define that the model updates the target network every target_network_update_freq steps.

Theses values have been tested in various combinations and compared to each other until the moment that the best combination had been achieved for the used SUMO scenario.

Hardware and Software

The hardware and software configurations, which have been used were the same for all the experiments in both single and multi agents approaches. The hardware consists of a Nvidia GK110GL Graphic Card, the Intel Xeon E5-2687W, a 4 core CPU and 32GB of RAM.

The software versions that have been used, were SUMO 1.6, python 3.7, Stable Baselines 2.10 and TensorFlow 1.14 for the Single Agent approach and Rllib (Ray) 0.8.6 and TensorFlow 2.2 for the Multi Agent approach.

3.1.3 Agent(s) and Traffic Management Indicators

The agent in the environment is the one Road Side Unit (RSU), having as one and only goal to send the take-over requests (ToR) to C(A)V vehicles, in order to notify them to change to full human driving mode before the end of the AD zone. This change to human driving mode, which is formally expressed as transition of control (ToC), is not happening instantly, but in a timeframe of 5-10 seconds, and this time period cannot be configured or controlled in any way.

The transitions to human driving control should happen without disrupting the traffic flow and produce congestion incidents inside the AD zone. Furthermore, the agent should distribute the ToR messages in a way that the C(A)V vehicles will cover as much as possible distance during their autonomous mode before the transition will happen.

The indicators that will help in understanding the efficiency of the agent(s) in terms of an efficient traffic management are provided and calculated by SUMO (except one) and are the following:

- TimeLoss (Seconds): The time the vehicle lost due to driving below the ideal speed.
- Duration (Simulation seconds): The time the vehicle needed to accomplish the route.
- DepartDelay (Simulation seconds): The time the vehicle had to wait before it could start his route.
- WaitingTime (Seconds): The time in which the vehicle speed was below 0.1 m/s.
- WaitingCount (Number): The number of waiting time incidents during the simulation.
- MeanSpeed (Meters/Second): The mean speed of the vehicles in a specific lane.
- TravelTime (Seconds): The estimated average travel time of vehicles in the given lane.
- AvgCavDistance (Meters): A custom developed indicator, which represents the average covered distance of the C(A)Vs] vehicles until the moment they received the ToR message. It is calculated by the developed python code.

3.1.4 Training

In order to automate as much as possible the processes during this research, the training of the models was scheduled trough a separate python file.

Initially, all the essential machine learning libraries were inserted (TensorFlow, OpenAI Gym and either Stable Baselines or Rllib, depending on Single or Multi Agent Approach). After that, the needed initializations about the environment and training procedures were happening, according to the arguments that the user could pass to the python file.

These arguments, are presented in the following list:

- trains: The preferable number of trainings for the model.
- sim_steps: The number of simulation steps that was wanted to be investigated. The number should be smaller or equal to the maximum allowed simulations steps for the scenario (48335).
- name: The custom name what should be given to the trained model, after a successful training.

In case that no values were provided to the above arguments at the beginning of the training procedure, there were some default values assigned. That were 30 for trains, 48335 for sim_steps and dqn_sample for the name of the model. The training for each model was done on random seed numbers for the SUMO simulations, always between the values 1025 and 2035.

To keep a record during the many training rounds, the different selected parameters that have been used for each model set up and the useful information regarding the selected reward function, has been documented in tables. In these tables, the columns were representing the configurations and model parameters, and the rows the selections in each test. A detailed example of the training record tables, that have been produced during this project, is presented in the figure 3.1.

Training Record Table

tests	Reward Functions Options: RF1 RF2 RF3	learning _rate	buffer_ size	batch _size	exploration _fraction	learning _starts	target_network _update_freq	trains	sim_steps
test1	DQN-RF3	0.005	300000	32	0.1	1000	100	40	48335
test2	DQN-RF3	0.001	300000	32	0.1	1000	100	40	48335
test3	DQN-RF2	0.005	50000	32	0.1	1000	100	20	10000
test4	DQN-RF2	0.001	50000	32	0.1	1000	100	20	10000
test5	DQN-RF2	0.005	50000	64	0.1	1000	100	20	10000
test6	DQN-RF2	0.005	50000	32	0.2	1000	100	20	10000
test7	DQN-RF2	0.005	50000	32	0.1	1000	500	20	10000
test8	DQN-RF1	0.005	50000	32	0.1	1000	100	20	48335
test9	DQN-RF1	0.005	100000	32	0.2	1000	500	30	48335

Figure 3.1: Screenshot of one used Training Record table.

After the first training round the ideal combination of these parameters have been found. The final used ones for the DQN algorithm in both libraries are the following:

- learning_rate: 0.005
- buffer_size: 300000
- batch_size: 32
- exploration_fraction: 0.1
- learning_starts: 1000

- target_network_update_freq:100
- trains: 40
- sim_steps: 48335

Moreover, the DQN versions of Stable Baselines and Rllib libraries, support the extraction of information during the training process in a format suitable for visualization through the Tensorboard tool. An ability that has been used and enabled so as to visualize and constantly evaluate the trained models.

Finally, after each training round, the developed system was saving the model's current version. This approach has been followed in order to be able to use the model at its best version. In other words, to get the inside knowledge after which training round, the training process should be finished, as it is not beneficial anymore for the model.

3.1.5 Simulations

Because of the high randomness that characterizes the nature of the used SUMO scenario, one generated simulation for the trained models was not a sufficient metric. For this reason has been decided to run 10 simulations for each trained model, and use values from these simulations to evaluate them. The seed numbers for each simulation, were between 1024 and 1034, defined in sequence for the simulation rounds.

During the simulations of the models, the system was updating at each time step the indicators of *TravelTime*, *MeanSpeed* and *AvgCavDistance* and saving the values in a csv file. This file was very crucial for the evaluation of the models. Furthermore, the tripinfo.xml has been used, which contained the values for the rest indicators and was created at the end of each simulation by SUMO.

3.1.6 Baseline Example

For a proper evaluation of the simulations results, a comparison with the baseline model was necessary. The baseline model was actually an already implemented and validated traffic management approach that has been developed through the transAID project. It was kindly provided by a member of TransAID team, Mr Robert Alms, for research purposes.

The process to get the simulation results, was the same that already mention in the section 3.1.5. The baseline model should be executed for 10 times in a row, store the needed values after each run, and report in terminal the success of each experiment.

However, before that process, some modification were necessary as the original code of the transAID traffic management python file, was written in python 2, whereas the proposed system was developed in python 3.

Luckily, apart from some modifications in the way of handling the dictionaries and list values, nothing else changed in the core code of the script. After the transition of the TransAID traffic management file to python 3, the additional functions for running the file in repeat, storing the values in the suitable csv files and exporting them to tripinfo xml files, were added successfully.

Finally, the required values for the comparison with the other models had been obtained and was time for the further development of the reinforcement learning approaches.

3.1.7 Evaluation

The evaluation of the simulation results for each model can be divided in two main steps.
The first step is the study of the whole training procedure with the Tensorboard tool. By having some eye friendly visualizations through the produced graphs for the evolution of rewards during the continues training phases, the accuracy of the model after the numerous iterations and the training efficiency, were finally easier to be monitored. This evaluation method made easier the understanding of which changes in DQN algorithm parameters cause specific changes to model behaviour and which parameters should be investigated further for achieving a better model configuration. After all, this evaluation method could not provide any real value about the improvement or not of the model as far as the traffic indicator is concerned.

The second step of evaluation was to create a new python file, which will collect for the necessary models, the traffic indicator values from the generated csv and tripinfo files for each of the ten simulations that have been performed and will store them in appropriate lists. After that, it will create box plots for all the traffic indicators, except the WaitingTime and the WaitingCount indicators, as the barplots have been more useful for these two. The boxplots made easier the process of presenting the distribution of the values and targeting the outliers of the ten simulations. Whereas, the barplots provided a more complete overview of the total waiting - congestion time and the number of their incidents, as the goal was to eliminate as much as possible these two values.

The first phase has been used to figure out how many training rounds are really beneficial for the models and which DQN parameters should be changed, so as to achieve the best performance in terms of higher reward.

The second phase has been used to evaluate each strategy of the single and multi agent approaches between each other.

The first comparison regarding the traffic indicators for each model was with the baseline example, which has been mention in 3.1.6. That was the first sign of the performance for the developed models.

However, in order to be able to understand if the models are really doing something efficiently because there were designed to do it and not by accident a bad model was trained and performed the simulations. These model was designed train the RSU agent to to send ToR messages in a way that it will increase the average covered distance by C(A)Vs, but do not care at all about traffic indicators. After validating its bad behaviour, there was a green line for further development.

3.2 Single Agent

An analytical overview of the strategies that had been implemented in the Single Agent approach for optimizing the ToR distribution in the scenario will be presented in the following sections.

The idea behind each strategy, the optimization goal and implementation details from the design phase to real coding information will be documented in order to cover in best way the effort made during this dissertation. At the end an analysis of the results in comparison with the baseline will be documented.

3.2.1 First Strategy

The first strategy that has been developed for achieving a more efficient ToR distribution for the autonomous connected vehicles, than the TransAID traffic management solution, was by far the most simplistic one in terms of reward functions design and invested time for the training and evaluation of the agent. It has been used mainly for the understanding of the traffic indicators importance for the manipulation of the RSU agent, an initial exploration of which parameters could be tweaked for the DQN algorithm and to record the training and evaluation duration of the developed framework. Furthermore, it was the first deep introduction in the mixed autonomy scenario, its purposes and conditions. The gained knowledge was an essential driver not only for the design decisions of future reward functions and strategies, but also for being aware of the things that should be avoid during these designs.

A visual representation of the Road Side Unit agent with the one-way highway for the first strategy is the figure 3.2.

Figure 3.2: Scenario representation for strategy one.

The road has been modeled based on the two lanes that already existed in the scenario. In this way the RSU agent has one action for sending ToR messages to C(A)Vs for each lane and one action for not sending any ToR messages. For convenience, the actions mapped as follows:

- action 0: RSU agent does not send any ToR messages.
- action 1: RSU agent sends ToR messages to all the C(A)V vehicles of the bottom lane.
- action 2: RSU agent sends ToR messages to all the C(A)V vehicles of the top lane.

The observations that the RSU agent has about the environment is going to play a crucial role on what it can understand at each time step. The observation space in this experiment contains the MeanSpeed and TravelTime values for both lanes, which are the traffic indicators provided by SUMO through the TraCI API. Moreover, density of vehicles in each lane, the numbers of Legacy Vehicles, Connected Vehicles, Connected Automated Vehicles and the vehicles which has received the ToR messages for each lane, it is also known to the agent via the observation space. These values are not directly provided by SUMO, but have been calculated based on information that the system could get during the simulation, using the TraCI API, with the help of TraciManager class.

The developed reward functions were aiming on different target each time. The first reward function, was trying to decrease the total Travel Time indicator for both lanes. Similar logic was also behind the second reward function, for making the RSU agent to send ToR messaged in a way that the total Mean Speed for the two lanes will be increased. By these two reward functions, it was believed that the agent will act in such way that the traffic flow of the scenario will not be disrupted.

However, when the trained models were evaluated, they showed the same behaviour. They produced significant congestion incidents and periods, disrupting completely the traffic flow, despite the fact that they were designed

to act differently and achieve their goals of decreasing the total Travel Time and increasing the total Mean Speed for the two lanes of the highway.

To deal with congestion episodes during the simulation, two more reward functions have been developed. The first one had the goal of sending the ToR messages to the C(A)V vehicles of the lane with the smaller vehicles density. The second one was training the agent to send as few ToR messages as possible to C(A)Vs at each time step. In other words, the agent should choose the lane with the smaller number of available C(A)Vs in it.

At first glance it seemed that the two models which were using the last two reward functions, were dealing in slightly better way with congestion incidents. Nevertheless, a more thorough evaluation of the models proved that the RSU agent was sending the ToR messages the moment that every vehicle was entering its lane. That has a result the indicator of *AvgCavDistance* to be close to zero, as practically the moment of appearance of C(A)V vehicles in autonomous driving zone, the RSU agent has already send the take-over request messages to them.

Sadly, the trained models did not perform in the expected way. There were many congestion episodes with long congestion duration and the *AvgCavDistance* indicator was always close to zero. These results led to some useful realizations. Initially the agent should have more control on the area of the highway. This could be achieved by giving the ability to RSU agent to control smaller lane areas during the simulation or by controlling directly the C(A)V vehicles. Additionally the reward functions should be more sophisticated and take in account more details from the simulation environment. These factors established the suitable path for the consideration of a cells approach.

3.2.2 Second Strategy

Before going deeper to the second strategy of the single agent part, more details should be written down for the cells approach which was mentioned earlier, as it is a very crucial part for all the following strategies.

Cells Idea

The scenario used in this project is very dynamic and is characterized by randomness in terms of the total number of inserted vehicles and their type. From the findings of the first strategy in section 3.2.1, turned out that the agent need to have better control over the whole highway, which means in terms of reinforcement learning, more actions for the agent.

The first idea was to model every sent ToR message by the RSU agent to a C(A)V vehicle, as one action for the agent. But that was impossible to be implemented in real world. That's because the total number of C(A)V vehicles could not be estimated in advance and the fact that at each time step the number of C(A)V vehicles in the simulation is not the same as it changes all the time. Even without these two main problems, the approach of connecting agent's actions with single ToR messages could lead to chaos as the number of these ToR messages would be big enough to increase highly the complexity of the problem.

After these considerations, the finding of an alternative method which will provide more actions to the agent, was considered as vital for the future steps. The cells idea was the solution to this problem.

The autonomous driving zone of the highway will be divided in cells, which will be practically smaller parts of a lane. The number of cells will be the equal for both lanes, and their size will be approximately the same. Each cell will represent an action for the agent. In other words, when an agent will choose a cell, it will be able to send ToR messages to all the C(A)V vehicles of this cell at the specific time step and only to this cell. In this way, would be possible to model the environment in a more suitable way by giving more actions and more detailed observations to the DQN agent.

In the second strategy, the cells ideas was followed and applied. The highway was divided in ten cells, five for each lane. The way of how the RSU agent perceived the environment after this change is visualized in the figure 3.3 .

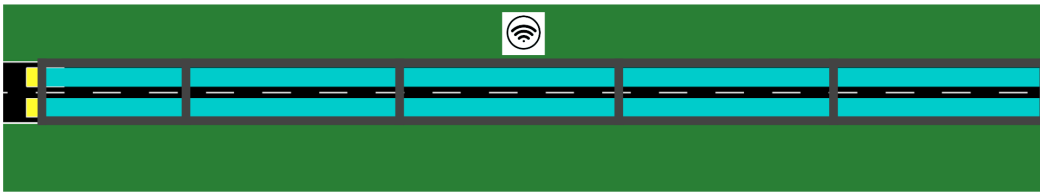

Figure 3.3: Scenario representation for strategy two.

The first cell in each lane has a length of 350 meters, the last cell 500 meters, and the rest 1650 meters of the 2.5 km lane, was divided in 3 equal parts. The first cell has the smaller length (350 meters) because it was preferable to avoid sending ToR messages at the beginning of the highway as one of the goals in this project is to keep the autonomous mode of the C(A)Vs for a distance as long as possible. The three cells in the middle, have

the same length of 550 meters and it was expecting the majority of ToR messages to be distributed in these three cells. The fifth and last cell in both lanes is 500 meters long which is a sufficient distance for a C(A)V vehicle to perform the transition of control before entering the non autonomous zone. The reason behind this design decision can be found in section 2.2.1.1.3, where it was mentioned that the procedure of changing from autonomous to human driving mode would be between five and ten seconds and cannot be configured. The last cell operates as the safe gap before the end of the AD zone. That means that the agent will select this cell in case that there are C(A)Vs which did not receive ToR messages in the previous cells, but by entering the last cell should do the transition human driving mode anyways.

By modelling the scenario with ten cells, the RSU agent has one action for sending ToR messages to C(A)Vs for every cell in each lane and one action for not sending any ToR messages. For convenience, the number of the cells starts from left to right and the actions have been mapped as follows:

- action 0: RSU agent does not send any ToR messages.
- action 1,3,5,7,9: RSU agent sends ToR messages to all the C(A)V vehicles in the corresponding cell of the bottom lane.
- action 2,4,6,8,10: RSU agent sends ToR messages to all the C(A)V vehicles in the corresponding cell of the top lane.

The observation space for the agent was also influenced by the cells idea. For every cell there were values for the average speed of vehicles, the numbers of Legacy Vehicles, Connect (Automated) Vehicles and the vehicles which have received the ToR messages and were performing the transition of control. Every cell was expressed with these four values, which constitute the perception of the agent for the environment.

The reward function was more sophisticated than the developed ones in first strategy, trying to give more meaning to agent's actions. The goals of course were the same. Minimize the congestion episodes and the duration of them, and increase the covered distance in autonomous mode. There was experimentation on which observation values should be used in reward function and how to manipulate the agent in order to prefer the three middle cells for each lane and avoid the ones close to the edges of the highway. Based on the action of agent there positive or negative (penalty) reward.

The total reward value for the agent at each time step was the combination of three different values, the sub rewards. The first value was representing the appearance of a congestion episode. By making a function call to *TraciManager* class, there was the ability to get the information through the TraCI API for possible congestion episodes and their duration. If that was

the case, this value was -10 and used to punish the agent. Another punishment was established, for making the agent avoid the last (fifth) cell in each lane and was also a negative reward. Whenever the agent was sending ToR messages to the entered to last cell C(A)V vehicles, the second sub reward was equal to -1. The third value was getting a float number based on which cell the agent decided to activate (send ToR messages). It was a positive value and more specifically for the first cell 0.1, for the second 0.2, for the third 0.3, for the fourth 0.4 and for the fifth and last one 0. In this way it was tried to model the importance of every cell or the cell influence in other words for the ToR distribution. If the action was 0, which means that the agent decided not to act, this value does not take part in final reward consideration.

The final reward was calculated based on two main if cases. The first one was examining if the agent selected a cell or not. If the agent decided not to send ToR messages in any of the available cell, will receive a final reward of 0 only if the punishment for congestion is not triggered and remained 0. Otherwise the final reward will be equal to -10.

In case that the agent activated a cell, then there was one more complicated examination dependent on three cases. This will check if the vehicles which received the ToR message and are in the phase of transition of control are more than 3 or if there was a congestion episode or an activation of last cell in any of the lanes. If at least one of these cases was valid, the final reward will be sum of the negatives sub rewards for congestion and last cell option. But in case that none of the previous three terms was valid, the reward is the sum of the number 10 and the cell influence value.

By this strategy the ToR distribution was improved in both terms of traffic flow stability and covered distance in autonomous mode for the vehicles. The reward function, the increased number of actions and the more detailed observation space, helped the reinforcement learning agent to get a deeper understanding of the problem and produce the first significant results until that moment.

However, the cell idea was giving enough space for experimentation around the number of created cells in the highway, which will lead to more actions for the agent. By combining it with the fact that there was continues need for better results in traffic efficiency and bigger covered distance in autonomous driving, there was the result of a mixture which was the trigger for the design of the third strategy.

3.2.3 Third Strategy

A high cell number could give a better control for the agent during the simulation, but could also increase the complexity on the training phase and affect the performance of the RSU agent in ToR distribution in a negative way. For this reason even numbers between 10 and 20 were investigated, leading to a final decision of increasing the cell number to 14, with each lane to have 7 cells. The way of how the RSU agent perceived the environment after this change is visualized in the figure 3.4 .

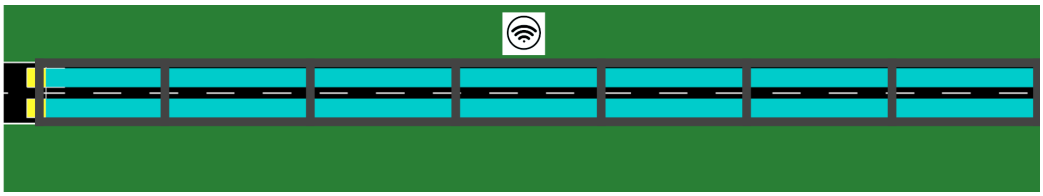

Figure 3.4: Scenario representation for strategy three.

By increasing the cell number, automatically the actions for the agent are getting equal to 14 plus the one action for not sending ToR messages. The correlation between actions and cells is under the same principles with second strategy and can be described as follows:

- action 0: RSU agent does not send any ToR messages.
- action 1,3,5,7,9,11,13: RSU agent sends ToR messages to all the C(A)V vehicles in the corresponding cell of the bottom lane.
- action 2,4,6,8,10,12,14: RSU agent sends ToR messages to all the C(A)V vehicles in the corresponding cell of the top lane.

The first cell in each lane has a length of 300 meters, the last cell 500 meters, and the rest 1700 meters of the 2.5 km lane, was divided in 4 equal parts of 350 meters and one cell (the penultimate one) of 300 meters. The same logic with the previous strategy has been followed in the cells division. The RSU should avoid to send many ToR messages in first cell of each lane, the last cell should operate as a safe gap before the end of autonomous driving zone and the majority of the ToR messages should be distributed in the 5 middle cells.

The observation space for the agent was exactly the same with the strategy 2, so the agent was extracting information for the environment through the same four values for each cell. The average speed of all vehicles in cell, the numbers of Legacy Vehicles, Connect (Automated) Vehicles and finally

vehicles had received the ToR messages and were performing the transition of control.

Apart from the increase on the cells number, there was also desire to explore further the restriction and possible elimination of congestion episodes by developing reward functions which will examine also the connection between the average cell speed and an upcoming congestion episode. That was because the appearance of congestion episode during the simulation is not a result of the last action, but a sequence of numerous actions may lead huge drops on vehicles speed values and finally long or short congestion incidents in simulation.

As it is already mentioned in 2.2.1.1.3, after the transition from to autonomous to human driving mode, the appearance of reduced driving performance on the vehicle, is quite possible. This event was visible in the simulation and was connecting with a decrease in vehicle's speed.

By studying carefully the simulation and focusing on the vehicles behaviour from the moment that they were receiving the ToR message and until the moment they will finish their transition of control, it was observed that many addressed ToR messages in one cell, can heavily affect the average vehicles speed in the immediately next cell in a negative way.

Furthermore, it was noted that when the average cell speed was reaching values under the 20 meters per second, then the appearance of congestion episode on this cell or the next one, was quite possible.

At the beginning, the same reward function with strategy 2 was used, but in a slightly modified version. The punishment for congestion episode increased to -100 but the punishment for sending a ToR message to last cell of the lane remained -1. The if statement for checking if the agents sent ToRs or didn't act was the same but the upper limit for the if statement for the allowed number of vehicles which received ToR message in the chosen cell and are in the phase of transition of control, reduced to 2 from 3. The new introduced punishment was targeting on low average cell speeds. If there are cells with average vehicles speed less than 20, then the punishment was enabled and the value was the number of the affected cells multiplied by 10. All the punishments were initialized to 0 and their values were changed only if they were activated.

The final reward was a combination of these values. Either the agent acted or not, in case of punishments with different values than 0, the final reward was the sum of the punishments. On the other hand, with all the punishments inactive and values of 0, the agent was receiving final reward of 0 if it didn't act or it was receiving the order number of the activated cell (1 for the first cell, 6 for the sixth cell in lane) as a reward.

However there was much more space for improvement. An effective way

to deal the congestion episodes in this scenario in more effective way than the previous strategies, was to incorporate to a new reward functions the additional information about the strong link between the activated cell and the in front of it in the same lane and the correlation of congestion with average speed values.

The reward function was redesigned from scratch. Firstly, the punishment for low average speed in the activated cell was enabled when the speed value was below 15, as it was observed that 20 was too high limit. The assigned negative value in that case was -1.

Secondly, the connection between two cells expressed through the sum of average cells speeds. For example, if the agent decided to send ToR messages to cell 3, the average speed of cell 5 will be also investigated. This punishment would be enabled and would take a value of -1, if the sum of the two average speeds will be below a defined limit. This limit after experimentation was set to 40 meters per second.

Thirdly, the punishment for sent ToR messages to last cell of a lane reduced to -10 and the punishment of the congestion incident appearance was increased to an extreme value of -10000.

Finally, it was decided to remove also the action of not acting and the agent had a number of actions equal to cells number at the end. The final reward would receive a value based on the priority of the punishments.

If there was congestion, the final reward would be -10000. In case of no congestion incident, but the agent chose the last cell of any lane, the reward would be -10. If none of the previous punishments have been activated, the agent would get a final reward of -1 if any of the two average speed punishments was activated. In the most successful scenario it would get a positive reward which will be the value of the cell order divided by 10 (0.1 for first cell, 0.6 for sixth cell).

The third developed strategy in the single agent approach was quite successful. It managed to achieve remarkable results with both of the developed reward functions, in terms of the traffic management indicators. That has a result, to leave at side the development of single agent strategies and focus the workflow on a multi agent approach, hoping to even better results. The following subsection will present and discuss further the results of trained models with the single agent approach.

3.2.4 Results Analysis

This section is going to provide a detailed overview of the results which have been produced with the single agent approach trained model and discuss the related to traffic indicators plots.

Table 3.1: Strategy mapping table with IDs (Single Agent).

Strategy	ID
TransAID	1
Strategy 2	2
Strategy 3a	3
Strategy 3b	4

Apart from the developed models of first strategy, which didn't produce comparable results, the evaluated models of second and third (two variations 3a and 3b) strategies will be compared with the baseline model, the TransAID traffic management solution. The mapping of strategies to ID models in the plots is presented in table 3.1 .

The comparisons will be on the three most representative traffic indicators, the *AvgCavDistance*, *TravelTime* and *WaitingCount*.

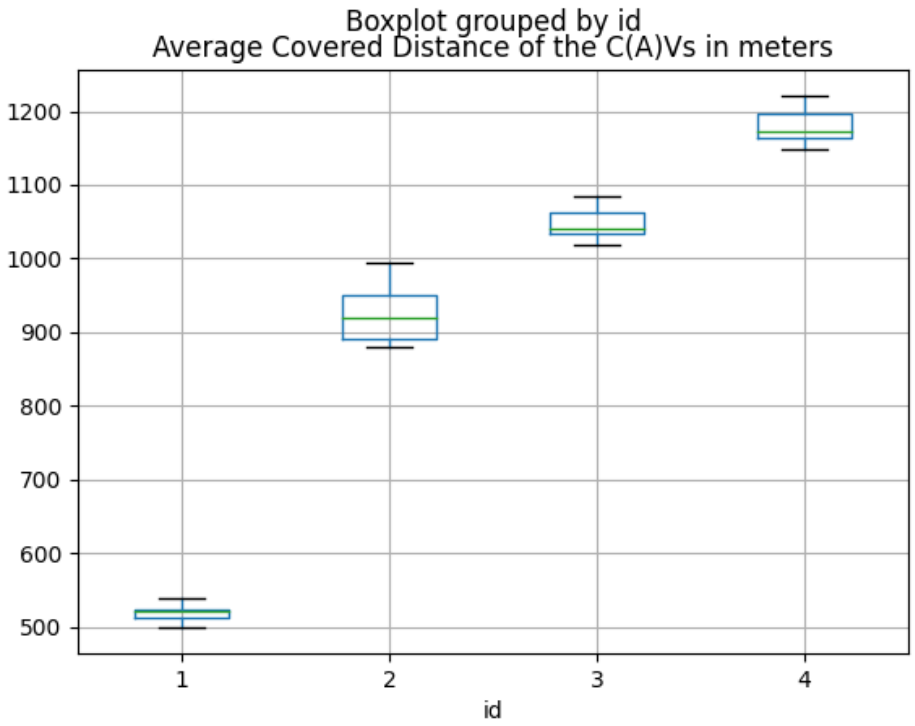

Figure 3.5: Boxplot for AvgCavDistance indicator for single agent approaches.

Regarding the *AvgCavDistance* indicator, the results are shown in boxplot 3.5.

The first model which is the TransAID traffic management solution, had the shortest average covered distance in autonomous mode for the C(A)V vehicles, of around 500 meters. The second model which belongs to strategy two, was the one introduced the cell idea and divided the highway in 10 cells (5 for each lane), managed to improve the indicator to almost a double value (between 900 and 1000 meters). The third strategy, with the fourteen cells (7 cells per lane) implementation, introduced the connection between two consecutive cells, as a sent ToR message in one cell could possibly be completed and contribute to congestion incident in the cell in front of it (left to right order). The first variation was the first solid implementation (model three) whereas the second variation was a more strict to above considerations implementation (model four). Both the variations produced results of over 1000 meters, around 1100 and 1200 meters in each case.

In was proved that deep reinforcement learning developed models, performed in a better way by keeping the automated mode of C(A)V vehicles for a longer distance than the baseline model of TransAID method, as it has been showed by the concrete boxpolts, with no outliers.

In order estimate the real value of the results, the boxplots for the *TravelTime* indicator in figure 3.6 should be studied also.

The TransAID model provided the baseline for an efficient traffic management and having a boxplot with no outliers and a median value of 323 seconds, for the average Travel Time of vehicles during the simulations.

Comparing the model of second strategy with the baseline, the slight improvement is visible, even with having outliers lower or not higher than the baseline.

For third strategy, the two variations produced different results. The model 3 of first variation produced a solid boxplot, with a median (321) less than baseline, where the model 4 of second variation has a boxplot with higher median (325) and 3 outliers, with two of them close to highest 330 seconds.

The results of the deep reinforcement learning models did not manage to improve clearly the *TravelTime* indicator and especially the fourth model which had negative impact. The investigation of *WaitingCount* indicator through the plot in figure 3.7, was playing a crucial role for understanding the traffic efficiency of the developed models.

For 10 simulations, the TransAID method (model 1) led to the occurrence of congestion episodes for 7 times in total. The congestion incidents for second strategy are 4 and for third strategy (both variations) 3. The bar plot is clear that the indicator *WaitingCount* has been improved by using

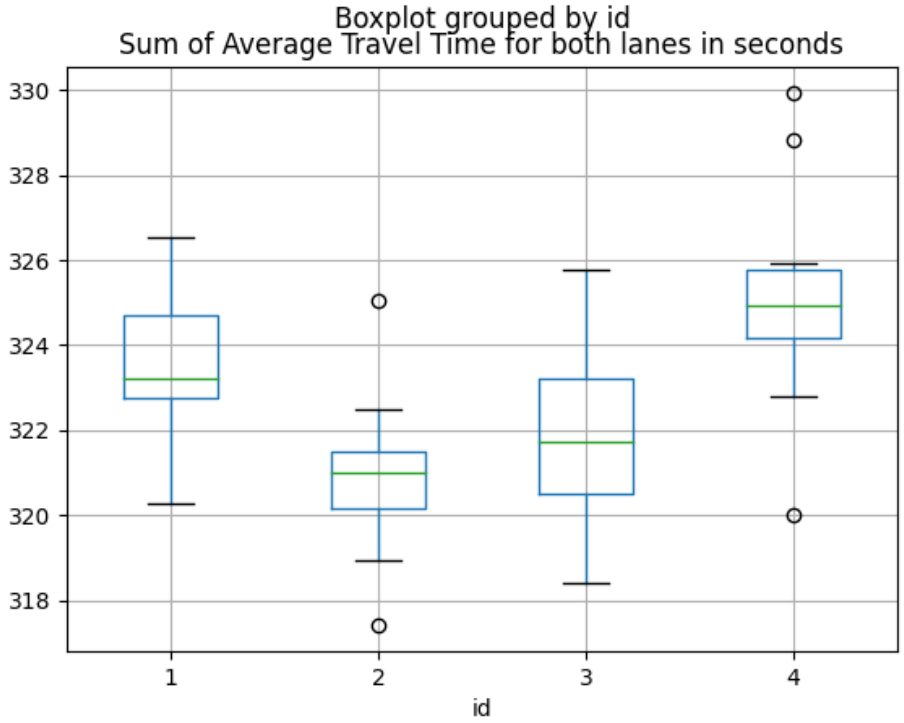

Figure 3.6: Boxplot for TravelTime indicator for single agent approaches.

the reinforcement learning model for ToR disribution in the mixed autonomy scenario.

However, by incorporating the results from the three traffic indicators, some specific new conclusions have been extracted.

One the one hand, the *AvgCavDistance* and *WaitingCount* indicators which represent the average covered distance in automated mode for the C(A)V vehicles and the existence of congestion episodes respectively, have been improved.

On the other hand, the *TravelTime* indicator, the average estimated travel time for every vehicle in simulation, achieved values similar to baseline for the developed models, either in positive or negative way.

Combing these considerations with a deeper understanding about the connection between traffic indicators and simulation, the conclusion that longer covered distance led to bigger vehicles average travel time, was reached.

That's because of the fact that the maximum speed of a C(A)V vehicle in automate mode was lower that the maximum speed in human driving mode. That was the reason why the second variation of the third strategy, which had an average covered distance in automated mode almost 2.5 times bigger than the TransAID model and also managed to decrease the congestion episodes

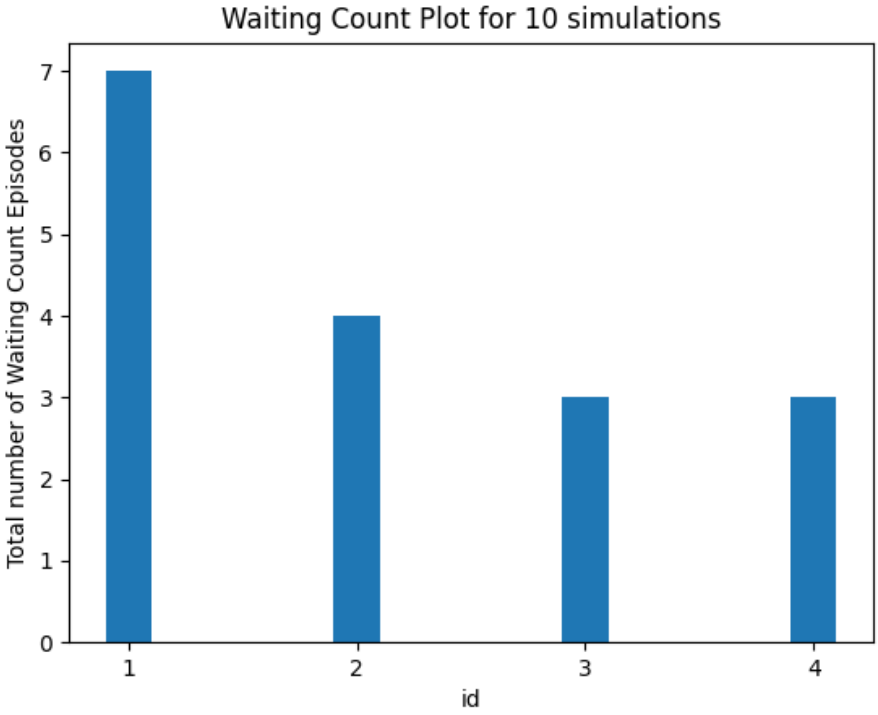

Figure 3.7: Boxplot for WaitingCount indicator for single agent approaches.

to less than half, did not achieve an improvement on the values for average vehicle travel time.

3.3 Multi Agent

The mixed autonomy traffic management scenario which has been studied in this dissertation, could be characterized quite complex as there some uncontrolled parameters in the environment. The most significant one is the transition of automated to human control on vehicles, especially in terms of duration and possible reduced driving performance. The results of them identified in simulation as congestion incidents and unstable traffic flow.

Having already implemented a single agent reinforcement learning approach, a possible development of multi agent approach could provide some additional useful results or observations on the ToR distribution.

The idea behind each strategy, the optimization goal and implementation details from the design phase to real coding information will be documented in order to cover in best way the effort made during this approach. At the end

an analysis of the results in comparison with the baseline will be documented.

Number of Agents

After the necessary code changes for making the framework able to support multi agent reinforcement learning, the number of the agents which will be examined, had to be decided.

The cells idea was going to be part also of this approach, as it would make it easier to assign cells to each agent, in order to control parts of the highway. The number of cells remained 14, where each lane has 7 cells. The length for each cells is the same as before. That means that the first cell in each lane has a length of 300 meters, the last cell 500 meters, and the rest 1700 meters of the 2.5 km lane, was divided in 4 equal parts of 350 meters and one cell (the penultimate one) of 300 meters.

The cells that an agent would control, could be defined in two ways. Either the agent would control sequent cells in one lane, or it would be able to control cells in both lanes. It was decided that the agents would have access to cells in both the lanes.

This approach, made easier the assignment of the cells to an agent and established a more cooperative relationship between them. The tested number of agent was 2, 3 and 6. Based on time restrictions for the developing, training and evaluation of multi agent reinforcement learning models, it was considered wiser to use these three agents combinations, as they could use the single agent cell assignment function with some minor modifications and was also quite representative for multi agent cases.

A visual representation for the scenario with three agents operating at the same time can be found in figure 3.8.

Figure 3.8: Scenario representation for multi agent approach with 3 agents.

The figure show which cells are assigned to which agent. The first agent is observing and is able to send ToR messages to C(A)V vehicles to the first two cells of each lane. The third and fourth cell in each lane is under the control of second agent. The third agent regulates the ToR distribution in

fifth and sixth cell in each lane, whereas the last cell in both lanes operates again as safe gap before the end of autonomous driving zone.

In general, the correlation between agent and cells was defined as follows:

- 2 Agents: Each agent observers and send ToR messages to 6 cells (3 in each lane), with order from left to right.
- 3 Agents: Each agent observers and send ToR messages to 4 cells (2 in each lane), with order from left to right.
- 6 Agents: Each agent observers and send ToR messages to 2 cells (1 in each lane), with order from left to right.

With the definition of the agents in highway and their assignments of cells completed, the workflow has been focused on defining the observation and action space for the available agents and the creation of reward functions suitable for multi agent approach.

The two developed strategies with all the design and implementation insights will be analyzed analytically in the following sections.

3.3.1 First Strategy

The first strategy in multi agent approach, was trying to incorporate the knowledge which has been gained during the development of the third strategy in single agent approach as presented in 3.2.3, with the needed modifications for multi agent support.

Regardless the number of agents (2,3 or 6) which is being examined every time, the observation space, the action space and the reward function should be designed in favor of high flexibility so as to support all the agents combinations.

Thankfully, these modification are well supported by the Rllib library and only small part of the developed code had to be adjusted.

The number of actions for each agent was linked to the number of cells under its control and one more action for not sending any ToR messages.

The observation space contained only values related the controlled cells for each agent. In this way, there was no overlap in the actions of the agents, as well as there was no common knowledge among the agents about the situation of the other cells on the highway.

For the reward function, the majority of the code was directly used from the strategy 3 of the single agent approach and adjusted in terms of code for supporting more agents than one.

After training and evaluation, the developed model was not behaving as expected. At this point it was realized that one of the key point of this reward function, the connection between the selected cell and the one in front of it,

in terms of average cell speed and congestion incidents, could not be used in meaningful way for the agent. The reason behind this was located in the definition of the observation space.

The observation space for each agent was strictly related to the cells which was only controlled by the agent, so there was not any knowledge about the others. It was considered as rational move to extend the modifications in observation space for each agent.

Firstly, a redefinition was crucial. The agent was extracting information for the environment through a new value combination for every cell. That were the average speed of all vehicles in a cell, the number of Connect (Automated) Vehicles, the number of vehicles had received the ToR messages and were performing the transition of control, and an integer value for all the ToR messages which has been sent in this cell until this moment.

Subsequently, two more cells were added to the agent's observations. These were the immediately following cells (one in each lane) of the last two cells that the agent could send ToR messages. For example, in case of three agents, the first agent was able to observe the first two cells (one in each lane) of second agent. The second agent was able to observe the first two cells (one in each lane) of third agent. The third and last agent was able to observe the safe gaps cell (last cell in each lane).

After these changes, the reward function was further adjusted having as key points the following:

- If the agent had managed to avoid the punishment, was receiving a positive reward based on cell influence. This value was calculated based on the same principled as in previous strategies, so it was the cell order in the lane divided by 10 (0.1 for first cell, 0.6 for sixth cell). The cell influence remained the same, as it wanted to give more importance on the latest cells in the accumulated rewards for the system in each action.
- The punishment for an average cell speed value below than 15 meters per second, remained to -1.
- The punishment for a sum of average cell speed values of the chosen cell and the in front of it, below than 40 meters per second, remained to -1.
- The punishment for sending ToR messages in the last cell of any lane, remained to -10.
- The punishment for a congestion incident remained to the extreme value of -10000.
- Introduced sub reward value for not sending ToR messages. This value was positive for all the agents, except the last one, and it was calcu-

lated based on the highest reward which could earn an agent (send ToR messages in the rightest cells in each lane which were under its control), multiplied by 2. The last agent, was receiving punishment of -1, whenever he was deciding not to act, as it would be better this agent to act in every possible moment and send all the ToR messages. That was a way to enforce the ToR distribution to be more concentrated in the rightest agent(s), so as to achieve a large covered distance by vehicles in automated mode.

The final reward for each agent would be based on the examination of the punishments, according to predefined importance priority. In other words, agents actions would get a positive reward only if they didn't trigger none of the punishments, for congestion episode, for sending ToR messages to last cells of the highway and for low average cell(s) speed.

After training and evaluating models for 2,3 and 6 agents, the generated results were not the expected ones. The idea of getting the strategy from single agent approach and adjust it to multi agent strategy, proved to be as successful as expected.

In all combinations, the agents could not coordinate to achieve improvements on traffic indicators and especially the *AvgCavDistance* for all the simulations. These problems, tried to be handled with a new strategy which would have a more aggressive manipulation on forwarding the majority of ToR messages to rightest agents in every case.

3.3.2 Second Strategy

The second strategy in multi agent approach was the last developed one in this dissertation. It was designed with the optimal goal to distribute the majority of ToR messages to the agents which where closest to the end of autonomous zone, where it was possible, so as to avoid possible disruptions in stability of traffic flow.

The observation and action space for each agent remained intact in this new strategy, regardless their combinations in each experiment (2,3 or 6 agents). Every agent had number of actions equal to the cells which it had the right to send ToR messages, plus the action for not sending any ToR messages.

Each cell was represented with four values, the average speed of all vehicles in the cell, the number of Connect (Automated) Vehicles, the number of vehicles had received the ToR messages and were performing the transition of control, and the integer value for all the ToR messages which has been sent until this moment.

Every initialized agents was observing six cell, the four which were under its control and the first two cells (one in each lane) which were under the control of the agent after it. The only unique case existed for the last agent in the highway who was observing the four cells under it control and the last two cells of the highway, known as safe gaps.

The reward function has been redesigned in order to manipulate in different way the agents, but keeping the core assignment for the rewards. This means that the agent will receive either a negative reward, as its action had negative impact or it will receive a positive reward.

The main negative rewards were -10000 for the appearance of congestion episode, -100 for the case of sent ToR messages in safe gaps cells, or -1 for a an average selected cell vehicles speed less than 15 meters per second.

If agent's action was not related to any previous negative cases, it would have the possibility to receive a positive reward. The positive reward for agents was based on the cell influence method, which has been used widely used in past strategies and it is already mentioned in previous subsections.

Firstly, it will be examined if it was an action for sending ToR message or not. For the agent(s) except the last one, the action of not sending ToR messages, returns the highest reward that an agent could (the cell influence of its rightest controlled cell multiplied by 2). The rightest and last agent in the environment would always receive a negative reward of -1 for not sending ToR messages.

Secondly, in cases of sent ToR messages, a new if statement had been introduced, which was working differently again for the last agent. It was using a value, which was representing the sent ToR messages in the cell. It was existed in the observation space of the agent. The if statement was comparing these values for the selected cell and the one after it (the right one in the same lane). When the sent ToR messages in the preferred cell were more than the ToR messages in next one, a reward of -1 was assigned. For obvious reasons, that case could not be examined when the last agent sent messages to its last cell, as the next ones were the safe gaps cells.

This aggressive way of pushing the ToR messages to the rightest possible cell, managed to slightly improve the covered distance in automated driving mode and established a safe direction for focusing future variations on the reward function of the second strategy of multi agent approach.

The main parameter which was detected not to behave as expected during the evaluation of the trained models with the previous reward function, was the manipulation of the rightest (latest) agent which was still not active as should be in sending ToR messages.

That could be justified also in an opposite way, by saying that the last agent acted as it should in order to avoid congestion episodes and disruptions

in traffic flow.

However, a detailed examination on the distribution on cells in this and previous strategies was pointing out that there is still room for improvement.

For this reason, it was introduced an additional if statement, which was targeting only the last agent and was using the value of the Connect (Automated) Vehicles in a cell. It was summing these values of the cells which were addressed to last agent. After that it was punishing the agent with -1 reward in case that the value was not equal to 0. The idea behind this parameterization, was to constantly punish the last agent for existed C(A)V vehicles that had not yet received the a ToR message.

By integrated the new if statement in previous reward function, a second version had been generated with a more aggressive approach. The evaluation of the new trained models proved that the changes had a real impact on the system, providing very interesting results.

The traffic indicators were not improved at the preferred levels and were not better than the previous variations of the same strategy. The changes were showing a shorter covered distance in automated vehicles mode before their transition of control and not significant results on *TravelTime* indicator. The following subsection will present and discuss further the results of trained models with the multi agent approach.

3.3.3 Results Analysis

This section is going to provide a detailed overview of the results which have been produced with the multi agent approach trained model and discuss the related to traffic indicators plots.

The evaluated models of first and second (two variations 2a and 2b) strategies will be compared with the baseline model, the TransAID traffic management solution, for all the agents combination (2, 3 and 6 agents). The mapping of strategies to ID models is presented in table 3.2.

The comparisons will be one the three most representative traffic indicators, the *AvgCavDistance*, *TravelTime* and *WaitingCount*, on the same format as in single approach results analysis.

Table 3.2: Strategy mapping table with IDs (Multi Agent).

Strategy	ID
TransAID	1
Strategy 1 - 2 Agents	2
Strategy 1 - 3 Agents	3
Strategy 1 - 6 Agents	4
Strategy 2a - 2 Agents	5
Strategy 2a - 3 Agents	6
Strategy 2a - 6 Agents	7
Strategy 2b - 2 Agents	8
Strategy 2b - 3 Agents	9
Strategy 2b - 6 Agents	10

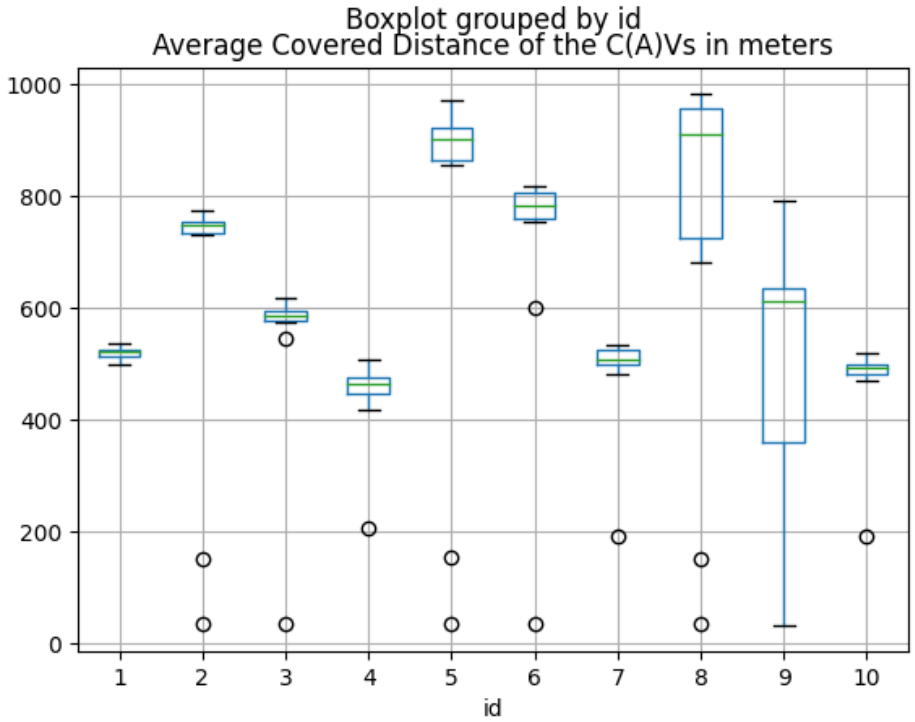

Figure 3.9: Boxplot for AvgCavDistance indicator for multi agent approaches.

Regarding the *AvgCavDistance* indicator, the results are shown in boxplots of figure 3.9.

The first model is the TransAID traffic management solution, which is the baseline and its average covered distance in autonomous mode for the C(A)V vehicles, is around 500 meters.

The models 2,3 and 4 are the models of first strategy for 2,3 and 6 agents respectively. This strategy is using the cell idea in 14 cells configuration (7 cells per lane) and it was the adaptation of the third strategy in single approach, as it presented in 3.2.3, for multi agent purposes.

The models 5,6, and 7 are the produced models with the first variation of second strategy in multi agent approach, for 2,3 and 6 agents, and the last three models 8,9 and 10 are the models of the second variation of the same strategy for 2,3 and 6 agents. Same with first strategy, the second strategy is implemented with 14 cells design with 7 cells per lane.

As it can be seen from the boxplots, none of the models managed to improve the indicator to a value more than 1000 meters. Moreover, almost all the models have at least one outlier, with most of the case two outliers.

Only the model 9 is not presenting outliers, but its lower limit is in the same area with the outliers of the other models. The outliers of the models (2,5,8) with two agents in all strategies are located in the same values and more specifically around 50 and 170 meters, whereas the rest of them are distributed mainly under the 200 meters.

This strange behaviour has been detected on the evaluation of the models of first strategy (2,3,4) and was tried to be solved by the first variation of second strategy. But as can be seen in the results, was not solved successfully, as the models (5,6,7) presented in the same odd behaviour.

One of the reasons behind the aggressiveness on distributing the majority of ToR messages by the last agent, in second variation of second strategy, was this symptom that all the previous models had and it was still presented in last models (8,9,10).

It was named as First Agent Symptom, as the extensive studying of the evaluation results files, pointed out that for some strange reason the first agent in some of the simulations, was acting in way that was ignoring the next agents for all the duration of simulation. In other words, the first agent was sending almost all the ToR messages and there was not proper ToR distribution in one simulation, whereas in next simulation, the behaviour was completely normal.

Regardless the number of agents or the followed strategy, the First Agent Symptom was there. It still has not be cleared if this coming from a design or implementation failure or it is a possible bug in Rllib, the multi agent reinforcement learning library which has been used This is something that should be investigated in future work.

Apart from that symptom, one other pattern which has been detected

is that the 2 agents models (ids 2,5,8) despite the outliers, present higher medians and more solid boxplots than the implementations of 3 (ids 3,6,9) and 6 (ids 4,7,10) agents respectively.

However, in comparison with the baseline model of TransAID solution and by ignoring temporally the First Agent Symptom, only the models with 2 (ids 2,5,8) and 3 (ids 3,6,9) agents improved the indicator, while the 6 (ids 4,7,10) agents approach seems to be not very successful and to present equal to baseline boxplots.

The boxplots for the *TravelTime* indicator in figure 3.10 do not also present significant improvements, but their results are highly connected to *AvgCavDistance* indicator ones.

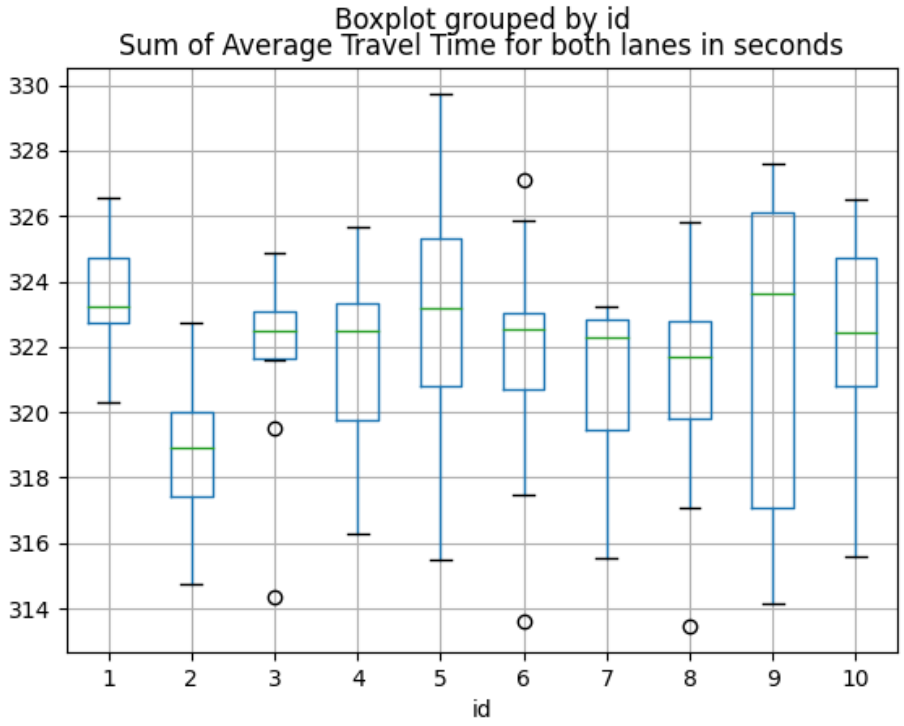

Figure 3.10: Boxplot for TravelTime indicator for multi agent approaches.

The baseline model of TransAID is known that has an efficient traffic management, so its boxplot with no outliers and a median value of 323 seconds, is going to determine if the rest boxplots have a positive or negative impact.

There are some outliers, located in models 3, 6, 8 with no detected pattern as they are different agents variations and strategies and can only be justified by the outliers in boxplots of *AvgCavDistance* indicator.

The model 9, which is the model with 3 agents of second variation in

second strategy, as in the previous boxplots for *AvgCavDistance* indicator, presents a boxplot with very stretched limits. The lower limit of 314 seconds for the average Travel Time of vehicles during the simulations, is explained by the fact that it produced simulation with a very small covered distance in automated mode for the C(A)V vehicles.

The model 5, the two agent model in first variation of second strategy, has its upper limit on the highest value of 330 seconds and the lower limit on around 316 seconds. The upper limit value can be explained by the fact that this model achieved the highest automated covered distance and has been proved since the single agent approach, that a close to 1000 value of the *AvgCavDistance* indicator links to high values for *TravelTime* indicator. The lower limit value is a result of the First Agent Symptom, as two of the simulations were affected by it.

The rest of the models did not present any significant variation, except the two agent model of first strategy, model 2, which seems to improve slightly the indicator, as the median has been dropped to 319 seconds and the boxplot is solid enough.

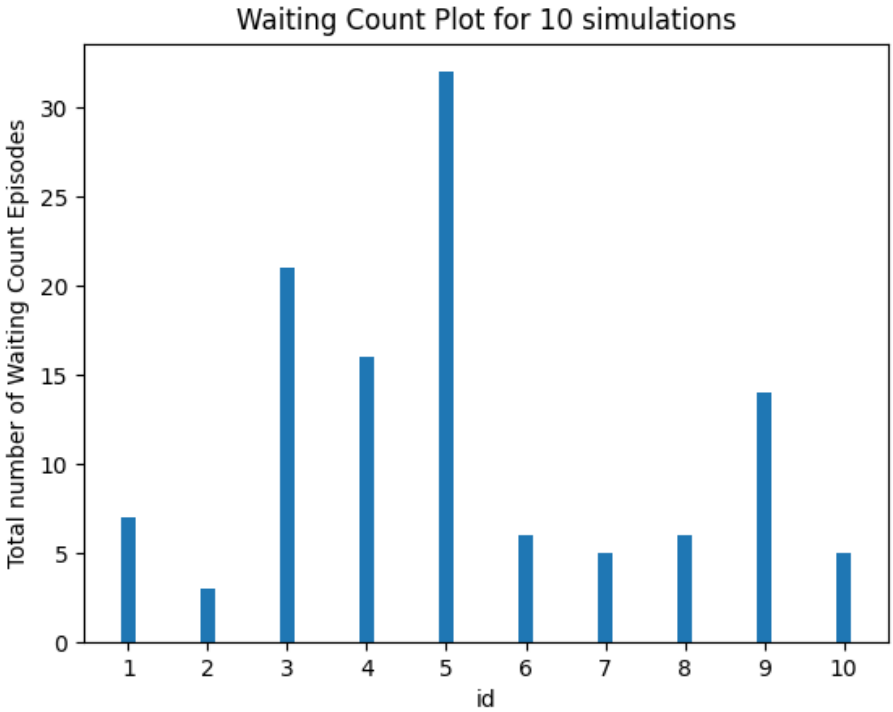

Figure 3.11: Boxplot for WaitingCount indicator for multi agent approaches.

The investigation of *WaitingCount* indicator through the plots in figure

3.11, is going to provide more insight information for the traffic management efficiency in multi agent approach.

For 10 simulations, the TransAID method (model 1) led to the occurrence of congestion episodes for 7 times in total. For the rest of the models, there is no clear evidence that a strategy had produced better results.

Taking for example the results of first strategy, the results are completely different for each agents combination. The model of two agents (id 2) decreased the congestion episodes to a total of 3 incidents, where the models of 3 (id 3) and 6 (id 4) agents have increased the congestion incidents to 23 and 17 respectively.

In first variation of second strategy the results are also very different. The model with two agents (id 5) noted the highest limit of 33 total congestion episodes, whereas the models of three (id 6) and six (id 7) agents decreased the congestion incidents to 6 and 5 in each case.

The second variation of last strategy, provides results where the models of 2 (id 8) and 6 (id 10) agents improved the indicator with values of 6 and 5 respectively and model of 3 agents (id 9) had negative impact on indicator and with a value 16.

A final complete evaluation for the multi agent approach is very hard to be made. The main reason is the First Agent Symptom, which affected all the models for more than one simulation out of ten and could not be faced effectively.

Furthermore, even in same strategies, the models with different agents combination, had way different results in term of traffic indicators.

All the above points out that the multi agent approach was more complicated and was needed more time to be invested on it, not only for designing and developing strategies, but also for understanding the problems and face them properly.

3.4 Overall Results Comparison

In this section all the results for single and multi agent approaches will be presented and summarised.

The mapping of strategies to ID models is presented in table 3.3, where the first id is for baseline, the next three for single agent approach and the rest of them for multi agent appraoch.

Table 3.3: Strategy mapping table with IDs (Both Approaches).

Strategy	ID
TransAID	1
Strategy SA 1	2
Strategy SA 2a	3
Strategy SA2 2b	4
Strategy 1 - 2 Agents	5
Strategy 1 - 3 Agents	6
Strategy 1 - 6 Agents	7
Strategy 2a - 2 Agents	8
Strategy 2a - 3 Agents	9
Strategy 2a - 6 Agents	10
Strategy 2b - 2 Agents	11
Strategy 2b - 3 Agents	12
Strategy 2b - 6 Agents	13

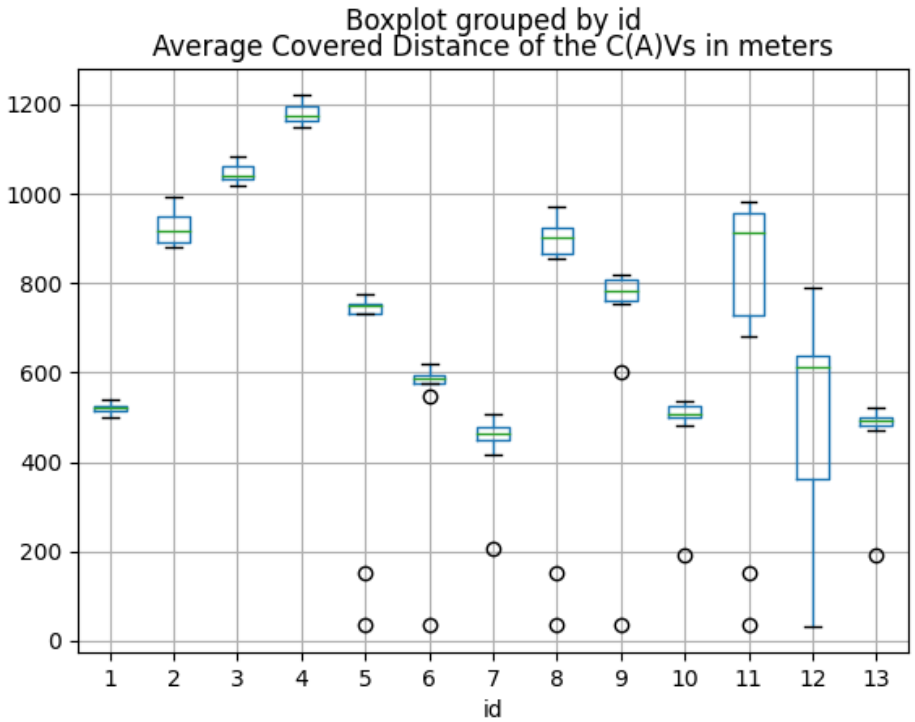

Figure 3.12: Boxplot for AvgCavDistance indicator for both approaches.

The overall results for the *AvgCavDistance* indicator are shown in boxplots of figure 3.12.

The overall better performance of single agent models (ids 2,3,4) in comparison with the multi agent ones (ids 5-13) is obvious. The First Agent Symptom affects all the multi agent models, as there are outliers under the 200 meters. However, even without taking in account the outliers, the multi agent boxplots are located under the single agent ones for all the cases.

The boxplots for the *TravelTime* indicator in figure 3.13 are not giving a different overview than those in *AvgCavDistance* indicator.

The First Agent Symptom has consequences also in the *TravelTime* indicator as in the affected simulations, the C(A)V vehicles could drive in the maximum speed of human driving mode, which is higher than the automated one and could reduce the average travel time for vehicles in this way.

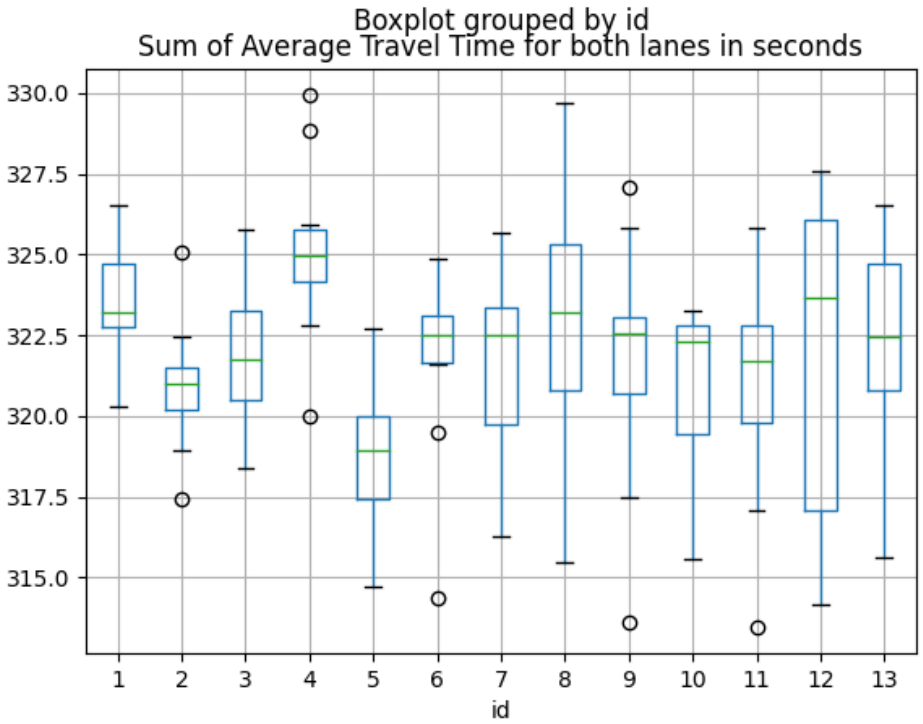

Figure 3.13: Boxplot for TravelTime indicator for both approaches.

However, the only model with three outliers and two of them close to the value of 330 seconds, is the model (id 4) of second variation in third strategy of single agent approach, which achieved the highest value for *AvgCavDistance* indicator.

The overall results for the *WaitingCount* indicator are shown in boxplots of figure 3.14.

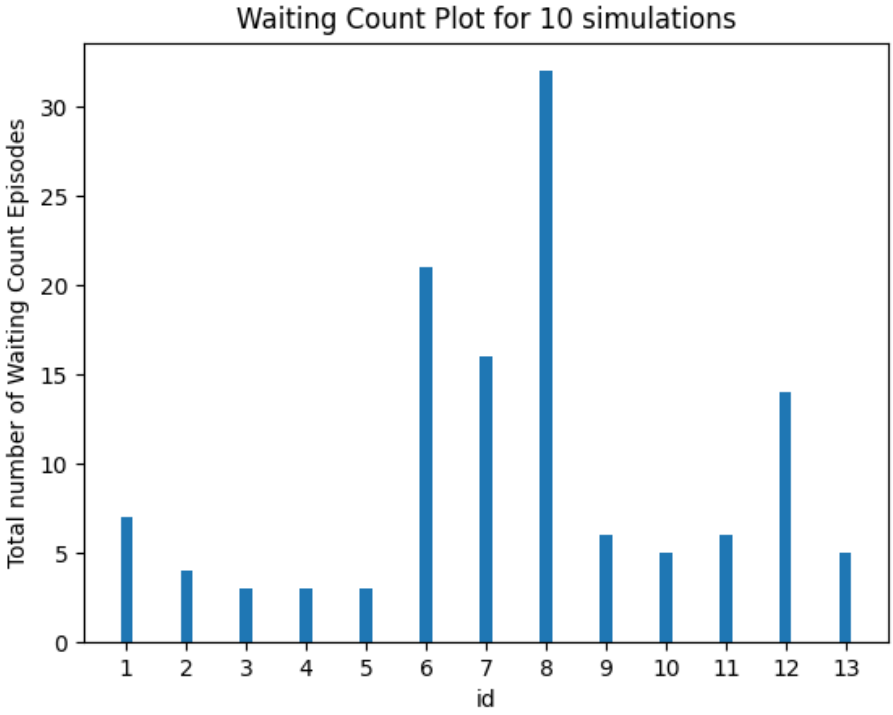

Figure 3.14: Boxplot for WaitingCount indicator for both approaches.

It is very clear, that the multi agent approach did not handle in efficient way the congestion episodes as it can be seen in its results (ids 5-13). For one more time the single agent approached produced better results overall for this indicator.

The First Agent Symptom affected all the multi agent models and the results for all the three indicator have proved it, showing that single agent models was better and more stable in general.

Conclusion

4.1 Conclusions

The produced results from single and multi approach were very interesting in terms of traffic efficiency, as it was expressed by the traffic indicators *AvgCavDistance*, *TravelTime* and *WaitingCount*.

It could be consider easier to choose best strategy for every approach than for both of them. For the single approach the best results for all the three traffic indicators were produced my the model of the first variation in second strategy. It achieved a high value of around 1000 meters for *AvgCavDistance*, improved the *TravelTime* compared to baseline and minimized also the incidents of congestion as the *WaitingCount* value has been dropped to minimum of 3 total times.

On the other hand, in multi agent approach, the First Agent Symptom left its trails to all the results, making really difficult to chose the optimal model. Under these restrictions, the model with two agent of the first strategy, may be the best among the others, as at least performed better than them for all the traffic indicators.

In case of the need for selecting only one model as the most successful then the best model of single agent approach should play this role. The single agent approach was allocated also more time than the multi agent one. This could be a reason for investing more time on multi agent approach and investigating deeper the design principles for more than one agent implementations.

All the results showed that the complete elimination of congestion episodes may not be possible. The causes behind that opinion could be found in the scenario specification, as it may be consider as very high demanding one, or the whole design of the reinforcement learning approaches.

Finally, the goal of keeping the automated mode for the C(A)V vehicles as long as possible, could increase the average travel time of vehicles in simulation. This part is highly important, as it shows that the traffic indicators of *AvgCavDistance* and *TravelTime* could have different directions. The

improvement of *AvgCavDistance* indicator should be more important as the increase in *TravelTime* could be maintained in accepted limits.

4.2 Further Research

There are many different ways of extension for this dissertation, either on solving problems or exploring new methods.

The main problem which appeared was the First Agent Symptom which could be either an implementation and design problem or a bug in the used reinforcement learning library. With more time allocated on the investigation of the causes of this problem, a final solution could come, which will also give a different importance on multi agent models results.

Using reinforcement learning approaches, which are a sub-field of machine learning, automatically the models become very sensitive in stability issues and the robustness of them should be improved. In order words, these models act based on some observation, but a small change on these observations could lead to completely different action. For example, a very small change on average speed value of a cell, normally would not affect the agent's action, at least in human eyes, but it could lead the agent to take a completely different action.

As far as the used reinforcement learning libraries and the available time for training the models, suggests that could be handled in different way. This can be either by allocating better hardware and more time resources to models and using different libraries. There is a chance some of the faced problem to be vanished by using a different set up.

Finally, more mixed autonomy scenarios could be tested with the same algorithms to validate their performance in different situations and the flexibility and stability of the whole framework.

Bibliography

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. USA: Prentice Hall Press, 3rd ed., 2009.

[2] T. M. Mitchell, *Machine learning*. Singapore: McGraw-Hill, 1997. OCLC: 278438226.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[4] L. Deng and D. Yu, "Deep Learning: Methods and Applications," *Foundations and Trends® in Signal Processing*, vol. 7, no. 3–4, pp. 197–387, 2014.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, Nov. 2018. Google-Books-ID: 6DKPtQEACAAJ.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *arXiv:cs/9605103*, Apr. 1996. arXiv: cs/9605103.

[7] R. BELLMAN, "A Markovian Decision Process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. Publisher: Indiana University Mathematics Department.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv:1707.06347 [cs]*, Aug. 2017. arXiv: 1707.06347.

[9] M. van Otterlo and M. Wiering, "Reinforcement Learning and Markov Decision Processes," in *Reinforcement Learning: State-of-the-Art* (M. Wiering and M. van Otterlo, eds.), pp. 3–42, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[10] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018. arXiv: 1811.12560.

[11] "SSM Device - SUMO Documentation." https://sumo.dlr.de/docs/Simulation/Output/SSM_Device.html. (Accessed on 09/20/2020).

[12] "Definition of Vehicles, Vehicle Types, and Routes - SUMO Documentation." https://sumo.dlr.de/docs/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html. (Accessed on 09/20/2020).

[13] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, p. 3–30, Jan. 1998.

[14] "ToC Device - SUMO Documentation." https://sumo.dlr.de/docs/ToC_Device.html. (Accessed on 09/20/2020).

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[16] F. Chollet *et al.*, "Keras," 2015.

[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv:1606.01540 [cs]*, June 2016. arXiv: 1606.01540.

[18] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, *Stable Baselines*. GitHub, 2018. Publication Title: GitHub repository.

[19] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," 2017.

[20] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A Distributed Framework for Emerging AI Applications," *arXiv:1712.05889 [cs, stat]*, Sept. 2018. arXiv: 1712.05889.

[21] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for Distributed Reinforcement Learning," *arXiv:1712.09381 [cs]*, June 2018. arXiv: 1712.09381.

[22] N. Kheterpal, K. Parvate, C. Wu, A. Kreidieh, E. Vinitsky, and A. Bayen, "Flow: Deep reinforcement learning for control in sumo," in *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems* (E. Wiessner, L. Lucken, R. Hilbrich, Y.-P. Flotterod, J. Erdmann, L. Bieker-Walz, and M. Behrisch, eds.), vol. 2 of *EPiC Series in Engineering*, pp. 134–151, EasyChair, 2018.

[23] R. T. Underwood, *Traffic Management: An Introduction*. Hargreen Publishing, 1990. Google-Books-ID: 14VxAAAACAAJ.

[24] A. Correa, S. Maerivoet, E. Mintsis, A. Wijbenga, M. Sepulcre, M. Rondinone, J. Schindler, and J. Gozalvez, "Management of Transitions of Control in Mixed Traffic with Automated Vehicles," in *2018 16th International Conference on Intelligent Transportation Systems Telecommunications (ITST)*, pp. 1–7, 2018.

[25] R. Alms, Y.-P. Flötteröd, E. Mintsis, S. Maerivoet, and A. Correa Vila, "Traffic management for connected and automated vehicles on urban corridors - distributing take-over requests and assigning safe spots," 07 2020.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *arXiv:1602.01783 [cs]*, June 2016. arXiv: 1602.01783.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv:1312.5602 [cs]*, Dec. 2013. arXiv: 1312.5602.

[28] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *arXiv:1509.06461 [cs]*, Dec. 2015. arXiv: 1509.06461.

[29] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," *arXiv:1511.06581 [cs]*, Apr. 2016. arXiv: 1511.06581.

[30] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *arXiv:1511.05952 [cs]*, Feb. 2016. arXiv: 1511.05952.