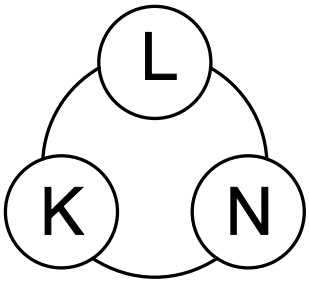


Technische Universität München
Lehrstuhl für Kommunikationsnetze
Prof. Dr.-Ing. Wolfgang Kellerer

Master‘s Thesis

Distributed Resource Allocation with Multi-Agent Reinforcement Learning for 5G-V2X Communication

Author:	Alperen Gündoğan
Address:	Biedersteiner Straße 28
	80805 Münich
	Germany
Matriculation Number:	03694565
Supervisor:	M.Sc. Murat Gürsu, Dr. Volker Pauli
Begin:	01. August 2019
End:	03. March 2020

With my signature below, I assert that the work in this thesis has been composed by myself independently and no source materials or aids other than those mentioned in the thesis have been used.

München, 03.03.2020	
Place, Date	Signature

München, 03.03.2020	
Place, Date	Signature

Abstract

Vehicle-to-Everything(V2X) communication where the vehicles exchange messages between other vehicles, infrastructure or pedestrians has vital importance in order to prevent road accidents that cause millions of death annually according to the World Health Organization(WHO)[Wor18]. V2X enables vehicles to broadcast safety messages including their positions, velocity, collision warning information to reduce road accidents. Furthermore, 3GPP is working on 5G-V2X to support advanced driving applications e.g. remote control, platooning to provide a safer driving experience. In 5G-V2X, the presence of a base station to allocate frequency and time resources to vehicles for transmission can not always be guaranteed. Thus, 5G-V2X supports distributed resource allocation where the vehicles decide to use radio resources on their own based on their local observation e.g. without interacting with a base station. The standardized algorithm in LTE-V2X which is also supported in 5G-V2X e.g. mode 2 transmission is known as Semi-Persistent-Scheduling(SPS). The SPS algorithm senses the resources used by other vehicles and determines available resources for transmission. The drawback of this algorithm is that near vehicles are likely to choose the same resources for transmission due to similar spatial-local sensing results which leads to persistent collisions [Res19a]. Contrarily, it is of utmost importance to receive safety messages from nearby vehicles.

In this thesis, we propose a distributed resource allocation mechanism using multi-agent reinforcement learning which selects resources based on local information. Specifically, we try to answer the following question; *Can we design a distributed scheduling algorithm with multi-agent reinforcement learning to maximize packet reception ratio(PRR) while sacrificing the PRR of far vehicles via making them use the same resource?* Our proposed method is based on the view-based positional distribution(VPD) which is inspired by multi-agent common knowledge reinforcement learning [dWFF+18]. Each vehicle observes and piggybacks the positions of other vehicles together with safety messages to ensure that all vehicles know the positions of other vehicles. Then, each vehicle observes the positions of others from its own view and generates VPD vector as an input for algorithm. The intuition is that if vehicles in the observation range of each other can know the positions of each other, then a vehicle can estimate the observations of the others from its own observation and selects a resource by estimating the resources will be used by others to maximize the system objective.

The proposed method is compared with the standardized SPS algorithm. Our approach outperforms SPS on average PRR over time and provides lower information age for the near vehicles. Furthermore, PRR value within near vehicles is higher than the SPS algorithm. Hence, vehicles sacrifice the communication with far vehicles to ensure reliable transmission with the vehicles in the vicinity.

Acknowledgments

I would like to express my gratitude to everyone who helped me throughout this thesis.

First of all, I would like to thank my supervisor Dr. Volker Pauli at Nomor Research GmbH, Munich for giving me the opportunity to work on this interesting project, for always being available for questions and steering me in the right direction whenever I encountered problems and needed clarification. I would also like to express my sincere appreciation to M.Sc. Murat Gürsu of the Chair of Communication Networks at Technical University of Munich for valuable discussions.

Finally, I also would like to thank my parents and Elgiz Bagcilar who have always showed their unfailing support and continuous encouragement for all this time.

Contents

Contents 5

1 Introduction 7

2 Background 10
2.1 Overview of V2X Communication . . . 10
2.1.1 5G-V2X . . . 11
2.1.2 5G-V2X Enhancements . . . 12
2.2 Sidelink Radio Access in 5G-V2X . . . 13
2.2.1 5G-V2X Sidelink Mode 2 . . . 13
2.2.2 Semi-Persistent Scheduling . . . 14
2.3 Reinforcement Learning Overview . . . 15
2.3.1 RL Elements . . . 16
2.3.2 Notations . . . 16
2.3.3 Markov Decision Processes(MDPs) . . . 17
2.3.4 Formulation . . . 18
2.3.5 Q-Learning with Function Approximation . . . 19
2.3.6 Deep Neural Networks(DNN) . . . 20
2.3.7 Recurrent Neural Networks(RNN) . . . 20
2.3.8 Deep Q Networks(DQN) . . . 22
2.3.9 Deep Recurrent Q Networks(DRQN) . . . 24
2.4 Multi-Agent Reinforcement Learning . . . 24
2.4.1 Stochastic/Markov Games . . . 24
2.4.2 Training Approaches . . . 26
2.5 State Of The Art . . . 27
2.5.1 Dynamic Spectrum Access . . . 29
2.5.2 Position Based Scheduling . . . 30

3 Algorithm Design 31
3.1 Problem Definition . . . 31
3.2 State Design . . . 33
3.2.1 Neighbor Table Updates . . . 34
3.2.2 View-based Positional Distribution . . . 35
3.3 Action Space . . . 35
3.4 Reward Design . . . 36
3.5 Deep Neural Network Approximation . . . 37

4 Implementation 38

4.1 Distributed MARL Scheduler . . . 38
4.1.1 Architecture . . . 38
4.1.2 Centralized Training Decentralized Execution . . . 39
4.2 Environment Setup . . . 42
4.2.1 Test Simulator . . . 42
4.2.2 Real Simulator . . . 42
4.2.3 Software . . . 43

5 Results **46**
5.1 Training Performance . . . 46
5.2 Discussions . . . 49
5.2.1 Why Reinforcement Learning? . . . 49
5.2.2 Discount Factor . . . 50
5.2.3 Granularity of Observations . . . 50
5.2.4 Impact of LSTM . . . 52
5.2.5 Random Velocity . . . 52
5.2.6 Higher Velocity . . . 52
5.2.7 Common Knowledge Learning . . . 54
5.2.8 Networked Agents . . . 54
5.3 RealNeS Analysis . . . 54
5.3.1 Semi-Persistent Scheduling . . . 55
5.3.2 Network Settings . . . 55
5.3.3 Packet Reception Ratio . . . 55
5.3.4 Information Age . . . 57

6 Conclusions and Outlook **64**

List of Figures **66**

List of Tables **67**

List of Abbreviations **68**

Bibliography **70**

Chapter 1

Introduction

1.35 million deaths around the world annually are caused by road accidents according to the report of the World Health Organization(WHO) [Wor18]. Also, the expenditure of car collisions in the United States is $300 billion per year as stated in the American Automobile Association(AAA) study [Ame11]. The recent advances in the automotive industry and information technology pave the way for collaboration to offer high-end applications such as driving safety, remote control, platooning, and new infotainment experiences for passengers. Autonomous vehicles can combine the information received by other entities in the vicinity along their sensor and computing intelligence to provide safer driving experience and even collaboration to utilize traffic flow. Fortunately, Vehicle-to-everything(V2X) technology can enable a vehicle to communicate with not only with other vehicles but also with pedestrians, infrastructure and even internet. Analyses by the U.S. Department of Transportation's National Highway Traffic Safety Administration showed that approximately 80 percent of the road accidents involving non-impaired drivers can be avoid with V2X communication [HS14]. Thus, the most crucial one among the other applications of connected and autonomous vehicles is the safety.

In vehicular safety communication, each vehicle broadcasts Cooperative Awareness Messages(CAMs) periodically which consists of its geographic position, speed, heading. Some of the use cases of the CAMs are emergency vehicle warning, collision risk warning, etc. [CS18]. The neighboring vehicles can expedite reaction time in responding to possible collision by benefiting from the information in CAM, therefore reduce the collision risk. Furthermore, autonomous driving cars can operate only in line-of-sight condition with a limited range of 100-200m whereas vehicular communication can disseminate information over a larger range and promotes advanced driver-assistance system(ADAS) which now enables cars to see from "someone else's eyes".

There are two key Radio Access Technologies(RATs) for V2X communication; Dedicated Short Range Communications (DSRC) and Cellular-V2X (C-V2X). DSRC is an 802.11p-based wireless communication technology that is designed to operate only in the 5.9 GHz band whereas C-V2X can also operate in the cellular operators' licensed carrier [WMG17]. DSRC technology suffers from poor scalability and challenges caused by a high mobility environment [NMCP19]. Further studies also show the advantage of C-V2X over DSRC technology in terms of its additional link budget, higher resilience to interference and better non line-of-sight (NLOS) capabilities [GAA18]. Therefore, C-V2X technology which has been developing under 3GPP is a suitable candidate for the future of

V2X communication. 3GPP is now working on New Radio(NR) V2X to support advanced applications that require much more stringent QoS requirements than the applications supported by C-V2X [3GP18b].

The development of Vehicle-to-Everything communication has started in release 12 of 3GPP in March 2015 with the proximity services to be exploited for public safety use cases. Later, 3GPP has introduced C-V2X which was built upon the previous standard with improvements to fulfill the low latency requirement of V2X communication. 3GPP release 14 also introduced new sidelink transmission modes where resource allocation is conducted either by the network or in a distributed manner, directly by the vehicles. The latter one, called Mode 4, and is required when a vehicle is outside of the coverage area of a base station. Mode 4 transmission is also preferred to alleviate the burden of a base station and to diminish signaling overhead which can lead to the violation of QoS requirements for low latency applications. The standardized approach for mode 4 transmission is called Semi-Persistent-Scheduling(SPS). SPS algorithm works in a distributed manner where each user listens to the time and frequency resources over the last 1000 subframes e.g. 1sec and determines the best available transmission resources in every resource reservation interval(RRI). The selected resources are used for a number of subsequent transmissions where the number is determined randomly based on the RRI value. SPS benefits from the periodicity of RRI and predictable sizes of the safety messages. However, it is likely that the close vehicles can select the same resources due to the spatial sensing, then the collision will persist due to the nature of the algorithm and leads to high Information Age(IA) [Res19a].

High mobility of vehicles leads to the varying environment and the demanding QoS requirements of V2X applications challenge the usage of conventional resource allocation methods which are mostly designed for static or low-mobility environment assumptions [LMY+]. Furthermore, close vehicles are likely to use the same resource in congestion case so that a vehicle can not receive any safety message from the close vehicles. This is for sure is undesirable for 5G-V2X safety communication.

These reasons motivate us to discover novel approaches like multi-agent reinforcement learning for distributed resource allocations in V2X communication. Recently, reinforcement learning approaches with deep neural network approximation that can extract inside statistics from the available data e.g. channel condition, traffic pattern for resource management in V2X have been exploited to promise better performance than traditional approaches [CLN19, SB18a, FBZ19, YLJ19, ZLGV, NC17]. The combination of the multi-agent reinforcement learning with one of key features of 5G i.e. flexible numerology for distributed resource allocation has not been considered in the previous studies, and offers a good research direction.

In this work, we adopt multi-agent reinforcement learning(MARL) to solve the distributed resource allocation problem. Each user observes the environment locally and selects a resource for transmission. The algorithm and reward is designed to encourage the users not choose the same resources for a transmission. Also, we assumed that packet reception ratio can be maximized if far vehicles use the same resources in congestion. Thus, we encourage these behaviours in the reward design. In this thesis, we propose a unique state design with view-based positional distribution(VPD) vector. Each vehicle piggybacks a neighboring table including the positions of other vehicles together with CAM message, e.g. therefore it does not create extra burden. The vehicles create view-based positional

distribution based on this table before requesting a resource from the MARL scheduler. The details of VPD are explained in Section 3.2.2.

For training, we use a common *centralized training, decentralized execution* framework with parameter sharing to train the policy. After the training, each user exploits the trained policy based on the locally observed state.

Main contributions of this work can be summarized as follows;

- To the best of our knowledge, this is the first work for distributed resource allocation for 5G-V2X communication in the absence of the base station using a novel multi-agent reinforcement learning approach.
- We formulate the state representation of the learning model different than the existing approaches in the literature. We used the positional distribution of the vehicles around observed from the perspective of the transmitter as a part of the state instead of observing channel information e.g. interference or occupancy of channels. Observing channel information as a part of the state makes the problem non-stationarity since the next state and reward of the transmitter is affected by the actions of the others. In this work, we use the positional distribution of the vehicles as a part of the state which varies with the system dynamics e.g. velocity, not by the actions of vehicles. The only received reward is depending on the actions of the other vehicles. Thus, in our work, we can alleviate the impact of non-stationarity and make the problem tractable and solved without requiring additional feedback for transmission.
- We tested the performance of the proposed approach in a realistic 5G system-level simulator and compared the performance with a standardized SPS algorithm. We analyzed the performance not just for the packet reception ratio but also for the information age.

In Chapter 2 background information for the concept is provided. The design of the proposed algorithm including state, action space and reward design are described in Chapter 3. The implementation of the proposed distributed scheduling algorithm and environment setup are presented in Chapter 4. The training performance of the proposed algorithm is revealed in the following Chapter 5. Also, the performance of the proposed algorithm is compared with the standardized distributed SPS algorithm in terms of packet reception ratio and information age. At the end, the final remarks for the results and for the future work are discussed in Chapter 6.

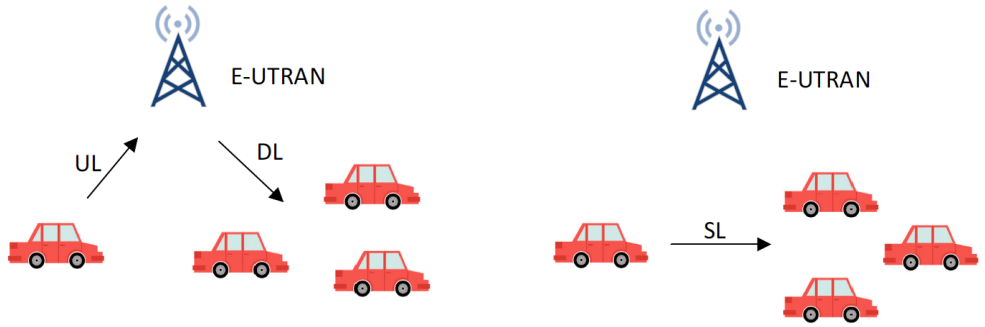
Chapter 2

Background

In this chapter, the required background for the next chapters of this thesis is provided. Throughout the chapter, an overview of the 5G-V2X communication is given and standardized semi-persistent scheduling for distributed resource allocation is briefly explained. Then, the reinforcement learning concepts and deep learning architecture LSTM are clarified. Lastly, the state of the art and contribution of this thesis are mentioned.

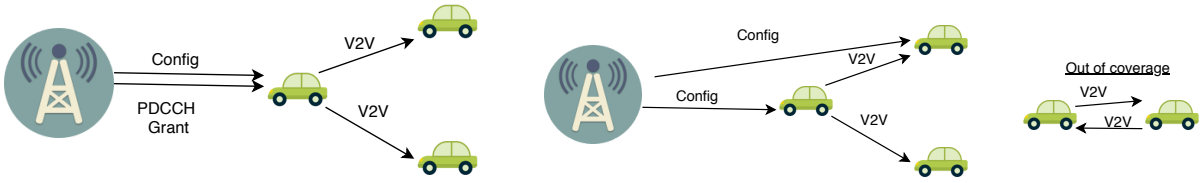
2.1 Overview of V2X Communication

Vehicle-to-Everything communication enables message exchange between vehicle-to-vehicle(V2V), vehicle-to-infrastructure(V2I), and vehicle-to-pedestrian(V2P). The development of V2X communication has started in release 12 of 3GPP in March 2015 with the proximity services to be exploited for public safety use cases. Later, 3GPP has published the first version of C-V2X with release 14 in September 2016 as an alternative to IEEE 802.11p. C-V2X can support both classical cellular air interface(named Uu) and the sidelink(PC5) air interface. The vehicles that use the cellular air interface first transmit their message to the base station in the uplink and the base station forward the message to the destination vehicles in the downlink. The sidelink interface enables vehicles to directly communicate with the other vehicles in the vicinity without using a base station as a relay. Vehicles can use the sidelink interface both in-coverage and out-of-coverage scenario. The representation of both air interfaces are shown in Figure 2.1.



(a) Cellular interface(Uu) (b) Sidelink interface(PC5)

Figure 2.1: Air interfaces [WSGY19]



(a) Mode 3: Scheduled by base station (b) Mode 4: Autonomous scheduling

Figure 2.2: C-V2X resource allocation [Res19b]

Scenario	**Max latency (ms)**	**Reliability (%)**	**Data rate (Mbps)**
Vehicle platooning	10-25	90-99.99	50-65
Advanced driving	3-100	90-99.999	30-53
Extended sensors	3-50	90-99.999	50
Remote driving	5	99.999	UL: 25 DL: 1

Table 2.1: V2X application requirements [3GP18a]

C-V2X introduces two modes of operations i.e. mode 3 and mode 4 for V2V communication [3GP17c]. In mode 3 operation, radio resources are selected and managed through a cellular interface by cellular network for direct V2V communications. However, safety services can not rely on the existence of cellular infrastructure in coverage. Thus, mode 4 operation is designed to operate without cellular coverage, radio resources for transmission are selected autonomously by vehicles. Mode 4 transmission is also preferred to alleviate signaling burden of base stations and to diminish signaling overhead which can lead to the violation of QoS requirements for low latency applications. Mode 4 is a standardized distributed resource allocation scheme in 3GPP as an alternative to 802.11p. The following Figure 2.2 illustrates the operating modes for resource allocation in C-V2X communication.

2.1.1 5G-V2X

5G-V2X also known as New Radio(NR)-V2X, is designed to support advanced V2X applications that are not supported by C-V2X [3GP17d, 3GP17c]. The 3GPP defined four applications for enhanced V2X scenarios; vehicle platooning, advanced driving, extended sensors, and remote driving [3GP18a]. The requirements for those applications are listed in the Table 2.1. 5G-V2X does not only support those advanced applications but also the basic safety applications supported by C-V2X.

The following study items determined by 3GPP summarizes the objectives of NR-V2X [NMCP19];

- *Enhanced sidelink design*, to support advanced V2X applications.
- *Uu interface enhancements*, to support advanced V2X applications.
- *Uu interface based sidelink allocation/configuration*, to improve the sidelink resource allocation through Uu interface.

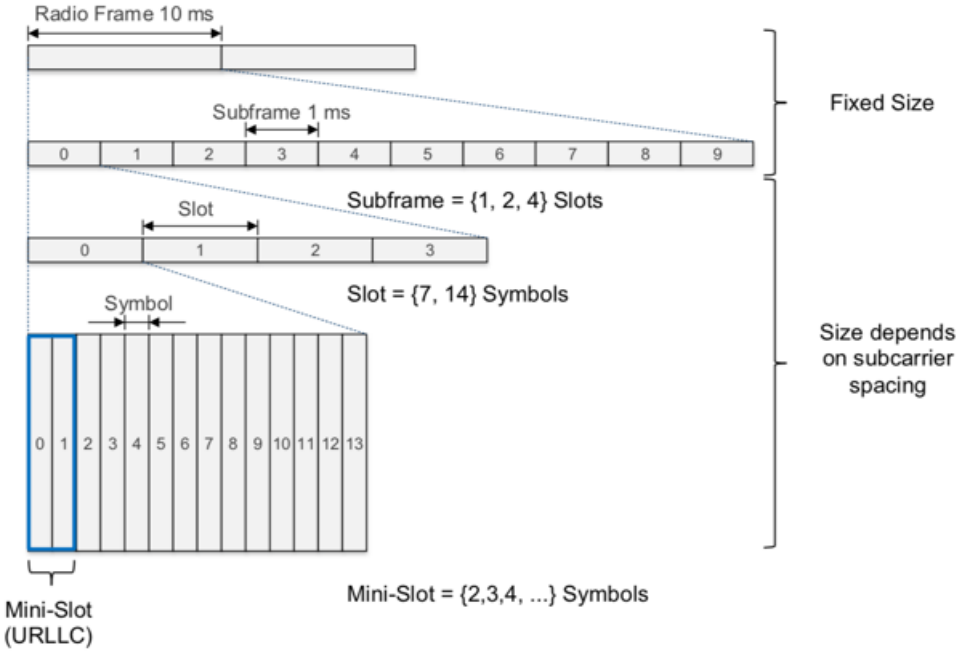


Figure 2.3: 5G Frame Structure [FK17]

- *RAT/Interface selection*, to identify the best interface (among LTE sidelink, NR sidelink, LTE Uu and NR Uu) for a given V2X message transmission.
- *QoS Management*, to satify the QoS requirements of different radio resources.
- *Coexistence*, to solve the coexistence problem of C-V2X and NR-V2X within a single device.

In this work, we focus on the distributed resource allocation in which vehicles determine resources based on their own observation for NR-V2X also known as 5G-V2X.

2.1.2 5G-V2X Enhancements

In this section, some of the important enhancements of V2X communication provided by 5G technology are discussed.

Flexible numerology

One of the key features of 5G introduced in 3GPP Rel. 15 is the support of flexible numerology. Numerology is defined by its subcarrier spacing (the width of subcarriers in the frequency domain) and by its cyclic prefix. Radio frames and subframes are of fixed lengths and are the same as in LTE for better LTE-5G co-existence. However, slots and symbols are of flexible length depending on subcarrier spacing (Figure 2.3). 5G supports sub-carrier spacing of 15, 30, and 60 kHz for sub-6 GHz(i.e. Frequency Range 1, FR1), and 60 and 120 kHz for frequency bands above 6 GHz (i.e. FR2) [R1-18a]. The usage of higher sub-carrier frequencies also reduces the packet transmission time.

The impact of flexible numerology structure for V2X communication has been studied in [R1-18c] and results indicate that large sub-carrier spacing performs better in terms

of block error rate, especially at higher velocities. Thus, in this work, we use larger subcarrier spacing with lower TTI so that at each subframe we have only use one subchannel to transmit the message.

Minislot Scheduling

The other key feature of 5G is the support of mini-slot scheduling. In LTE, a user has to wait for the subsequent slot for its transmission and it allocates the whole slot for its transmission even if it has a small amount of data to send. Mini-slot scheduling enables a user to transmit at any of the 14 OFDM symbols and users can allocate any number of symbols for their transmission. This feature is crucial for low latency applications such as Ultra-Reliable Low Latency Communications (URLLC). Furthermore, users can aggregate slots to form multi-slot to transmit large packets.

Sidelink Feedback Channel

5G-V2X introduces a new feedback channel, Physical Sidelink Feedback Channel(PSFCH) to enable feedback based re-transmissions and acquire channel state information. Feedback channels can also be exploited to assist the radio resource allocation of the other users. For example; users can notify the other users in the vicinity about its preferred resource to utilize resource selection through the feedback channel [R1118]. In our work, we don't have any explicit feedback channel since it creates an extra burden on resources but we piggyback the neighboring table that holds the positions of vehicles together with the CAM messages to assist resource allocation of other vehicles.

2.2 Sidelink Radio Access in 5G-V2X

5G-V2X defines two modes of operations for sidelink communication like C-V2X; mode 1 and mode 2. *Mode 1* is similar to the mode 3 in C-V2X, i.e. radio resources for direct sidelink communication are allocated by the base station. Thus, mode 1 can only operate for the vehicles in the coverage of the base station. *Mode 2*, on the other hand is designed to operate at out-of-coverage scenarios.

2.2.1 5G-V2X Sidelink Mode 2

5G-V2X sidelink mode 2 consists of 4 different submodes unlike in C-V2X mode 4 [R1118].

- Mode 2 (a): Each vehicle autonomously select the resources for transmission. This is similar to C-V2X mode 4.
- Mode 2 (b): Vehicles assist other vehicles to facilitate their resource selection. The assistance can be conducted via the feedback channel.
- Mode 2 (c): Vehicles exploit the pre-configured sidelink grants for their transmission.
- Mode 2 (d): A vehicle selects radio resources for a group of vehicles in the vicinity. This mode is useful especially for platooning applications where a group of vehicles drive through the same direction with small relative speeds [R1-18b].

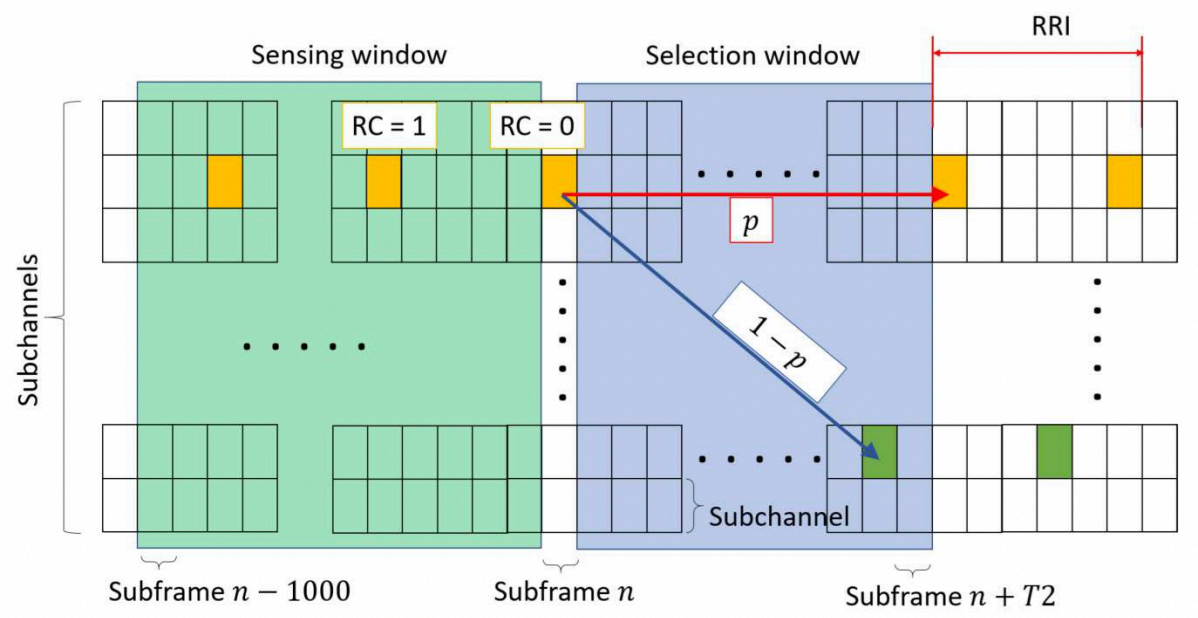


Figure 2.4: Reselection in SPS; p is the resource keeping probability [JKK18]

In this thesis, we are interested to design a distributed resource allocation technique that provides a better packet reception ratio than 5G-V2X mode 2 with the novel multi-agent reinforcement learning approach. Therefore, in the next section, we analyze the details of the semi-persistent scheduling that is specified in Release 14 [3GP17b] and will be used for both C-V2X mode 4 and NR-V2X mode 2 and discuss its performance and drawbacks.

2.2.2 Semi-Persistent Scheduling

SPS algorithm in LTE mode 4 works in a distributed manner where each user listens to the time and frequency resources over time and determines transmission resources in every resource reservation interval(RRI). SPS benefits from the periodicity of RRI and predictable sizes of the safety messages. Semi persistent reservation of the selected resources facilitates the understanding of free resources for other vehicles. Main processes of the algorithm are *sensing*, *selection*, and *reselection* [JKK18].

Sensing

The algorithm relies on the sensing of the resources used by other users. Transmission resources are separated over time and frequency axes and divided into subchannels in frequency and subframes in time. The radio resources are divided into time and frequency domain as subframes and subchannels, respectively. Each user monitors RSSI values in the sensing window which consists of the last 1000 subframes. If the received signal strength indicator(RSSI) on the observed resource is higher than sidelink RSSI threshold, a resource is considered as busy. The transmission sub-frame is selected among the resources left after excluding busy resources from the selection window. Vehicles use a threshold to determine whether a resource is used by other vehicles or not.

Selection

A user select the resource from the selection window which is defined as $[n + T_1, n + T_2]$ where n is the start time of current subframe, $T_1 \leq 4$ and $20 \leq T_2 \leq 100$ [3GP17b]. Available resources are determined after excluding the two groups' resources. First, a user removes the single subframe m if the user was transmitting in subframe $m - RRI.j (j \geq 1)$ since it can not sense the RSSI value due to half-duplexity. RRI is the resource reservation internal and it is related with the periodicity of the safety messages. For example; if the periodicity of messages is 100ms, then a vehicle exploits the same resource at every 100ms. Second, busy resources that are specified in sensing are omitted from the available resources. If the remaining resources after excluding those two groups are less than 20% of the selection window, the sidelink RSSI threshold is increased 3 dB and the selection procedure is repeated. If we have more than 20% resources, we choose the ones with the lowest RSSI values, then a transmission resource is selected randomly within the best 20% of the resources.

Reselection

After each selection process, a user randomly pick a *reselection counter(RC)* within the interval $[C_1, C_2] = [5, 15]$ for $RRI = 100ms$. Then, selected resources is exploited for number of transmissions which is determined by *reselection counter* [3GP17a]. *reselection counter* is reduced by one after each transmission and when it reaches to zero, a user either continues to use the same resource with a probability *probResourceKeep* or choose a new resource as explained in *selection* procedure. The standard specifies the *probResourceKeep* within $[0, 0.8]$. Figure 2.4 illustrates the reselection process in SPS.

Although sensing-based SPS is a convenient approach for the periodic safety message in V2X communication, it suffers from the persistent nature of the algorithm. If two users close to each other happen to select the same resource, the collision will persist for a while which leads to high information age [Res19b]. This behaviour of the SPS in undesirable, especially for safety messages since the most valuable information for a vehicle is to receive safety messages from the vehicles in the vicinity.

Thus, in this thesis we investigated this problem of SPS and proposed a solution using multi-agent reinforcement learning for autonomously resource allocation for V2X communication. In the next section, we briefly explain the general concept of reinforcement learning and examine the methods exploited in this thesis.

2.3 Reinforcement Learning Overview

Reinforcement learning is a machine learning technique that learns based on trial-and-error. The algorithm, considered as an agent, takes an action each time step and receives the next state and reward from the environment for the taken action. The agent evaluates the taken action on the state based on the received reward which can be positive or negative(punishment or mistake). After many interactions with the environment, the agent is able to match the best actions for the given state that maximize its long term reward.

The goal of the agent is to learn the best sequence of actions called a policy that maximizes its long-term reward. When an agent discovers all possible states and actions, then

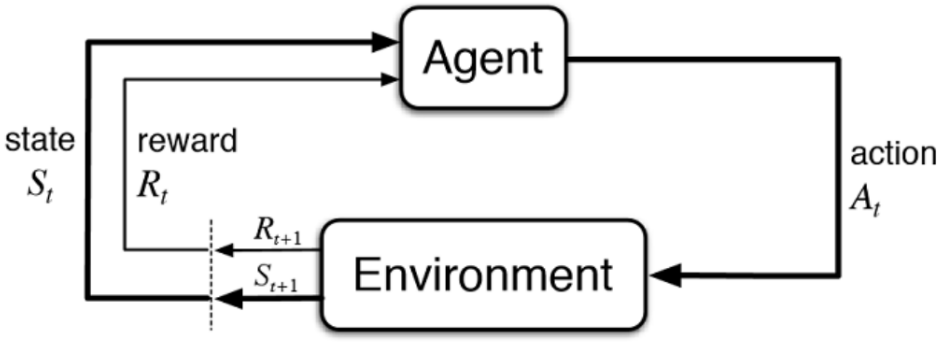


Figure 2.5: The agent–environment interaction in a Markov decision process [SB18b]

it is able to find the optimal policy(sequence of actions) that maximizes its reward. Reinforcement learning problems can be formulated using the ideas from dynamic systems theory i.e. optimal control of incompletely-known Markov decision processes.

In order to understand how to formulate a problem with reinforcement learning, we would need to develop intuition about the key elements in RL and Markov decision processes.

2.3.1 RL Elements

We will start with the formal definitions of some of the key elements.
Agent: A hypothetical entity that takes decisions at each step time and learns from those decisions based on the received reward from the environment.
Environment: Demonstration of the problem to be solved. In this thesis, the scenario is vehicular communication networks.
State: Situation of the agents at a specific time-step in the environment. An agent observes a new state from the environment whenever it performs an action.
Reward: An immediate return sent back from the environment to evaluate the last action of the agent.

Action can be any decision that an agent wants to learn and state is modeled such that it includes useful information to cater to learning. Anything that the agent cannot change arbitrarily is considered to be part of the environment. The calculation of reward is considered as a part of the environment. Although the agent can construct the reward function internally after many interactions with the environment, it can not change the rewards arbitrarily. The reward is only received after each interaction with the environment.

An agent interacts with the environment at each time-step and receives the next state and reward based on the performed action. The Figure 2.5 illustrates this agent-environment interaction procedure.

2.3.2 Notations

In the section, we describe some of the symbols and notations before diving into the details of RL. In general, we use lower script to indicate the time-step. and upper script i for agent id. $Q^{\pi}(s, a)$ is the value function Q by following the policy π. Table 2.2 provides

a concise reference describing the notation used throughout this thesis.

a	a scalar (integer or real), refers to an action
$\mathbf{a}$	a vector, joint action
t	discrete time-step
r	reward
s, s'	states
a_t, A_t	action at time t
s_t, S_t	state at time t
r_t	reward at time t
$\mathbf{a}_t$	joint action at time-step t
a_t^i	element i of vector $\mathbf{a}$ at time-step t with indexing starting at 1
r_t^i	reward of element i at time-step t
$\mathcal{R}$	set of all possible rewards, a finite subset of $\mathbb{R}$
$\mathcal{R}_s^a$	reward for the given state s and action a
$\mathcal{S}$	set off all states
$\mathcal{A}$	set off all actions
π	policy (decision-making rule)
$\pi(a\|s)$	probability of taking action a in state s under *stochastic* policy π
$\|\mathcal{A}\|$	number of elements in set $\mathcal{A}$
$\mathcal{P}_{ss'}^a$	state transition probability from s to s' under action a
$\mathbb{E}$	expectation operator
$\mathbb{E}_\pi$	expectation under policy π
$*$	element-wise multiplication

Table 2.2: Summary of notations

2.3.3 Markov Decision Processes(MDPs)

Sequential decision-making problems where the action at the current state does not only affect the current reward but also subsequent rewards can be formalized using MDPs. MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made [SB18b]. MDPs is an environment in which all the states have the Markov property.

The Markov property states that the future states are independent of the past given the present. This statement can be mathematically expressed as;

$$P\big[S_{t+1}|S_t\big] = P\big[S_{t+1}|S_1, S_2, ..., S_t\big], \tag{2.1}$$

where S_t is the current state, S_{t+1} is the next state and P is the probability distribution for the given input. This equation means that the transition from S_t to S_{t+1} is entirely independent of the past.

An MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}$ is the state transition probability function,

$$\mathcal{P}_{ss'}^a = P\big[S_{t+1} = s'|S_t = s, A_t = a\big], \tag{2.2}$$

and the reward,

$$\mathcal{R}_s^a = \mathbb{E}\big[R_{t+1}|S_t = s, A_t = a\big], \tag{2.3}$$

with γ is the discount factor $\gamma \in [0, 1]$.

2.3.4 Formulation

An agent/vehicle observes the current state $s_t \in \mathcal{S}$ at each discrete time step t, and chooses an action $a_t \in \mathcal{A}$ based on a policy π, observes a reward signal r_t, and transitions to a new state s_{t+1}. The goal of the agent is to maximise the total future reward; $G_t := r_t + r_{t+1} + r_{t+2} ... + r_{t+n}$. The reward in future states e.g. r_{t+n} can not be guaranteed due to the stochasticity of the environment. Therefore a discount factor is introduced to determine the present value of the expected future rewards i.e. $G_t := r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} ... + \gamma^{n-t} r_{t+n}$. Then, the total reward can be represented as follows; $G_t := r_t + \gamma(r_{t+1} + \gamma(r_{t+2} ...)) = r_t + \gamma G_{t+1}$.

Discount Factor: It determines the impact of future rewards to the current immediate reward. It is necessary to avoid infinity as a reward in continuous tasks. If it has the value of 0 than there is no impact of future rewards to the current state. When it reaches to the value 1, the agent considers the future rewards when evaluating the current state. Having a large discount factor in a non-stationary environment might be problematic since the expected future rewards can not be guaranteed.

Policy

A policy π is a distribution over actions given states i.e. $\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$ in other words sequences of actions an agent takes, which fully defines the behaviour of an agent.

Action-Value Function

The action-value function $Q^\pi(s, a)$ is the expected total reward starting from the state s, taking action a and then following policy π,

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]. \tag{2.4}$$

The action-value function can be decomposed into immediate reward plus discounted value of successor state,

$$Q^\pi(s, a) = \mathbb{E}_\pi[r_t + \gamma Q^\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]. \tag{2.5}$$

The optimal action-value function, denoted as $Q^*(s, a)$ is the maximum action-value function over all policies,

$$Q^*(s, a) = \max_\pi Q^\pi(s, a). \tag{2.6}$$

Once the optimal action-value function is found, the optimal policy can be easily obtained,

$$\pi^*(s, a) = \begin{cases} 1, & \text{if } a = \arg\max_{a \in A} Q^*(s, a) \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

The Bellman Optimality Equation

The equation below says that the maximum future reward for this state and action is the immediate reward i.e. r_t plus a maximum expected future reward for the next state. If the agent acts optimal a' i.e. chooses the action that maximizes action-value function at

each state-action pair and visits all the state-action pairs frequently, it can calculate the optimal action-value function.

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[r_t + \gamma \max_{a'} Q^{*}(S_{t+1}, a') | S_t = s, A_t = a\right]. \tag{2.8}$$

The action-value function can be obtained by applying the Bellman equation as an iterative update [Ber00] i.e. $Q(s,a) = \mathbb{E}\big[r + \gamma \max_{a'} Q(s',a') | s,a\big]$. With such a value iteration algorithm, the optimal action-value function $Q \rightarrow Q^{*}$ can be obtained as the number of iterations reaches to infinity [SB18b]. Then, the optimal policy can be obtained easily from $Q^{*}(s,a)$ using the Equation 2.7.

Solving the Bellman optimality equation is non-trivial since the equation is non-linear and there is no closed-form solution. There exist many iterative solutions methods; value iteration, SARSA [RN94], and the most famous one is called Q-Learning.

Q-Learning

The main idea in Q-learning is to iteratively approximate the Q-function using the Bellman equation i.e. $Q(s,a) = r + \gamma \max_{a'} Q(s',a')$.

$$Q(s,a) = Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big], \tag{2.9}$$

where $\alpha \in [0,1)$ is the learning rate and s', a' are the next state and the selected action at the next state.

The action-value function $Q(s,a)$ is represented by a lookup table i.e. every state-action pair (s,a) has an entry $Q(s,a)$. Once the optimal state-value $Q^{*}(s,a)$is reached for each state and action pair, the agent takes *greedy* actions, i.e. always choose the action that maximizes its long term reward. However, during the training/iterations of the algorithm acting greedy always is problematic. What if there is another action at the current state which will provide a higher reward at the end? This phenomena in RL known as *exploration-exploitation* dilemma. The agent should explore all state-action pairs enough to obtain the optimal policy. This can be sustained by adding a random exploration, known as $\epsilon - greedy$ policy. The agent performs the random action with a probability ϵ, or greedy action $a_t = \arg\max_{a_t \in A} Q(s_t, a_t)$ with a probability of $1 - \epsilon$ at time step t where $\epsilon \in [0,1]$. At the beginning of the training ϵ value starts from a high number e.g. 0.99 and decreases through the episodes. This would ensure that the agent has explored enough state-action pairs before acting greedy [SB18b].

2.3.5 Q-Learning with Function Approximation

In practice, the tabular method is not computationally efficient and realistic for large state and action spaces. For example, the backgammon consists of $\approx 10^{20}$ states, and the game of Go has $\approx 10^{170}$ different states [Tro16]. It would take so much time for the algorithm to learn the optimal behaviour and most of the states will not be visited, the algorithm would be stuck at the local optimal. Therefore, value approximation methods e.g. linear value function approximation(LVFA), neural networks are required to approximate the value of state-action pairs. The most common one among them is the neural networks, which is preferred due to capabilities of capturing non-linearities of the given input.

2.3.6 Deep Neural Networks(DNN)

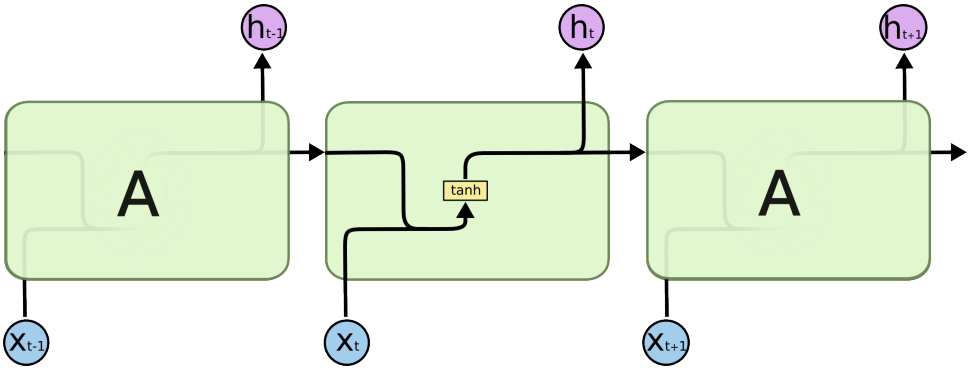
In many real problems, a number of states and actions are very high, thus classical tabular method to store the value of each state-action pairs is impractical. LVFA is one of the solutions to approximate value function but it requires to define good features to represent input data. Extracting the feature from the data to diminish its dimension requires domain expertise and in many cases, it is difficult and time-consuming. Deep learning techniques are one of the best solutions to extract meaningful features from the high dimensional data without requiring expert knowledge and interference [BCV13]. The DNN as a function approximator of value function $Q(s, a)$, can be represented as $Q(s, a; \theta)$ with θ are the parameters of the neural networks. However, learning the parameters θ which represents the value function, requires an immense amount of computational resources [LBBH98]. Fortunately, the development of the parallel processing technologies e.g. GPU or tensor processing units (TPU) alleviates this burden. In the following, recurrent neural networks(RNN) which are a special type of deep learning technique is explained.

2.3.7 Recurrent Neural Networks(RNN)

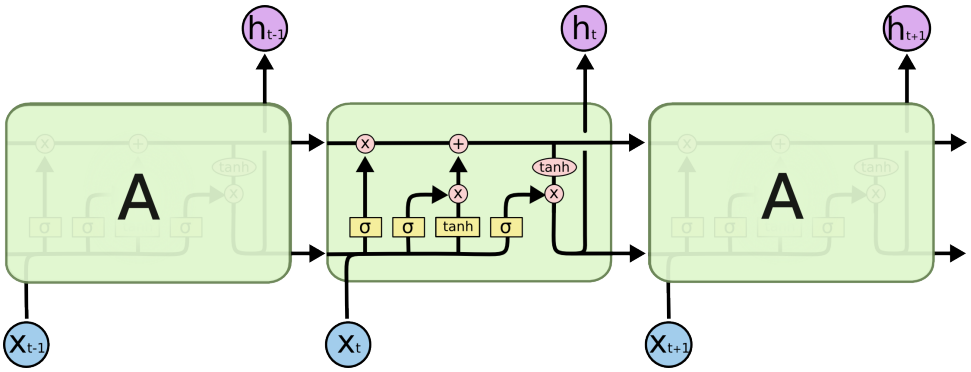
RNNs are enhanced neural network structure with memory capabilities i.e. can keep information about previous data, that can be exploited for applications like speech recognition which has temporal and sequential data [DHK13]. Vanilla RNNs suffer from vanishing gradient problem, the magnitude of error gradient vanishes during training which makes it difficult for RNN to learn long-term dependencies [BSF94]. Simply, prediction error is calculated based on the output of RNN and the gradient of the error is computed using back propagation process. Gradient is calculated by continuous multiplications of derivatives in this process. The value of the derivative can be also so small, and this continuous multiplications can lead the gradient so called "vanish" **??**. Some variants of RNN such as Long Short Term Memory networks, usually called as LSTMs are a special type of RNN which are capable of learning long-term relations in data by solving vanishing gradient problem via special "gates" [HS97]. The Figure 2.6 illustrates the difference between simple RNN and LSTMs.

The structure of one LSTM unit can be shown as in Figure 2.7. An LSTM unit decides which information is added or removed to the previous cell state C_{t-1}, supervised by structures called gates. Each unit consists of three gates namely, forget, input and output gates to protect and control the cell state. The forget gate receives the previous cell output together with new input and determines which information should be forgotten. So that, it determines which information will be excluded from the previous received cell state C_{t-1}. The input gate decides which information will be updated in the new cell state C_t. Finally, the output layer decides which parts of the new cell state C_t will be part of the output value. The calculations are summarized in Equations 2.10, 2.11.

$$\begin{aligned} f_t &= \sigma(W_f.[h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i.[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_c.[h_{t-1}, x_t] + b_c) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t, \end{aligned} \tag{2.10}$$



(a) The repeating module in a standard RNN contains a single layer



(b) The repeating module in an LSTM

Figure 2.6: Comparison between simple RNN and LSTM[Ola15]

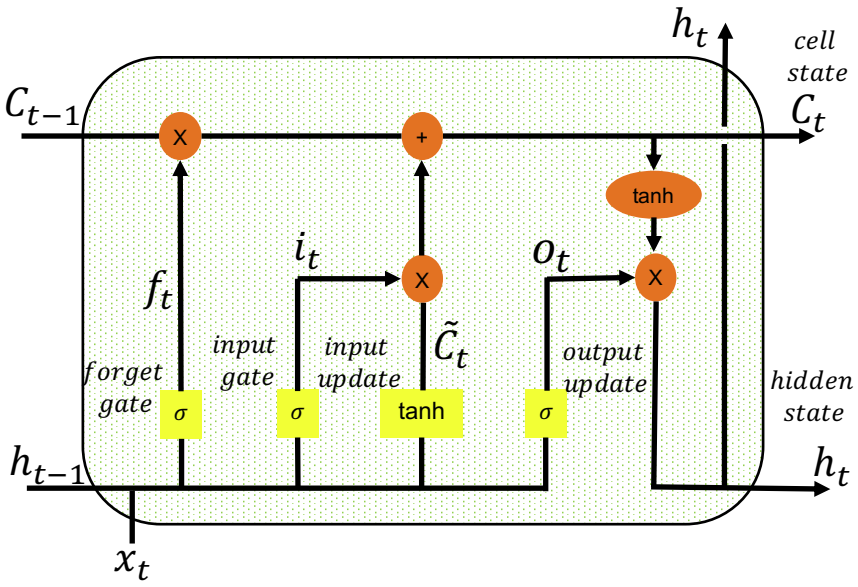


Figure 2.7: Structure of one LSTM unit

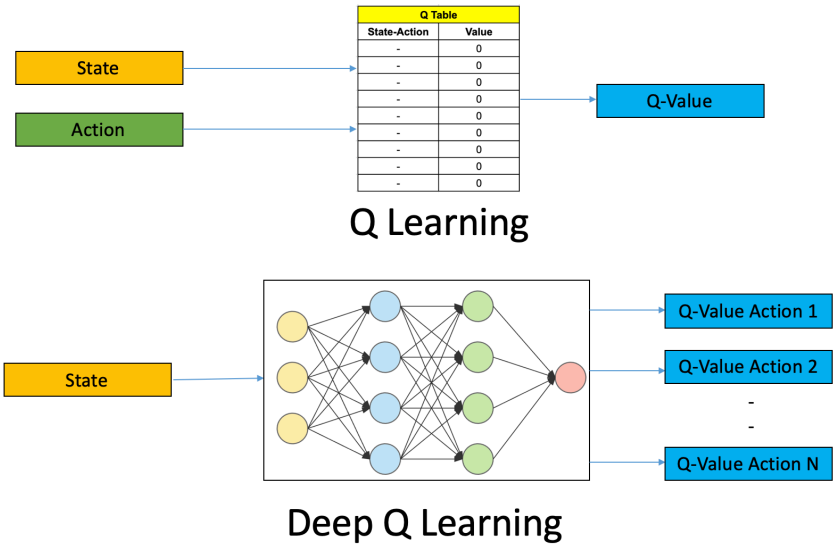


Figure 2.8: Comparison between tabular method and neural networks

where σ is the sigmoid layer which outputs between zero and one, determines how much of each component should be let through. After the new cell state is calculated, the output of the LSTM unit which is a filtered version of the cell state is derived.

$$\begin{aligned} o_t &= \sigma(W_o.[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh C_t. \end{aligned} \tag{2.11}$$

2.3.8 Deep Q Networks(DQN)

In DQN, neural networks are used to approximate the Q-value function [MKS+15]. The neural network takes the state as an input and output a Q-value for each possible action (Figure 2.8). Instead of calculating each state and action value through, neural networks have an additional output layer to calculate the value of each action for the given state. This simple trick reduces the amount of computation required to calculate the values of each state-action pairs. So, instead of running calculations in neural networks for $|\mathcal{A}|$ times for each action $a \in \mathcal{A}$ in action space $\mathcal{A}$, one just increases the size of output layer to $|\mathcal{A}|$ and calculates all the values at one time.

The value function can be represented as $Q(s, a; \theta)$ where θ denotes the trainable weights of the network. Then, the Bellman equation becomes $Q(s, a; \theta) = r + \gamma \max_{a'} Q(s', a'; \theta)$ and

$$Q(s, a; \theta) = Q(s, a; \theta) + \alpha\big[r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)\big].$$

The term inside the bracket is called temporal difference(TD) error $\delta = r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta)$ and $r + \gamma \max_{a'} Q(s', a'; \theta)$ is known as TD target. The objective is to find the θ parameters of the neural network which approximate the optimal value of each state-action pair. If the TD error is zero, then the Q value is converged to its final value. The cost $L(\theta)$, current assumption $Q(s, a; \theta)$ for state-action value minus the assumption after receiving immediate reward for that action $r + \gamma \max_{a'} Q(s', a'; \theta)$, that we want to minimize can be written as follows;

$$L(\theta) = \big[(r + \gamma \max_{a'} Q(s', a'; \theta)) - Q(s, a; \theta)\big]^2. \tag{2.12}$$

The gradient descent is applied to the neural network parameters i.e. θ in order to minimize the cost.

Experience Replay

Experience replay is a concept in which the agent can benefit from the previous experience while performing gradient descent update to the parameters of the neural networks. With this technique, the correlations among experiences can be removed via sampling randomly from the replay memory and the changes in the data distribution smoothing over [MKS+15]. The agent stores the experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ at each time-step t in a data set $D_t = e_1, ..., e_t$ and during learning samples(or mini-batches) of experience from the data set $(s, a, r, s') \sim U(D)$. The loss function which minimized the TD error at each learning iteration i becomes;

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i) - Q(s, a; \theta_i) \right)^2 \right]. \tag{2.13}$$

In Equation 2.14, experiences i.e.(s, a, r, s') stored in the memory D are randomly selected. Experience replay memory enables the agent to benefit from the previous experiences in other word makes the learning sample efficient while stabilizing the learning updates.

Target Q-networks

Updating the parameters of the neural network by using the loss function in Equation 2.14 generates a problem, known as *moving Q-targets*. The first component of the TD error is the Q-target and is calculated as the immediate reward plus the discounted max Q-value for the next state. In the training phase, the parameters of the neural networks are updated based on the TD error and the same weights apply for the calculation of both Q-target value and Q value. So, the agent wants to reach that Q-target value but it also moves this value after each update since the same neural network is used to calculate both values. Deepmind suggested applying a target network as a separate network for the calculation of Q-target values [MKS+15], which also periodically updates with the current weights of the Q-networks. This technique reduces correlations with the target and provides stable training. Then, the loss function becomes;

$$L_i(\theta_i) = \mathbb{E}_{((s,a,r,s')\sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right], \tag{2.14}$$

where θ_i^- are the parameters of the network used to compute the target at iteration i.

Double Q-networks

Although target networks can fix the *moving Q-targets* problem, there is another problem in DQN called *overestimation of Q values*. In DQN, it is hard to ensure that the best action at the next state is the action with the highest Q-value. At the beginning of the training, the agent does not know the best action since it does not explore the state-action pairs enough. Therefore, taking always the maximum Q value which would be noisy as the best action makes the learning complicated. Another enhancement proposed by the paper [vHGS15] is to use two deep Q networks; one is responsible for the selection

of the next action, the other one is responsible for the evaluation of that action. With the technique, an unbiased expected target value can be obtained while also solving the *moving Q-targets* problem. In this thesis, we deployed this approach to utilize the training performance.

2.3.9 Deep Recurrent Q Networks(DRQN)

The convergence of the DQN algorithm is guaranteed under the MDP assumption i.e. next state of the agent only depends on the current state and action, of the given problem. However, multi-agent settings break the MDP assumption since the next observed state is highly dependent on the actions of the other agents in the vicinity. For example; if a user observes the signal strength of channels as a part of the state, then next state depends not only the action i.e. channel used the user but also the actions i.e. channels used by other users in the previous time-step. Therefore, the true state s_t is hidden and each agent observes only o_t which is correlated with s_t.

The recurrent neural networks can be exploited to alleviate the effects of partial observability by providing an internal state and combining the observations over time, leading to better policies in partially observed environments [HS15]. Instead of approximating the $Q(s_t, a_t; \theta)$ with feed-forward neural network, now we approximate $Q(o_t, a_t, h_{t-1}; \theta)$ with recurrent neural networks where o_t is the current observation, a_t is the current action and h_{t-1} is the hidden state of the agent at the previous step. This can be done by replacing the first feed-forward layer of the DQN with a fully connected Long-term-short-memory(LSTM) layer. The hidden state $h_t = LSTM(o_t, h_{t-1}) = LSTM(o_{t-(L-1)}, ..., o_t)$ where L is the number of observations that we consider, thus we now estimate $Q(h_t, a_t; \theta)$. In this work, we used the term DRQN and DQN with LSTM interchangeable.

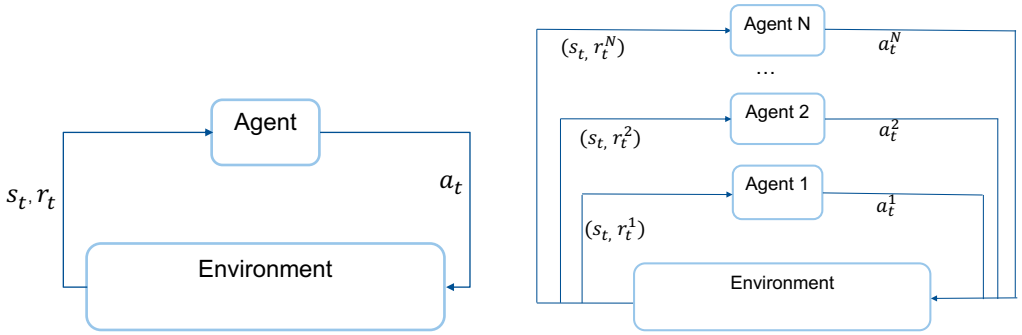
2.4 Multi-Agent Reinforcement Learning

Multi-agent RL(MARL) approach is also designed to cater to sequential decision-making problems in which more than one agent involved. However, It is harder to learn in multi-agent environments than single-agent environments due to the intrinsic complexity of the environment. In other words, all agents are part of the environment and each agent interacts not only with the environment but also with each other. This would invalidate the *Markov property* if observation of agents are affected by the actions of the other agents(the future dynamics, transitions, and rewards depend only on the current state) since the environment is no longer stationary [NVD12, TW12]. The environment is non-stationary from the perspective of agent because the other agents are also part of the learning e.g. there is no deterministic policy yet and actions of each agent can affect the reward of the other agents.

A common way to formalize MARL is to use stochastic/Markov games.

2.4.1 Stochastic/Markov Games

Markov games, also known as stochastic games are the direct generalization of MDP for multi-agent settings and it has been widely used to develop MARL algorithms as a framework after the seminal work [Lit94]. We adapt the formal definition which is introduced by the paper [ZYB19](definition 2.2).



(a) Single agent MDP (b) Multi agent, Markov game

Figure 2.9: Single agent MDP vs Multi-agent Markov game

Definition 2.4.1. A *Markov game* is defined by a tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \gamma)$, where $\mathcal{N} = \{1, ..., N\}$ denotes the set of $N > 1$ agents, $\mathcal{S}$ denotes the state space observed by all agents, $\mathcal{A}^i$ denotes the action space of agent i. Let $\mathcal{A} := \mathcal{A}^1 \times ... \times \mathcal{A}^N$, then $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ for any joint action $a \in \mathcal{A}; \mathcal{R}^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathrm{R}$ is the reward function that determines the immediate reward received by the agent for a transition from (s, a) to $s'; \gamma \in [0, 1]$ is the discount factor.

Note that in a Markov game, each agent can have individual action space and reward design. At each time-step t, each agent $i \in \mathcal{N}$ executes an action a_t^i based on the received environment state s_t. The joint action $\mathbf{a}_t := \{a_t^1, a_t^2, ..., a_t^N\}$ can be constructed after each time-step, and the environment alters to the next state s_{t+1} when the joint action is applied. The reward received by each agent i is calculated as a function of current state s_t, the joint action $\mathbf{a}_t$ and the next state s_{t+1} i.e. $\mathcal{R}^i(s_t, \mathbf{a}_t, s_{t+1})$. Each agent i wants to optimize its long-term reward with the policy $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ such that $a_t^i \sim \pi^i(.|s_t)$.

We can define a value function of states which represents how important of being at state $s \in \mathcal{S}$ for an agent to reach its goal. Some states have more impact to reach/maximize the goal of the agent. For example; in a maze problem in which the agent maximizes its reward when it finds the exit from the maze, being near states of the exit is more valuable than being at the beginning of the maze. In Markov games, the value-function $\mathcal{V}^i : \mathcal{S} \to \mathbb{R}$ if agent i becomes a function of the joint policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ defined as $\pi(a|s) := \prod_{i\in\mathcal{N}} \pi^i(a^i|s)$. Specifically, for any joint policy π and state $s \in \mathcal{S}$,

$$\mathcal{V}^i_{\pi^i,\pi^{-i}}(s) := \mathbb{E}\left[\sum_{t\geq 0} \gamma^t \mathcal{R}^i(s_t, \mathbf{a}_t, s_{t+1}) \middle| a_t^i \sim \pi^i(.|s_t), s_0 = s\right], \tag{2.15}$$

where $-i$ indicates the indices of all agents in $\mathcal{N} \setminus i$. Since the system is non-stationary for such games, it can not be modeled as MDP. The most famous solution method is Nash equilibrium(NE) which can be defined as follows [BO99];

Definition 2.4.2. A *Nash equilibrium* of the Markov game $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, \gamma)$ is a joint policy $\pi^* = (\pi^{1,*}, ..., \pi^{N,*})$, such that for any $s \in \mathcal{S}$, $i \in \mathcal{N}$ and π^i

$$\mathcal{V}^i_{\pi^{i,*},\pi^{-i,*}}(s) \geq \mathcal{V}^i_{\pi^i,\pi^{-i,*}}(s). \tag{2.16}$$

Equation 2.16 expresses that for any agent $i \in \mathcal{N}$, the policy $\pi^{i,*}$ is the best response to $\pi^{-i,*}$. In the Nash equilibrium, each agent's policy is optimal when considering the

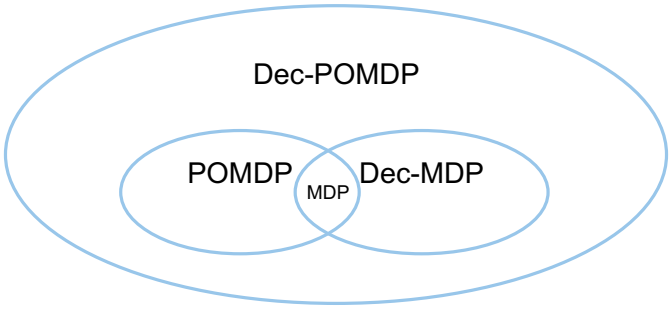


Figure 2.10: The MDP, POMDP, and Dec-MDP are the special cases of Dec-POMDP [BZI13]

policies of other agents. The purpose of the most MARL algorithms is to converge to such an equilibrium point.

As mentioned above, Markov game formulation consider the case where each agent has its own action space and reward design. In this thesis, we consider that the agents are *homogeneous* which means that all the agents share the same reward function i.e. $\mathcal{R}^1 = \mathcal{R}^2 = ... = \mathcal{R}^N = \mathcal{R}$ and same action space i.e. $\mathcal{A}^1 = \mathcal{A}^2 = ... = \mathcal{A}^N = \mathcal{A}$, set of actions available to the agents. We also consider *cooperative* settings, where agents collaborate with each other to maximize their shared goal. Furthermore, agents do not have any information about the global system state s_t, they only have local observations of the system state s_t at the time-step t. Thus, agents can *partially* observe the global system state s_t. Our designed system i.e. partially observable, cooperative multi-agent settings can also be formalized as *decentralized partially observable Markov decision processes*(Dec-POMDPs) [OSV08].

Dec-POMDPs

Dec-POMDPs consider that each agent performs an action based on only local observation which carries *partial* information about the global system state. Dec-POMDP has been known to be NEXP-complete due to the intractability caused by non-stationarity of the environment and partial observability [BGIZ02]. The Figure 2.10, show the relationship among different models.

Definition 2.4.3. A decentralized partially observable MDP (Dec-POMDP) is a tuple $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{\Omega^i\}_{i\in\mathcal{N}}, \mathcal{O}, \mathcal{R}, \gamma)$, where $\mathcal{N} = \{1, ..., N\}$ is a finite set of agents, $\mathcal{S}$ is a finite set of states, $\mathcal{A}^i$ is a finite set of actions available to the agent i, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ for any joint action $a \in \mathcal{A}$, ω^i is a finite set of observations available to agent i, $\mathcal{O}, \mathcal{R}$ are the observation, and reward models respectively and γ is the discount factor which determines the contribution of the horizon of the problem.

2.4.2 Training Approaches

Learning schemes for multi-agent reinforcement learning can be considered in three categories; centralized, concurrent and parameter sharing [GE].

Centralized

The centralized learning approach creates a joint model for both observations and actions of the agents. A centralized policy receives joint observations of the agents and generates joint actions. This approach is good for partial observability since policy can construct the global state from the joint observations of the agents. However, it is not applicable in reality since both training and execution require the observations of all agents in order to determine the joint actions. In other words, agents can not take any action based on just their local observation. Furthermore, the complexity of the model increases with the number of agents since the joint observation and action space increase.

Concurrent

In a concurrent learning approach, each agent learns its own individual policy. Each agent can exploit its local observations to map an action. Concurrent learning is useful when the agents are heterogeneous i.e. each has different purposes. However, this method suffers a lot from the non-stationarity of the environment. Each considers the other agents as a part of the environment, yet having an individual learning algorithm for each agent makes the environment non-stationary. Moreover, scaling to a large number of agents requires significant computational power since each agent has its own neural network to train.

Parameter Sharing

In parameter sharing approach, all the agents share the parameters of a single policy which allows the policy to be trained simultaneously with the different observations of each agent. The policy can be trained efficiently using parameter sharing if the agents are homogeneous [GE]. This approach is sample efficient and scalable to a large number of agents because the experiences of all agents utilize the training. Therefore, it is applied also in various applications [FNF+17, SSF16], including the wireless communication field [NMGM, NC17].

We adopt the parameter sharing approach in this work, simply because it is scalable, sample efficient and feasible to our system since all the vehicles are homogeneous i.e. share the same reward function, and action space.

2.5 State Of The Art

Distributed resource allocation in V2X communication has been standardized by the two key radio access technologies Dedicated Short Range Communications (DSRC) and Cellular-V2X. DSRC technology relies on 802.11p standard for medium access control layers which deploy Carrier Sense Multiple Access(CSMA) [HVS11]. C-V2X has developed the semi-persistent scheduling approach that is also part of the 5G-V2X mode 2 standard for resource scheduling in the absence of base stations. The studies so far show that C-V2X outperforms DSRC in terms of packet reception ratio, better non line-of-sight(NLOS) capabilities and required link budget [Res19b, GAA18]. SPS algorithm is based on the sensing of the received signal strength on available channels. Our analysis showed near vehicles are likely to choose the same resources since the results of the sensing would be similar due to the positional proximity of the vehicles [Res19b]. If the

vehicles choose to same resources for transmission, the collision will persist depending on *resource reselection counter* and *probability of keeping resource*. The details of SPS are mentioned in Section 2.2.2.

The collided vehicles can not detect and react to the collision due to half-duplex mode. This will lead to high information age within close vehicles which is for sure undesirable for the safety messages in V2X since the value of the information received by the vehicles in the vicinity is more crucial for safety. In this work, we developed a decentralized resource allocation algorithm which selects resources based on the positional distribution of the other vehicles observed by the transmitter vehicle.

The achievements of deep learning in various areas such as computer vision and natural language processing [Sch14, LBH15] pave the way for the development of reinforcement learning. In deep learning, neural networks are used to compress high dimensional data into compact representations. This enables reinforcement learning to be applied for high-dimensional state and action spaces by benefiting from deep learning to approximate the policy or value function. The studies on deep reinforcement learning(DRL) mainly focused on single-agent settings, where a single agent solves a static task. However, many problems e.g. autonomous vehicles, traffic light control, task and resource allocations, etc., involve more than one agent. In multi-agent settings, the agents are required to perform an action based on just their local observation in order to maximize their individual reward or overall goal by considering the presence of other agents in the environment [Foe18]. Thus, distributed resource allocation in V2X communication can also be modelled in multi-agent settings. A vehicle that chooses frequency and time resources for the transmission should consider the existence of the other vehicles in order to maximize its packet reception ratio or designed reward function. Multi-agent reinforcement learning has been applied in various problems to offer distributed solutions i.e. decisions based on local information [HSN17, GEK17, FFA+17, ZYC+17].

In the areas of communications and networking, DRL has been used as an emerging tool to effectively address various problems and challenges. In particular, modern networks such as Internet of Things (IoT), Heterogeneous Networks (HetNets), and Unmanned Aerial Vehicle (UAV) network become more decentralized, ad-hoc, and autonomous in nature. Network entities such as IoT devices, mobile users, and UAVs need to make local and autonomous decisions, e.g., spectrum access, data rate selection, transmit power control, and base station association. Some of the usage areas of DRL in communications and networking are network security, connectivity preservation, traffic engineering, resource scheduling, and data collection [LHG+18]. We refer the survey [LHG+18] for the interested people to learn more about the applications of DRL in communications and networking.

MARL has been recently exploited in modern network problems such as IoT, UAV, and V2X for spectrum access, data rate selection, transmit power control, etc. Each entity in those network demands autonomous decisions based on local observations in order to maximize the network objectives e.g. maximization of throughput or energy consumption utilization. The authors in [NMGM] proposed multi-agent DRL for distributed dynamic power allocation in base stations to maximize sum spectral efficiency. This problem has similarities with our problem. Base stations can not talk with each other but they also interfere. So they want to optimize the power levels to minimize "interferers" with each other while maximizing their throughput. They train only one DQN network (*parameter*

sharing) to be applied for each base station. Although they do not use LSTM networks to mitigate partial observability, they consider the last two measurements as a part of the state design. The paper [CLN19] deploys MARL approach with independent UAV agents(each agents considers other agent as a part of the environment) for resource allocation in UAV networks to maximize downlink throughput while considering interference among multiple UAVs.

DRL approaches for resource allocation in V2X communication is also adapted by many researchers in recent years. The authors in [SB18a] developed a resource allocation algorithm for the delimited out-of-coverage area(DOCA) deploying reinforcement learning scheduler used by the surrounding base stations of DOCA. The authors in [FBZ19] proposed Q-learning based resource management algorithm to maximize the sum rate of cellular users(C-UEs) and vehicle users(V-UEs) while satisfying the QoS requirements of safety V-UEs and minimizing the interference between V2V links and cellular links. In paper [YLJ19], the authors introduce DRL based decentralized resource allocation for both broadcast and unicast traffic. They considered V2V communication in the coverage of base station where each V2V link select transmission resource block and power to satisfy its latency constraint while minimizing the interference among V2V and V2I(vehicle-to-infrastructure) links. The same authors model the same problem above i.e. V2V and V2I coexistence with MARL [LYL19], specifically, using independent Q-learning where each agent learns a decentralized policy treating other agents as part of the environment. Our approach differs from them since we train only one policy to be applied for each vehicle. This makes our model sample efficient e.g. vehicles can benefit from each other's observation and computationally affordable for large scale e.g. optimizing each agent's DQN would require huge computation [GE]. In this work, we consider the case where there is no base station in the vicinity and vehicles need to develop a policy based on the local observation and piggybacking neighboring table from the other vehicles. When a transmitter vehicle receives a CAM message from above layers e.g. application, it determines which resources to use autonomously.

2.5.1 Dynamic Spectrum Access

There are some studies in the literature for dynamic spectrum access using reinforcement learning. The paper [ZLGV] proposes deep actor-critic(AC) reinforcement learning for dynamic multi-channel access for both single and multi-agent case. AC is a policy gradient method in which the agent utilizes the policy directly whereas DQN requires a value network to obtain a policy. In the multi-agent case, mentioned in the paper, each agent has its own AC network, acting other agents as a part of the environment. This would make the learning computationally expensive for large scale scenarios as mentioned above. Also, they did not consider the congestion case i.e. the number of vehicles is higher than the number of resources but in this work, our main focus is the behaviour of the proposed design in the congestion case. In an interesting study [NC17] the authors proposed MARL for distributed dynamic spectrum access. They used LSTM layer as an input to obtain internal state and aggregate observations over time, which is preferable for POMDP problems introduced by multi-agent design. They also tested the design with different utility functions(rewards), and proposed an efficient design principle. However, they do not consider varying channel conditions due to mobility which needs to be considered in V2X communication. Also, they assume that each user receives an acknowledgment message to construct an input state indicating whether its transmission was successful or

not. Sending feedback to the transmitter about its transmission would also create extra load on the limited resources. Our proposed model does not require a feedback whether the transmission was successful or not for state design to develop a policy for channel access, a vehicle just transmits positions of observed vehicles as a part of the CAM.

2.5.2 Position Based Scheduling

There are some studies in the literature based on position based scheduling for V2X communication. The paper [BKKF14] proposes a centralized resource allocation mechanism based on the location of vehicles. In [ABPS16] authors proposed a distributed resource allocation approach based on the location information of each vehicle for V2V communication. The resource blocks(RBs) are assigned into different clusters based on the the positions and load similarity of V2V links. Their proposal considers only unicast communication, however, broadcast communication for safety messages is vital and therefore standardized for V2X communication. Our approach differs from these two position based scheduling approaches since we proposed a distributed resource scheduling mechanism for broadcast safety messages.

Chapter 3

Algorithm Design

In this chapter, the details and reasons behind the proposed algorithm are explained. We start with the problem definition and formalize distributed resource allocation for synchronized traffic. We allocate additional sections to explain the details of state design, action space and reward design of the distributed MARL scheduler. The neural network architecture for the approximation of the state-action pair values is clarified.

We consider available radio resources for the transmission of cooperative awareness messages(CAMs) are divided into two parts based on the direction of the vehicles. Thus, the performance of the proposed approach can be analyzed separately for each direction. Vehicles generate CAM messages periodically to broadcast to all other vehicles in the vicinity. Each CAM message carries a neighbor table of the transmitter along with the other specific information such as the position and velocity of the vehicle. CAM messages allocate resources e.g. resource blocks in both time and frequency. When we define the action space, we let each discrete action to represent one resource block in which a CAM message can fit. The design of the system model for MARL scheduler is crucial for reinforcement learning tasks. In general, the system model includes the definitions of state space, action space and reward design. We additionally explain the view-based observation mechanism and piggybacking neighbor table to construct the state of an agent. Let $\mathcal{N} = \{1, 2, ..., N\}$ denote the number of vehicles in the scenario and $\mathcal{K} = \{1, 2, ..., K\}$ is the number of available resources. At each time-step t, each vehicle i receives a state s_t^i which consist of local observations and selects an action u_t^i from the shared policy i.e. $\pi(u_t^i|s_t^i)$ and receives reward $r_t^i(u_t^i, s_t^i)$. Note that, each agent receives an action individually based on its local observation e.g. not as a joint action. The aim of the agent is to maximize the discounted return $R_t = \sum_{l=0}^{H} \sum_{i=0}^{N} \gamma^l r_{t+l}^i(u_{t+l}^i, s_{t+l}^i)$ from episodes of length H. If more than one vehicle choose the same action e.g. resource for transmission, then receivers can decode the message of transmitter which has highest signal strength.

3.1 Problem Definition

Let $\mathcal{N} = \{1, 2, ..., N\}$ denote the number of vehicles in the scenario and $\mathcal{K} = \{1, 2, ..., K\}$ is the number of available resources blocks e.g. subframes in our case. We consider a fully synchronized time-slotted system i.e. all the vehicles receive CAM messages at the same time from the application layer. In this work, it is desired to maintain high reliability of the reception of CAM safety messages. Thus, we used the packet recep-

tion ratio(PRR), the ratio of successful reception among the total number of the vehicle transmitter neighbors, as a key performance indicator.

Each vehicle i with $i \in \mathcal{N}$ selects an action which indicates the resource for the transmission at each time step t such that

$$a_t^i \in \{1, 2, ..., K\}, \tag{3.1}$$

then, all the actions selected by the vehicles in the scenario at the time stamp t is as follows;

$$\mathbf{a_t} = (a_t^1, a_t^2, ..., a_t^N), \tag{3.2}$$

The PRR for the user i at timestamp t is calculated as a function of $\mathbf{a_t}$;

$$\begin{aligned} PRR_t^i(\mathbf{a_t}) &= \frac{1}{N_t^i} \sum_{j=1}^{N_t^i} 1\{\gamma_t^{ij} \geq \gamma^{th}\}, \\ \gamma_t^{ij} &= \frac{P|H_t^{ij}|^2}{\sigma^2 + \sum_{k \in N_t^c(\mathbf{a_t} \setminus \{i\})} P|H_t^k|^2}, \end{aligned} \tag{3.3}$$

where N_t^i denotes the number of the vehicles neighbors of the i^{th} vehicle at the time t, γ_t^{ij} is the receiving signal-to-interference-plus-noise ratio(SINR) of the j^{th} vehicle among N_t^i, γ^{th} is the SINR threshold, P is the fixed transmitter power of the vehicles, σ^2 is the power of additive white Gaussian noise, $N_t^c(\mathbf{a_t})$ is the number of collided vehicles determined by the actions at the time stamp t, H_t^{ij} and H_t^k are the channel gain between the transmitter i and receiver j, interference channel gain of the transmitter k respectively. Simply, $PRR_t^i(\mathbf{a_t})$ is the number of vehicles in the vicinity of the transmitter i that can decode the message divided by the number of vehicles in the vicinity. Whether a vehicle can decode the message or not determined by the value of γ_t^{ij}.

We are interested in maximizing PRR in the system. Specifically, the distributed resource allocation mechanism can be formulated as

$$\begin{aligned} \underset{\mathbf{a_t}}{\text{maximize}} \quad & \sum_{i=1}^{N} PRR_t^i(\mathbf{a_t}) \\ \text{subject to} \quad & a_t^i \in \{1, 2, ..., K\}, \\ & i = 1, 2, ..., N. \end{aligned} \tag{3.4}$$

Hence, the distributed resource allocation problem in the form of Equation 3.4 at the beginning of each timestamp t. For $N \leq K$, the Equation 3.4 can be satisfied easily if all the vehicles choose separate resources for transmission. However, in the congestion case i.e. $N > K$ obtaining the optimal policy $\mathbf{a}_t^*$ is challenging depending on the interference among vehicles i.e., which vehicles exploit the same resources for transmission. In this work, we assumed that PRR can be maximized if the far vehicles choose the same resources. In this way, the problem of SPS algorithm i.e. near vehicles are likely to choose the same resources can be also avoided. We designed the reward function in order to motivate this behavior.

In this work, we assumed that PRR can be maximized if the far vehicles choose to same resources in the case of congestion so that the interference among the vehicles that choose the same resources can be diminished.

3.2 State Design

The state vector s_t^i at timestamp t for the user $i \in 1, 2, ..., N$ includes the previous action that user i take at the previous time stamp i.e. a_{t-1}^i and the output of the function $v_t^i = f(p_t^i, B, R)$. The previous action $a_{t-1}^i \in \mathcal{A}$ is a one-hot vector with size $|\mathcal{A}|$ i.e. the action index is "1", others are "0".

The function $f(p_t^i, B, R)$ is called view-based positional distribution(VPD) function which receives the neighbor table p_t^i of the user i at the time stamp t, an integer $B \in \mathbb{Z}^+$ which determines the granularity of the view-based observation vector and an integer $R \in \mathbb{Z}^+$ which indicates the observation radius of the user i. Each vehicle holds a neighboring table which includes the positions of the other vehicles on the road. Also, each vehicle piggybacks the neighboring table together with the CAM messages so that every vehicle on the road knows the positions of the other vehicles. Scalability can be maintained by limiting the size of the neighboring table(e.g. each vehicle shares the positions of the nearest m users). The details of the update mechanism are explained in the next subsection.

The function $f(p_t^i, B, R)$ sorts the distance between the transmitter and receiver those within the range R in ascending order(if the position of the observed vehicle on the x-axis is less than the observer position, distance is multiplied by -1). Then, each vehicle observes underlying distributions from its view/perspective with simple histogram technique. The Figure 3.1 illustrates the output of the $f(p_t^i, B, R)$ function for the given positional distribution of the vehicles p_t^i, $B = 10$ and $R = 100m$. The details of the function are described in the pseudo-code Algorithm 1. Then, the state vector becomes $s_t^i = (a_{t-1}^i, v_t^i)$ for the user i at time stamp t with the total size of $|\mathcal{A}| + B$. Each entry e in the neighboring table p_t^i has the vehicle id, positions, sequence number for updates and counter that holds $lastUpdate$ of this entry.

Algorithm 1 View-based Positional Distribution

procedure F($p_t^i, B, R, lastUpdateThreshold, txID$)
  Initialize empty dist vector D
  for entry e in p_t^i **do**
    if e["$lastUpdate$"] < $lastUpdateThreshold$ **then** ▷ Always consider updated vehicles
      $dist = calculate\_dist(e$["pos"], p_t^i["$txID$"]["pos"])
      if $|dist| < R$ **then** ▷ If the receiver is in the range of the transmitter vehicle
        $D.push(dist)$
      end if
    end if
  end for
  $D = D.sort()$ ▷ Sort the distance in ascending order
  $H = histogram(D, B, [-R, R])$
  $v_t^{txID} = H/len(D)$ ▷ Calculate the distribution from histogram
  return v_t^{txID}
end procedure

The details of the neighbor table updates and the reason of the design choose of VPD are explained in the next section. Furthermore, we rationalize why VPD is exploited as

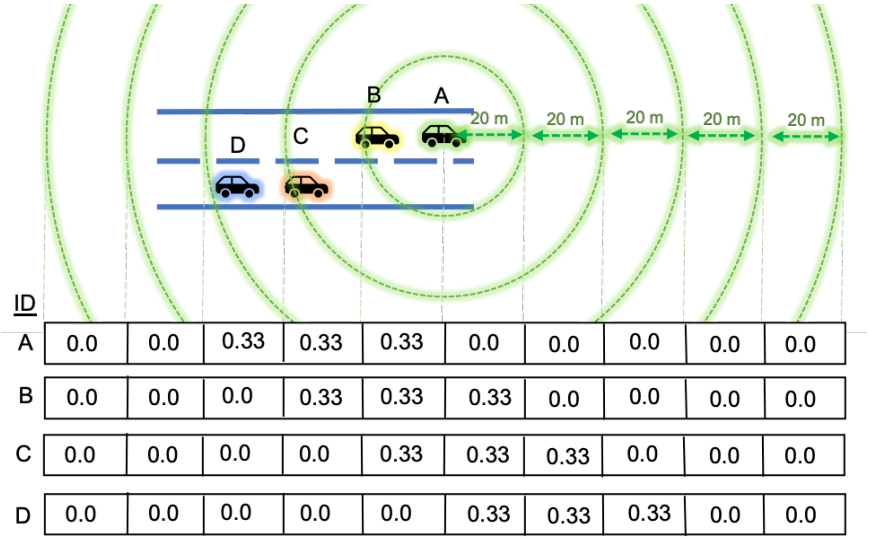


Figure 3.1: Example of the output of $f(p_t^i, B, R)$ with $B = 10$ and $R = 100m$

a part of the state.

3.2.1 Neighbor Table Updates

View-based positional distribution is a novel approach to be used as a part of the state input of the model. Each vehicle holds a neighboring table which includes the positions of the other vehicles on the road. Each user piggybacks the neighboring table together with the CAM messages so that every vehicle on the road knows the positions of the other vehicles. We adopt a similar mechanism to keep the neighboring table updated as a destination-sequenced-distance vector(DSDV) routing scheme which is based on the Bellman-Ford algorithm [PB94]. Each entry in the neighbor table contains a sequence number along with vehicle id and positions, which is updated(increased by "1") by the vehicle of the entry for every transmitted CAM message. When a vehicle receives a CAM message from the other vehicles, it checks every entry in the neighbor table of the transmitter in the CAM message. If the sequence number in an entry of the transmitter neighbor table is higher than the sequence number of the same entry(indicates the same vehicles), the receiver updates the positions of the vehicle in the entry with the positions received by transmitter neighbor table.

Additionally, each entry in the table includes a *last update* counter which expresses the time between the updates of this entry. When a vehicle piggybacks its neighbor table together with a CAM message, it increases the *last update* counter in all the entries except its own entry. When a vehicle receives a neighbor table of the transmitter which has a higher sequence number for the same entry, it reset the *last update* counter to zero. If the *last update* counter is higher than a threshold which can be adjusted based on mobility e.g. 2 seconds, then we omit this entry when extracting positional distribution from the table assuming that the receiver vehicle is no longer in the coverage of the transmitter vehicle. Note that, we apply such a mechanism so that all vehicles on the road can know the updated positions of the other vehicles. We refer to the paper [PB94] for readers to get more intuitions about the mechanism.

3.2.2 View-based Positional Distribution

We refer to the paper [dWFF+18] for the formulation. All vehicles have a circular field of view with a fixed radius(set based on communication range). Then, common knowledge $\mathcal{I}^{\mathcal{G}}$ between groups of $\mathcal{G}$ of agents arises through entity-based field-of-view common knowledge[dWFF+18].

The state s is composed of a number of entities $e \in \varepsilon$ with state features s^e i.e. $s = \{s^e | e \in \varepsilon\}$. In this work, all entities are agents $a \in \mathcal{A} \equiv \varepsilon$ and s^a is the position of vehicles i.e. $s^a = (x^a, y^a)$. The observation z^a contains the subset of state features s^e from all the entities e that a can see. Whether a can see e is determined by the binary mask $\mu^a(s^a, s^e) \in \{\bot, \top\}$. In our scenario, binary mask checks whether the distance between s^a, s^e is lower than the observation radius R. Also, μ^a and R are the same for all agents. The set of all entities the agent a can see is therefore $\mathcal{M}^a := \{e | \mu^a(s^a, s^e)\} \subseteq \varepsilon$. The agent's observation is specified by the deterministic observation function $o(s, a)$ such that $z^a = o(s, a) = \{s^e | e \in \mathcal{M}^a\} \in \mathcal{Z}$. Then, each agent receives z^a e.g. positions of the observed vehicles to create *view-based positional distribution* vector such that $v^a = g(z^a, B, R)$.

In our scenario, agents are homogeneous i.e. the function g, binary mask μ^a and variables B, R are the same for all agents. Furthermore, our aim is to learn cooperative behavior in centralized training e.g. far vehicles should learn to select the same action/resource. Thus, the model can learn the common knowledge function $\mathcal{I}^{\mathcal{G}}$.

Then, the commonly known trajectory τ_t^G;

$$\tau_t^{\mathcal{G}} := \mathcal{I}^{\mathcal{G}}(\tau_t^a) = \mathcal{I}^{\mathcal{G}}(\tau_t^{\overline{a}}), \forall a, \overline{a} \in \mathcal{G}. \tag{3.5}$$

From the observation trajectory, $\tau_t^a = (v_1^a, ..., v_t^a)$ of any agent $a \in \mathcal{G}$. The commonly known position distribution $v_k^{\mathcal{G}} = \{g(z_k^{\mathcal{G}}, B, R) | z_k^{\mathcal{G}} \in \mathcal{Z}, B \in \mathbb{Z}^+, R \in \mathbb{Z}^+\}$ with $z_k^{\mathcal{G}} = \{s_k^e | e \in \mathcal{I}^{\mathcal{G}}\}$. The commonly known trajectory

$$\begin{aligned} \tau_t^{\mathcal{G}} &= (v_1^{\mathcal{G}}, \mathbf{u}_1^{\mathcal{G}}, ..., v_t^{\mathcal{G}}, \mathbf{u}_t^{\mathcal{G}}) \\ \mathbf{u}_t^{\mathcal{G}} &:= (u_t^a, ..., u_t^a), a \in \mathcal{G} \\ \text{with policy } &\pi^{\mathcal{G}}(u_t^a | \tau_t^{\mathcal{G}}, \tau_t^a). \end{aligned} \tag{3.6}$$

Each agent can deduce the commonly known positional distribution vector from its view-based positional distribution and individual action for each agent can be obtained by condition on commonly known positional distribution and view-based position distribution Equation 3.6. Simply, if all agents in the system can see each other, then an agent can estimate the observations of the others from its own observation and selects an action by estimating the actions of others to maximize the system objective. Each agent view is the group's joint view, joint actions are generated based on this joint view. Each agent performs its part from the joint action which is determined by the view of the agent.

3.3 Action Space

In general, $\mathcal{A} := \{a | a = (t, \mathbf{f}), t \in \mathbb{T}, \mathbf{f} \in \mathbb{F}\}$, where t and $\mathbf{f}$ are available transmission time slots and frequencies respectively. In this thesis, the agent uses all the available chunks with transmission time interval $TTI = 0.5ms$. For example; the action space becomes

Vehicle id	Position x	Position y	Sequence number	Last update
1	12	8	35	0
2	43	9	34	1
3	123	10	34	1
4	53	11	33	2

Table 3.1: Example of neighbor table

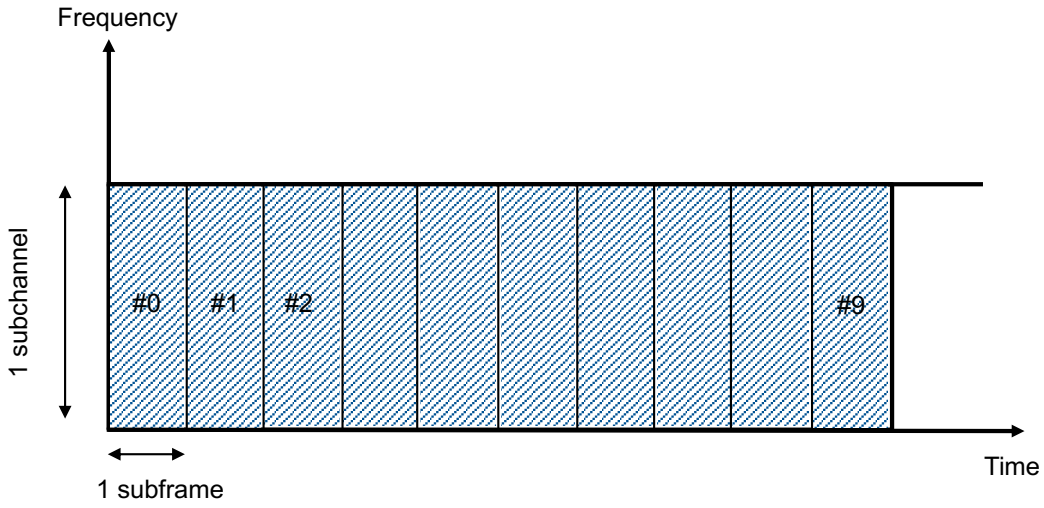


Figure 3.2: Illustration of resource pool for action space $|\mathcal{A}| = 10$

$\mathcal{A} := \{a|a = s, s \in \mathbb{T}\}$ where $s \in \mathbb{T} := 1, 2, ..., 10$ for $delay\_budget = 5ms$. Hence, $|\mathcal{A}| = 10$. The illustration of resource pool can be seen in Figure 3.2.

3.4 Reward Design

The purpose of each agent is to deliver its CAM message to the other agent, therefore agents should learn how to avoid collision with other agents. In this work, we consider the performance of the proposed approach also in congestion case. *We assume that the system performance can be improved in the congestion if far vehicles on the road exploit the same resources.* Therefore, the reward design should carry this responsibility to obtain cooperative behavior. Let $N = \{1, ...n\}$ denote the total number of users in the scenario. The reward of each agent is calculated as follows;

$$r_t^i(a_t^i|s_t^i) = \begin{cases} 1, & N_c^i = 1 \\ \left.\begin{matrix} 0 & \text{if } dist(\boldsymbol{c}^o) > R \\ -N_c^i & \text{else,} \end{matrix}\right\} & N_c^i = 2 \\ -N_c^i, & N_c^i > 2 \end{cases} \tag{3.7}$$

where the vector $\boldsymbol{c}^o$ is the collided users at resource o i.e. $\boldsymbol{c}^o = [i, k, ..., l]$ with $i, k, ..., l \in N$ and $N_c^i = |\boldsymbol{c}^o|$ is the total number of collided users. The average of the sum rewards at the time stamp t is added to the reward of the individual users. Our observation reveals that adding the averaging of reward motivates cooperative behaviour. Let $r_t = (r_t^1, r_t^2, ..., r_t^N)$,

the reward of the each user;

$$r_t^i(a_t^i|s_t^i) = r_t^i(a_t^i|s_t^i) + \frac{\sum_{j=1}^{N} r_t^j(a_t^j|s_t^j)}{N}. \tag{3.8}$$

To sum up, if more users exploit the same resources for their transmission they receive a higher negative penalty. If the two users that are not within their range use the same resources then we motivate them by applying neither negative nor positive reward.

3.5 Deep Neural Network Approximation

The algorithm is required to explore all state-action combinations in order to converge the optimal desired policy. In this work, the number of state-action pairs is huge since the state consist of a continuous positional density for each discrete observation range. The values of each state-action pair can not be stored in the tabular method. Therefore, we should use function approximation methods such as linear value function approximation(LVFA) or neural networks. We prefer neural networks since they can capture a pattern in the non-linear behaviour of data. Furthermore, some special architectures of neural networks e.g. Convolutional neural networks(CNN), recurrent neural networks can be exploited for unique applications e.g. image processing, time-series prediction etc.

In this work, we use Long-Term-Short-Memory(LSTM) architecture as a first hidden layer in order to avoid partial observability and predict the velocity from the changing positional distribution through time to estimate the future positional distribution. The LSTM layer is connected to a feedforward neural network layer. Then, the system is connected to the output layer with K number of neurons so that for a given state, the model gives values for each action $a \in \mathcal{A}$. We adopt Adam optimizer with a learning rate 10^4 for gradient-descent optimization. The details of the neural network architecture will be revisited in the next chapter.

Chapter 4

Implementation

In this chapter, the details of the implemented work are explained. The main purpose of this chapter is to provide the reader an understanding of how and with which methods the background information and algorithm design are merged with the goal of this thesis. This is ensured by starting with the explanation of the algorithm for multi-agent reinforcement learning. The simulator environment and the software tools to realize the proposed idea are described. Then, the previously explained semi-persistent scheduling algorithm is revisited and implementation details are mentioned. Note that, we implement the state of the art SPS algorithm for a fair comparison with the proposed approach.

4.1 Distributed MARL Scheduler

The computation of the optimal resource allocation in Equation 3.4 is a combinatorial optimization problem and mathematically intractable as the number of vehicles in the scenario increases. In this section, we propose multi-agent reinforcement learning algorithm for distributed resource allocation problems with each vehicle as an agent. The algorithm is trained to find the optimal action in Equation 3.4 for each vehicle by formulating the problem under a specific reward function as explained in the previous Section 3.4.

4.1.1 Architecture

In this part, the details of the designed deep neural network(DNN) structure of the DRQN algorithm are explained. We train only one network which is shared by all the other agents. The DNN consists of three parts; input layer, hidden layers, and output layer.

- *Input Layer*: The input s_t^i of the DRQN is a vector of size $|\mathcal{A}| + B$, where $|\mathcal{A}|$ is the size of the action space which indicates the number of available resources(subframes). Each vehicle adds the action taken in the previous timestamp a_{t-1}^i as a part of the state. B is the number of intervals that an agent divides its observation view which depicts the granularity of the observation and fixed for all vehicles.
- *Hidden Layers*: The first layer is the LSTM layer. A multi-agent learning environ-

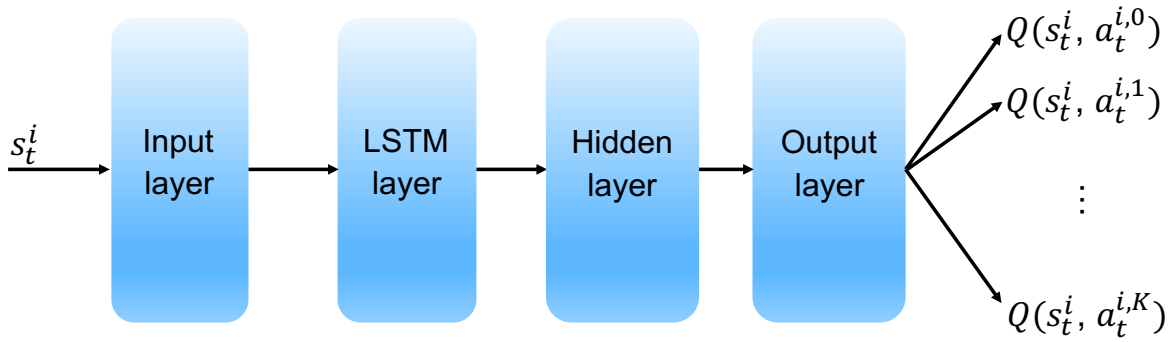


Figure 4.1: Deep neural network structure

ment breaks the MDP assumption since the next observed state of the agent is not only determined by the agent itself but also the action taken by the other vehicles. In our work, we do not observe the channel selection of the other vehicles since we use the positional distribution of the vehicle. However, partial observation is caused by the positional distribution of the other vehicles which is based on the unknown mobility pattern of the other vehicles. Several studies [LLZC18, FLZ+18, SKS16] show that LSTM networks can be deployed successfully for mobility prediction. LSTM layer sustains an internal state and combines the observations over time. LSTM layer is connected with a fully connected feed-forward network layer.

- *Output Layers*: The output of the DRQN is a vector with size $|\mathcal{A}|$ indicates the value of each action for the given state input. If the policy is greedy, vehicles choose the action with the highest value.
- *Double Q-learning*: The details of the technique are explained at Section 2.3.8. The performance of the algorithm degrades if a vehicle chooses and evaluates the action with the same network due to overestimation. Thus, double Q-learning is preferred to decouple the action selection and evaluation.

The deep neural network architecture can be illustrated as Figure 4.1.

4.1.2 Centralized Training Decentralized Execution

We adopt the centralized training, decentralized execution framework which is commonly used in many multi-agent reinforcement learning works [NMGM, NC17, FNF+17, LZG+17]. Agents still need to take actions based on only their local observations but in the centralized training part, we can access the actions of the vehicles and determine the reward based on the distance between them. This approach can also be applied in real-world settings such that the final policy can be trained in the simulator, later deployed in the real world. Vehicles then take a decision only with their local observations without requiring any other information used to train in centralized training. In this work, the parameters of the network are shared by all the other agents since the agents are homogeneous i.e. same reward function. Hence, each agent can benefit the experiences of the other agents in the centralized training phase. Note that, each agent is able to perform different actions because of the different observation histories.

As depicted in Figure 4.2, agent i takes an action a_t^i at timestamp t as a function of the current observation s_t^i based on the current policy. All agents are synchronized and

take their actions at the same time. We apply ϵ-greedy approach for the action selection starting from a high number of ϵ e.g. 0.99 and updating ϵ after every episode. Agents take a random action at the beginning of the training with a higher number of ϵ value which diminishes throughout the episodes.

After executing the action, agents observe the next state s^i_{t+1} and reward of that action r^i_t. The experiences e.g. $(s^i_t, a^i_t, r^i_t, s^i_{t+1})$ of the agents are stored in the experience replay buffer [MKS+15] to be used for the training. Note that, we only have one experience replay which contains experiences of all agents. Combining experience replay with the parameter sharing approach allows agents to learn the optimal policy faster. The experience replay memory is a FIFO buffer with a size proportional to the number of agents in the scenario. A new experience replaces the oldest experience in the replay buffer. As seen in Figure 4.2, the trainer selects a number of experiences uniformly from the buffer. Then, it calculates the loss through the target and evaluation networks. We apply Adam optimizer [KB14] to minimize the loss to update the parameters of the evaluation network which is used both for action selection e.g. policy and evaluation. The target network is updated with the parameters of evaluation networks once per E episodes. The details of the algorithm are represented in the pseudo-code Algorithm 2.

Algorithm 2 DRQN algorithm with experience replay

procedure MA-DRQN$(M, T, C, K, N\gamma)$
Initialize replay memory $\mathcal{D}$ with memory size C
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with random weights $\hat{\theta} = \theta$
for episode $e = 1, ..., M$ **do**
for $t = 1, ..., T$ **do**
for each vehicle $i = 1, N$ **do**
get state vector s^i_t
with probability ϵ select a random action a^i_t
otherwise select $a^i_t = \arg\max_{a^i_t \in A} Q(s^i_t, a^i_t; \theta)$
end for
Observe the all received reward $r_t = (r^1_t, r^2_t, ..., r^N_t)$
Store (s_t, a_t, r_t) into memory $\mathcal{D}$
end for
Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from $\mathcal{D}$
Calculate y_t as

$$y_t = \begin{cases} r_t, & \text{if episodes ends at } t+1 \\ r_t + \gamma Q(s_{t+1}, \arg\max_{a'} \hat{Q}(s_t, a'; \hat{\theta}), \theta) & \text{otherwise} \end{cases}$$

Perform a gradient descent step on $(y_t - Q(s_t, a_t; \theta))^2$ with respect to θ
Every K episodes set $\hat{\theta} = \theta$
Update ϵ
end for
end procedure

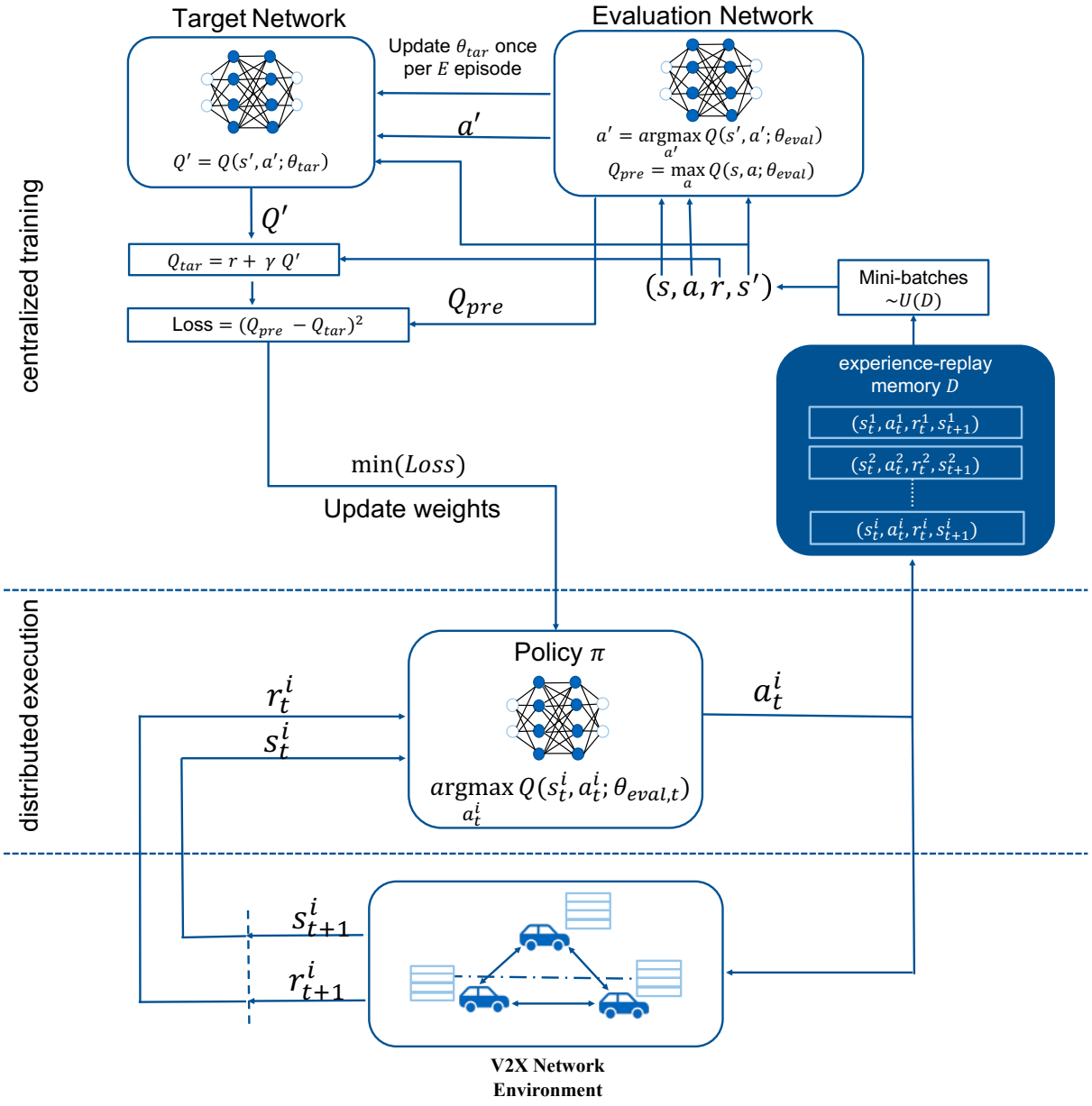


Figure 4.2: Illustration of the proposed algorithm

4.2 Environment Setup

In this section, the details of the simulator and software tools to realize the proposed distributed multi-agent reinforcement learning model are explained. We first analyze the performance of the algorithm in a toy example for efficient training to test various experiment settings. Then, the proposed approach is implemented to the real-time network simulator(RealNeS) for comprehensive analysis e.g packet reception ratio, age of information, etc.

RealNeS has a 5G-V2X protocol stack including PDCP, RLC, MAC and PHY layers for V2X traffic radio bearers. Scheduling of the packets is performed in the MAC layer. Whenever the MAC layer of a vehicle receives CAM messages from the upper layer, it requests a resource for transmission of the packet from the MARL agent. The communication between the agent and vehicle is realized through the python interface, explained in the previous section. During the training one agent is shared for all the vehicles in the network so that each can benefit from the experiences of each other. After the training i.e. decentralized execution, the same trained model is stored in the MAC layer of each vehicle. Vehicles decide which resources to use based on their local information and each decision cooperates with the other vehicles to maximize the system objective.

4.2.1 Test Simulator

A test simulator is required for faster analysis of various experiments. Once the proper architecture is determined in the test simulator, it is deployed and tested in the real simulator for elaborated evaluations such as packet reception ratio and information age. We assume that all vehicles are synchronized i.e. they receive the packets from the upper layer at the same time so that we can have a fair comparison of the performance of the proposed approach. At each time-step actions i.e. resource blocks of the vehicles are determined by the MARL agent based on local information at the vehicles. The actions are executed and each vehicle receives an individual reward based on action. Then, the average reward is calculated and is added to the individual rewards of the vehicles.

If there is more than one vehicle that exploits the same resource block, receivers decode the packets of closer vehicle. Also, we do not require a channel model for calculations of the rewards since we only check the distances among vehicles along with the actions to determine individual rewards. These simplifications speed up the learning process and allow us to test many different ideas. We also transferred the model trained in the test simulator to the real-time network simulator and proved that the trained model also performs better in terms of packet reception ratio than SPS.

4.2.2 Real Simulator

The simulator RealNeS of the company Nomor is used to simulate the system-level performance of the developed algorithm for 5G-V2X communication. The simulation granularity allows us to compute effective SINRs in the equivalent complex baseband (ECB) taking into account (codebook and non-codebook based) precoding and receive filtering techniques (MRC, LMMSE, LMMSE-IRC). Also, the simulation has PRB resolution for the calculations so that if multiple users transmit with the same resources e.g. PRBs, this will be considered in the error model. Simulator speed varies with the size of the

scenario ranging from above realtime for small scenarios and significantly below realtime for large scenarios with thousands of UEs in the simulation. RealNeS can simulate LTE (incl. eMBMS and various modes of LTE-V2X, 5G-V2X), 802.11, and 5G networks in frequency bands ranging from 700MHz up to 100GHz. The latest version of RealNeS supports new numerology and frame structure, packet duplication, massive MIMO, and beamforming features.

In LTE-V2X, the subcarrier spacing is fixed to 15kHz and subcarriers are used in groups of 12(i.e. 180kHz); in the time domain, 14 symbols from a subframe of 1 ms in other works TTI. The number of available resource blocks in a subframe for the transmission of 300 byte CAM message varies based on the modulation and coding schemes(MCS). The paper [BCM+19] suggests MCS 6 (QPSK, coding rate = 0.48) to fit two packets in one 1 TTI for highway scenario. However, having 2 packets using different frequencies in the same subframe(TTI) causes in-band emission(IBE). Studies in [LL15] shows that IBE degrades the system performance in V2X communication due to the interference between the subchannels in the same subframe. Therefore, we used shortened subframe length to TTI=0.5ms by increasing subcarrier frequency to 30kHz and transmit one packet per subframe. We can transmit the same number of packets in 1 ms as LTE-V2X by avoiding IBE.

The simulator has an interface to the SUMO simulator to simulate realistic mobility in RealNeS. Using the rich feature set of SUMO™ it is possible to either define own geometries of mobility paths or to import real-world data e.g. from OpenStreetMap [KHFW02]. For test purposes, we create a highway scenario in the SUMO simulator and read the mobility traces of the vehicles into the simulator and then the channel model. The visualization of the vehicles can be seen in Figure 4.3. Further information about the simulator can be obtained in this link[1].

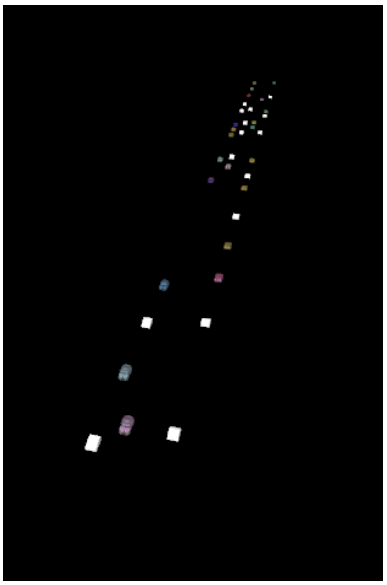
4.2.3 Software

In this work, we used python programming language for the development of the multi-agent reinforcement learning framework to test the algorithms as fast and as efficiently as possible. The simulation of the company "RealNeS" is written in the C++ language. Therefore, we create a python interface to communicate with the RealNeS simulator inspired by the paper [GZ19]. The messaging on the interfaces is realized over ZMQ sockets[2] using the Protocol Buffers[3] to interact with the simulator from the python process. The implementation details are depicted in Figure 4.4.

Python agent is required for the selection of action for the given input state. Then, the action is sent to the environment proxy which translates the action into the message and sends it towards an environment gateway over ZMQ sockets. Environment gateway is responsible for the initialization of the environment in the RealNeS. It informs the python agent about the *delay budget*, which determines the action space of the agent and number of vehicles in the scenario. Furthermore, it receives a neighbor table from the vehicles to be used to create a state input in the python agent. When a gateway receives the selected action from the python agent, it informs the MAC layer to prepare a transmission grant

[1] http://nomor.de/services/simulation/system-level-simulation/
[2] https://zeromq.org/
[3] https://developers.google.com/protocol-buffers/

(a) Visualization of the vehicles in 2D



(b) Visualization of the vehicles in 3D

Figure 4.3: Visualization of the vehicles with mobility in RealNeS

for the transmission. Simply, each vehicle in the simulator send their observations to the python agent and receive an action determined by this agent.

The advantages of this design choice can be summarized as follows;

- *Language*: Python programming is faster to implement and test new ideas than the C++ language. The development of machine learning frameworks is mostly based on the python language. Once the model parameters have been decided then the implementation can be realized in C++.
- *Debugging*: This is one of the biggest advantages of this design. The messaging between the python agent and RealNeS is realized by the REQ/REP pattern. This means that when a vehicle asks for an action simulation stops until it receives a reply from the agent. Therefore, we can stop the simulation in the agent to observe the training procedure in detail. This is very crucial for reinforcement learning design since there exist many variables in the system, one needs to *debug* in order to understand ensure the system reliability.
- *Dockerized agent*: Docker is an open platform for developing, shipping and running applications. Docker provides the ability package and runs an application in a loosely isolated environment called a container[4]. In this design, we can dockerize the agent independently from the simulator environment. This would allow us to use the computational resources efficiently. For example; we can run the simulation on one computer while having an agent in a separate computer that has higher

[4]https://www.docker.com

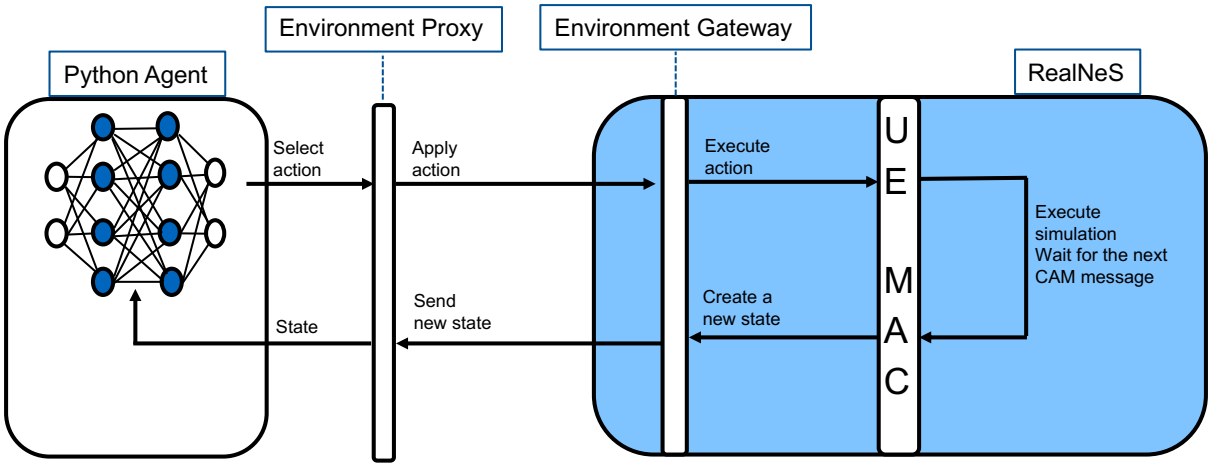


Figure 4.4: Illustration of communication through python interface

Library	Purpose	Version
Numpy[1]	scientific computing	1.16.5
matplotlib[2]	plotting	2.2.4
tensorflow[3]	deep neural network	1.14.0
pyzmq[4]	python interface	18.1.0
protobuf[5]	serializing structured data	3.9.1
pandas[6]	data analysis	0.24.2
PyYAML[7]	scenario scripts	5.1.2

Table 4.1: Python libraries for designing python agent

[1] https://numpy.org/; [2] https://matplotlib.org/;
[3] https://www.tensorflow.org/;
[4] https://pyzmq.readthedocs.io/en/latest/;
[5] https://developers.google.com/protocol-buffers;
[6] https://pandas.pydata.org/;
[7] https://pyyaml.org/

GPU capabilities and they are communicating through the python interface. This increases the speed of training.

Libraries

In this work, we benefit from some of the open-source software libraries for the implementation and visualization of the results. The proposed architecture in Subsection 4.1 is created by TensorFlow [AAB+15] which is an open source platform for machine learning applications. TensorFlow is basically a software library for numerical computation using data flow graphs. The libraries that are used in this work are listed in Table 4.1.

Chapter 5

Results

In this chapter, the performance of the proposed algorithm design and implementation is evaluated. We first start with the training performance of the approach based on the reward metric. Then, we analyze the performance at the RealNeS and compare the result with the standardized SPS approach.

5.1 Training Performance

In this section, we compared the performance of the proposed approach in various scenarios. Table 5.1 summarizes the main settings of the scenarios. We refer to the first scenario as a toy example. Each vehicle has different fixed velocities and moving towards the same direction as depicted in Figure 5.1. Each time-step in the simulator corresponds to $100ms$ in the real system since the periodicity of the CAMs is $100ms$. For example; $1.5m/step$ means that the speed of the vehicle is $54kmph$. When a vehicle reaches the edge of the highway, it just wraps around.

We trained the model with $\epsilon - greedy$ policy for 250000 time-step and the decay value of epsilon is 0.9992. After 200000 time-steps, agents take only greedy actions i.e. takes the action which has the highest state-action value. The granularity of the observations is $B = 40$ with $R = 100m$ i.e. 2.5 per bin. Hyperparameters for training are summarized in Table 5.2. Although numerous DNN models were tested, in this thesis only the best performer, thus advised, DNN model structure is elucidated.

Instead of giving reward 0 when 2 collided vehicles that are not within the range of R, we give 0 when the farthest vehicles choose the same resource in order to understand the

No	Vehicles	Resources	Highway length	Velocity	Mobility
1	4	3	100m	{18,36,45,54}kmph	Wrap-around
2	8	10	500m	∼35kmph	SUMO
3	10	10	500m	∼35kmph	SUMO
4	12	10	500m	∼35kmph	SUMO
5	12	10	500m	∼100kmph	SUMO
6	20	20	500m	∼35kmph	SUMO

Table 5.1: Scenario settings

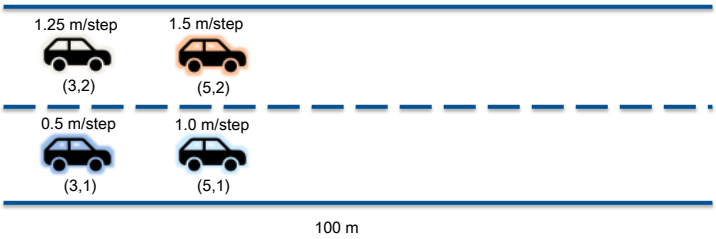


Figure 5.1: N=4, K=3, t=0, moving right direction

Parameter	**Value**
Time-steps	250000
Experience-replay memory	1024
Step size(for LSTM)	6
Batch size(Training)	512
Learning rate	0.0001
Discount factor γ	0.7
LSTM layer	256 neurons
Hidden layer	256 neurons
Target update	200 time-steps

Table 5.2: Training Hyperparameters

behaviour of the algorithm better. Note that, our goal is to teach far vehicles to use the same resources in the congestion case. So, the reward function becomes;

$$r_t^i(a_t^i, s_t^i) = \begin{cases} 1, & N_c^i = 1 \\ \left.\begin{matrix} 0 \\ -N_c^i \end{matrix}\right\} & \begin{matrix} \text{if } dist(i,k) \text{ farthest} \\ \text{else,} \end{matrix} \quad N_c^i = 2 \\ -N_c^i, & N_c^i > 2 \end{cases} \tag{5.1}$$

Also, the vehicles should act cooperatively. Therefore, we add the average sum reward in each time-step to the individual rewards i.e. Equation 3.8.

For each time-step, the maximum reward that vehicles can obtain is 2, 2 vehicles use separate resources $2 \times (r_t^i = 1)$ and the other two far vehicles choose the same resource $2 \times (r_t^i = 0)$. Thus, the maximum reward that vehicles get in each episode is $50 = 25 \times 2$. The results in Figure 5.2 shows that the proposed approach can reach very close to the optimal policy.

The granularity of the view-based positional distribution is adjusted to $B = 100$ for the second, third, fourth scenario and $B = 200$ for the fifth scenario with the observation range of $R = 500m$. The training results of the other scenarios are depicted in Figure 5.3. As seen in results, the proposed multi-agent reinforcement learning approach is able to evolve/develop a policy for the desired behaviour based on view-based positional distribution.

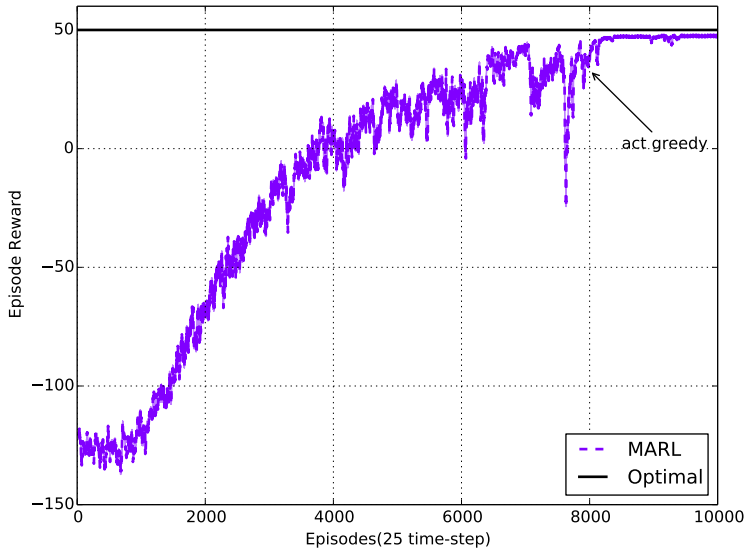
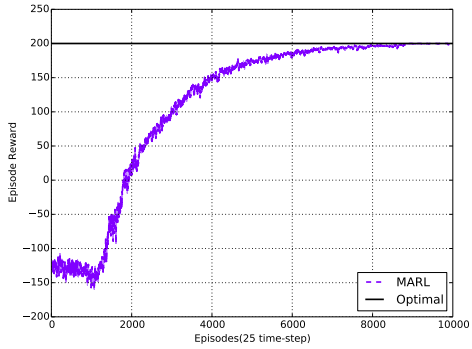
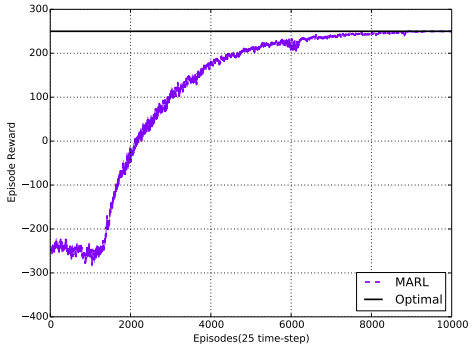


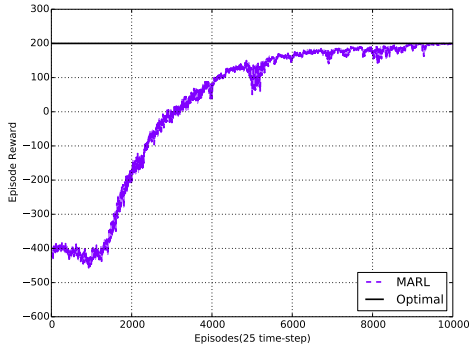
Figure 5.2: Toy example training results



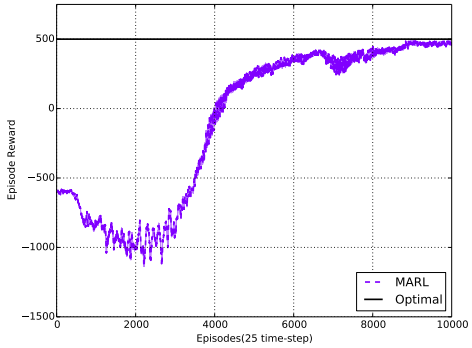
(a) Scenario 2: 8 vehicles, 10 resources



(b) Scenario 3: 10 vehicles, 10 resources



(c) Scenario 4: 12 vehicles, 10 resources, 35kmph



(d) Scenario 6: 20 vehicles, 20 resources

Figure 5.3: Training performance of different scenarios

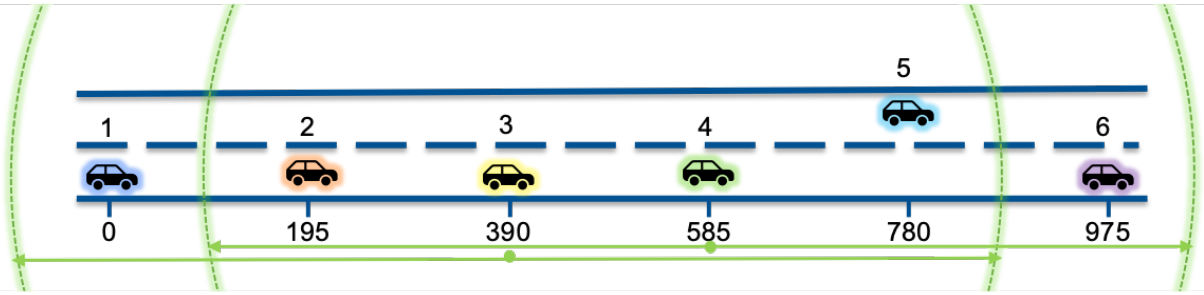


Figure 5.4: 6 Vehicles and 5 resources scenario

5.2 Discussions

In this section, the impact of design parameters e.g. bin size, discount factor on performance is investigated. Furthermore, the proposed algorithm is compared with normal DQN by replacing the first LSTM layer with a feedforward neural network. Training performance under random velocity is explored. At last, the convergence of the proposed method is analyzed from a common knowledge reinforcement learning perspective.

5.2.1 Why Reinforcement Learning?

What exactly the algorithm learns? Do we really need reinforcement learning? In this discussion, we try to answer these questions and show why we need RL with an example. After various evaluations and analyses, we observed that the policy converges to a joint action set based on the positional distribution of vehicles. This means that the resource used by the far vehicles varies and near vehicles also use various resources based on the positions of other vehicles. Simply, the policy generates joint actions based on the positional distribution of vehicles on road and each vehicle executes its action based on its view-based positional distribution. Then, do we really need reinforcement learning? Can not we just allocate resources statically based on the observed view-based positional distribution? For example; consider Figure 3.1 with 4 vehicles and 3 available resources. We can map the observations into specific resources such that A and D use the same resources. However, when the observations of vehicles are the same, then the vehicles use the same resources.

Consider the following example where 6 vehicles are distributed uniformly on the road and there are 5 resources to use for transmission. The aim is to schedule the vehicle 1 and vehicle 6 to the same resource and others to the different resources. The vehicles are homogeneous e.g. same $B = 20$ and $R = 500m$. Thus, vehicle 3 and vehicle 4 have the same view-based positional distribution. Therefore, in a static design vehicle 3 and vehicle 4 would choose the same resources.

We trained the MARL scheduler for this static scenario to understand whether vehicle 3 and vehicle 4 learn to use separate resources for transmission. Note that, in state design we have additionally previously taken action as a part of the state i.e. $s_t^i = (a_{t-1}^i, v_t^i)$. The results in Figure 5.5 indicates that the MARL scheduler is able to perform the optimal action i.e. vehicle 1 and 6 use the same resource and others use separate resources.

The MARL scheduler proposed in this thesis is able to interpret the optimal action not only from the view-based positional observation but also a previously taken action. Thus,

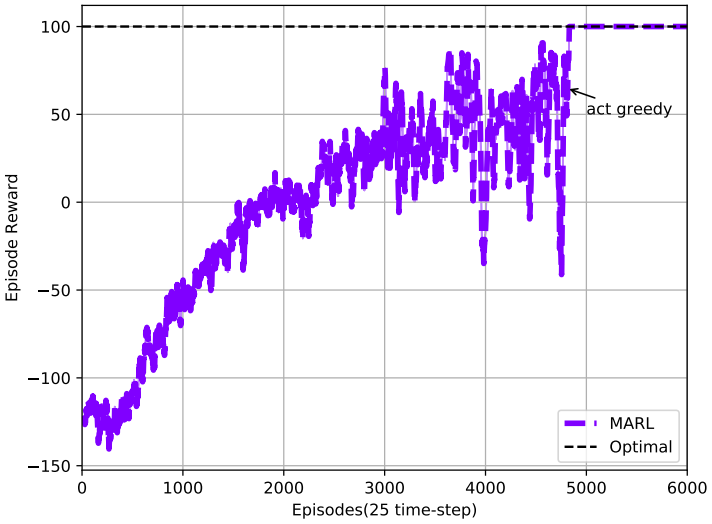


Figure 5.5: 6 Vehicles, 5 resources, training performance

it also works in the case where the view-based positional observations of the vehicles are the same.

5.2.2 Discount Factor

Discount factor γ determines the influence of future expected rewards when updating the value of state-action pairs as in Equation 5.2. When the discount factor is high, value updates rely mostly on future expected rewards instead of currently received reward r_t^i.

$$\begin{aligned} Q(s,a) &= Q(s,a) + \alpha\big[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\big] \\ TD\_error &= r + \gamma \max_{a'} Q(s',a') - Q(s,a) \\ TD\_target &= r + \gamma \max_{a'} Q(s',a'). \end{aligned} \tag{5.2}$$

Figure 5.6 shows the results of various discount factor γ for the toy example. The algorithm converges to the optimal policy with a lower discount factor whereas a higher discount factor degrades learning performance. In this work, the reward is highly associated with the actions of the other agents. Thus, relying on expected future rewards during learning(agents take random actions) can introduce estimation bias so that agents can not discover all state-action variations enough to develop the optimal policy. Therefore, for the rest of the experiments, the discount factor is selected as 0.7 since it performs better than other values as depicted in Figure 5.6.

5.2.3 Granularity of Observations

The performance of the proposed approach under different granularity i.e. B differs is also examined. We observed that the optimal bin size B is correlated with the number of vehicles in a scenario. Training performance of toy example and scenario 6 under various B value is depicted in Figure 5.7. In general, the variance of the learning curve diminishes with increasing granularity.

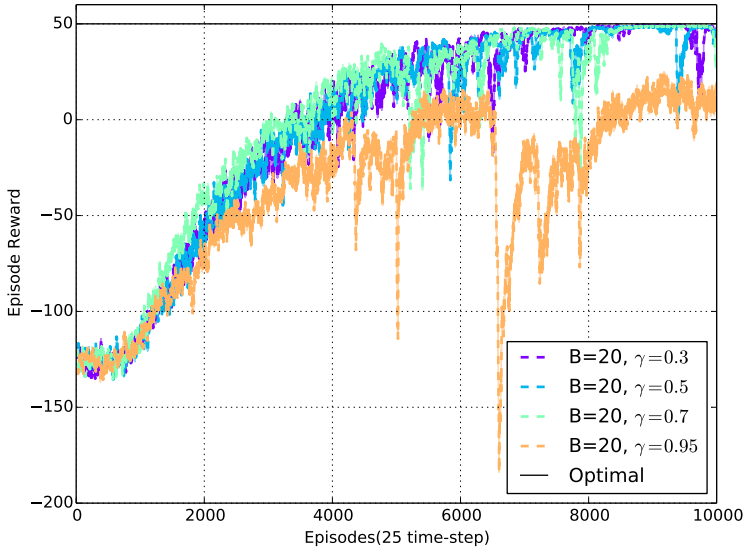
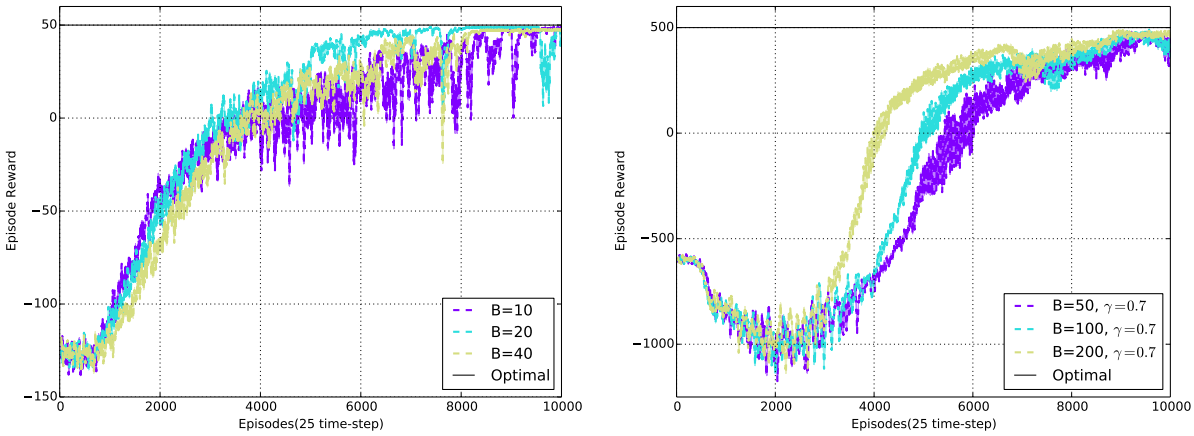


Figure 5.6: Scenario 1 with various γ



(a) Scenario 1 (b) Scenario 5

Figure 5.7: Training performance of various granularity

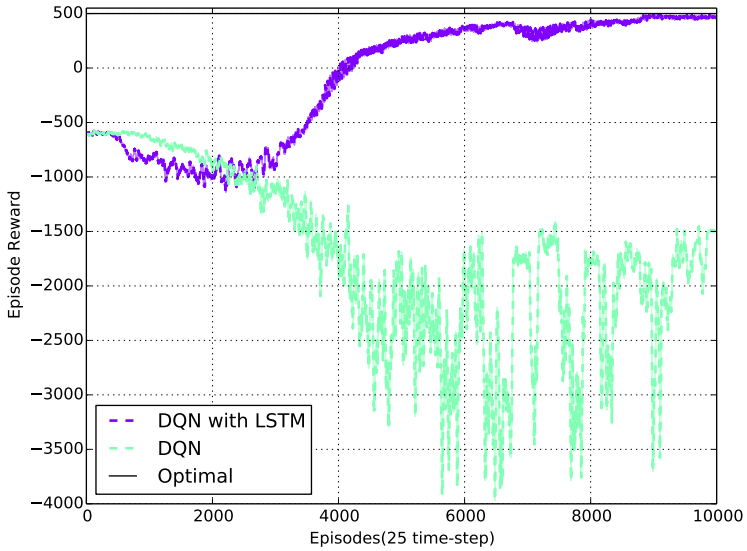


Figure 5.8: Scenario 6, Vanilla DQN vs DQN with LSTM

5.2.4 Impact of LSTM

We also analyzed the impact of the LSTM input layer on learning performance. Hence, the first LSTM layer is replaced by feedforward neural networks as vanilla DQN. Scenario 5 is exploited for comparing the performance of these two settings. The results in Figure 5.8 show that LSTM input layer enables learning. LSTM layer captures the previous m observations instead of learning only from the last observation. The details of LSTM networks are explained in detail in Section 2.3.7.

5.2.5 Random Velocity

In all of the experiments, the positional distribution of the vehicles on the road is finite and expected. After the training, the model can generate one joint action for each positional distribution and each vehicle can apply its part of the action based on their view-based positional observation. We tested the model with the toy example also under random velocities. At the end of the episode, each vehicle either accelerate(+0.1) or decelerate(-0.1) i.e. $\min(0.8, vel_t^i \pm \delta), \max(1.4, vel_t^i \pm \delta)$. The result in Figure 5.9 shows that the algorithm can not reach the optimal policy but vehicles can still learn a sub-optimal policy(2 vehicles use separate resources, 2 same resources). However, the results can be improved further by integrating velocity as a part of the state feature $v_t^i \in s_t^i$.

5.2.6 Higher Velocity

We also evaluate the performance of the proposed algorithm for a higher velocity of vehicles as in scenario 5 in Table 5.1. We exploit the same hyper-parameter settings as other scenarios. The training result in Figure 5.10 indicates that the algorithm is also able to learn in higher velocity settings but it can not reach the optimal desired policy. However, training performance can be improved further with a higher number of training steps since training is not saturated with the fixed value yet. Also, using the higher granularity for the observations e.g. $B = 400$ may improve the performance further.

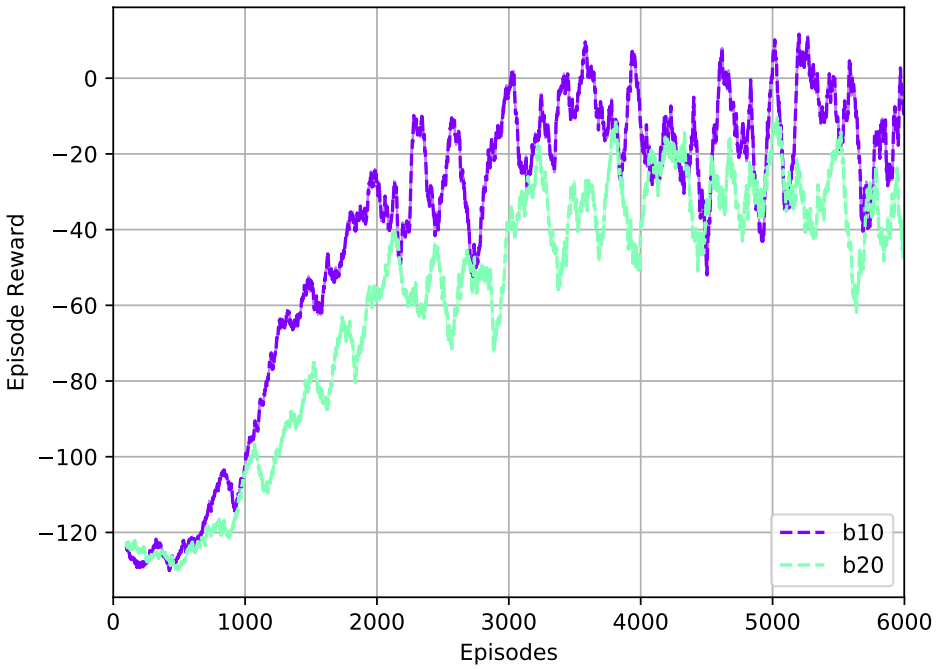


Figure 5.9: Scenario 1, random velocity

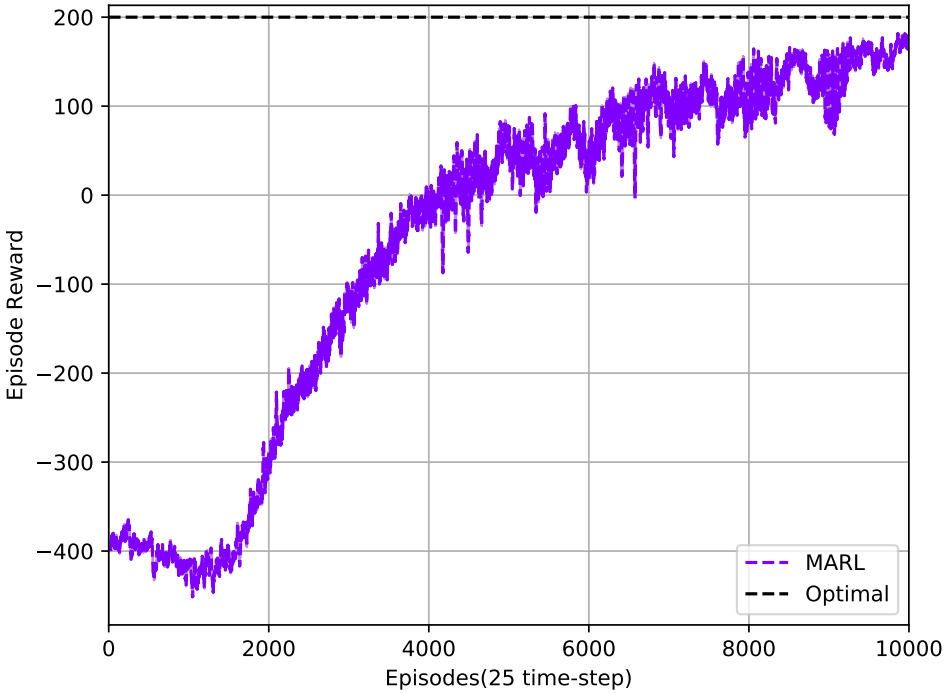


Figure 5.10: Scenario 5: 12 Vehicles, 10 resources, 100kmph

No	Vehicles	Resources	Highway length	Velocity	Mobility
1	6	5	250m	∼35kmph	SUMO
2	12	10	500m	∼35kmph	SUMO
3	12	10	500m	∼100kmph	SUMO

Table 5.3: Realness scenario settings

5.2.7 Common Knowledge Learning

In this work, observations of the agent that construct common knowledge are not *directly* affected by the actions of the agents. Unless all use the same resource persistently for transmission, then agents can not receive the messages of others due to half-duplexity constraint. Then, agents remove the positions of other agents from their neighboring table if they can not receive any update for the last m transmissions from other agents. Thus, the action of agent a_t^i has only a long term effect on common knowledge. In this work, common knowledge mainly depends on the system dynamics i.e. mobility of vehicles. This can simplify the problem since learning system dynamics(LSTM helps for mobility prediction) easier than developing common knowledge(otherwise, we would need to know the actions of other agents). Then, we might say that relying on common knowledge which is deducted by the information that is *not* directly affected by the actions of the agents, can improve cooperative behaviour and allow the model to develop individual actions instead of joint actions based on local observations. Statement: Then, the policy $\pi^{\mathcal{G}}(u_t^a|\tau_t^{\mathcal{G}}, \tau_t^a)$ can converge the optimal cooperative policy, if and only if u_t^a does not affect directly the observation-trajectory τ_t^a which construct common knowledge. If the statement above is true, then generating common knowledge under this condition can be a convergence metric for cooperative behavior in multi-agent systems.

5.2.8 Networked Agents

Our work can also be considered as a part of the networked agents in multi-agent reinforcement learning system in which agents exchange messages through a communication channel in order to maximize system objective. Communication in multi-agent reinforcement learning is a new emerging trend for cooperative and competitive settings [FAdFW16, ZYL+18, KMH+19]. In our designed model, each agent share its neighboring positional table with the other agents to keep updated position information of each vehicle in the system. This positional information is later used to construct state of the agent. Note that, sharing information with other vehicles to assist their resource selection is also part of the 5G-V2X i.e.mode 2(b) standard [R1118].

5.3 RealNeS Analysis

The performance of the algorithm is investigated in RealNeS in terms of packet reception ratio(PRR) and information age(IA) for congestion scenarios in Table 5.3. The trained policy in the test simulator is deployed to RealNeS and the performance is compared with random scheduling and standardized distributed resource allocation algorithm i.e. SPS.

5.3.1 Semi-Persistent Scheduling

SPS is implemented into the RealNeS simulator in order to compare the performance of the proposed distributed scheduling in this work with the conventional distributed scheduling approach which is also part of the 5G-V2X standard. In SPS, radio resources are selected from the *selection window* which is set by the given *delay budget*. Each vehicle has the same synchronization of time for sensing e.g. time corresponds to the same subframe. Each transmitter vehicle *senses* the last 1000 subframes for the selection of the resources. Sensing means that vehicles measure the RSSI value on each subframe in order to determine which resources are likely to be used by other stations. Then, the transmitter excludes the resources from the *selection window* if the expected RSSI value on that resource e.g. based on the sensing is higher than a given threshold. After excluding the resources, the number of resources must be at least 20% of the resources in the selection window. If not, the vehicle increases the threshold and iterates the same process until it reaches at least 20% of the resources. Then, a resource for the transmission is selected randomly within 20% of the resources. For example; if the number of resources is 10, then the user must find at least 2 available resources to select randomly 1 for its transmission.

When a vehicle selects a resource, it reuses the same resource for n (reselection counter) transmissions which are chosen randomly between $[5, 15]$. The reselection counter decreases by 1 after every transmission. When it reaches 0, a vehicle continues to use the same resource with %80 probability otherwise the vehicle selects a new resource for transmission based on the sensing results. This persistent behavior of using the same resource for subsequent transmissions allows other vehicles to determine which resources in the selection window are being used. However, the near vehicles are likely to choose the same resources since they can observe similar RSSI values, which leads to higher IA [Res19b]. The details of the resource selection mechanism of SPS is summarized in Algorithm 3.

5.3.2 Network Settings

The following Table 5.4 will be used as numerology for 5G-V2X settings in the physical layer. Even if the vehicles are able to detect multiple transmission theoretically at the same PRBs thanks to the linear minimum mean square error (LMMSE)-interference rejection combining (IRC) which will be the baseline for 5G, it is assumed that they decode only one transmission that has lower path loss e.g. decode the signal of near users. This will motivate users to select different PRBs for their transmissions. Furthermore, the flexible 5G numerology allows to use $TTI = 0.5ms$ with subcarrier spacing 30kHz which is different from classical LTE-V2X settings i.e. $TTI = 1.0ms$, subcarrier spacing 15kHz. This would facilitate sending all the CAM messages within one subchannel in a TTI which prevents in-band emission and improves system performance.

5.3.3 Packet Reception Ratio

Packet reception ratio in broadcast is defined as the ratio of successfully received packet over the total amount of packet sent within the transmission range between receivers and transmitters. The reliability of scheduling algorithms can be measured with PRR measurements. The Figure 5.11 shows the PRR over time for scenario 1 and 2 in Table 5.3. As seen in Figures, our method outperforms both random and SPS scheduling in

Algorithm 3 SPS algorithm overview

```
1: procedure SPS(RRI, T1, T2, C1, C2, probResourceKeep)
2:     txSubframe ← random(1, RRI)          ▷ Initializations for txSubframe, RC
3:     RC ← random(C1, C2)
4:     txSubCH ← NsubCH                     ▷ We use all available subchannels
5:     subframe ← 0                         ▷ This is the current time
6:     while True do
7:         if subframe == txSubframe then   ▷ It is time to transmit
8:             txPacket(txSubCH)            ▷ Transmit packet on the specified subchannel(s)
9:             if RC ≠ 0 then               ▷ Not time for reselection yet
10:                txSubframe ← txSubframe + RRI   ▷ Schedule next packet in one RRI
11:                RC ← RC − 1
12:            else                         ▷ Time to reselect for next streak
13:                RC ← random(C1, C2)
14:                if random(0, 1) < probResourceKeep then   ▷ keep the same resource location
15:                    txSubframe ← txSubframe + RRI   ▷ Maintain the same RRI across streaks
16:                else                     ▷ Must move to other location for the next streak call
17:                    call select_resource()   ▷ Reselect txSub frame; Section 2.2.2
18:                end if
19:            end if
20:        else
21:            call sensing_update()        ▷ Keep sensing and update resource use map; Section 2.2.2
22:        end if
23:        subframe ← subframe + 1          ▷ Push time
24:    end while
25: end procedure
```

Name	Value
Access scheme	OFDMA
TTI length	0.5ms
PRBs	24
System bandwidth	10MHz
Subcarrier spacing	30kHz
Carrier Frequency	5.9GHz
Antenna numbers	4
Receiver filter	LMMSE Spatial Equalizer
Transmit power	23.0dBm
TFC index	4(QPSK)
CAM message	300bytes

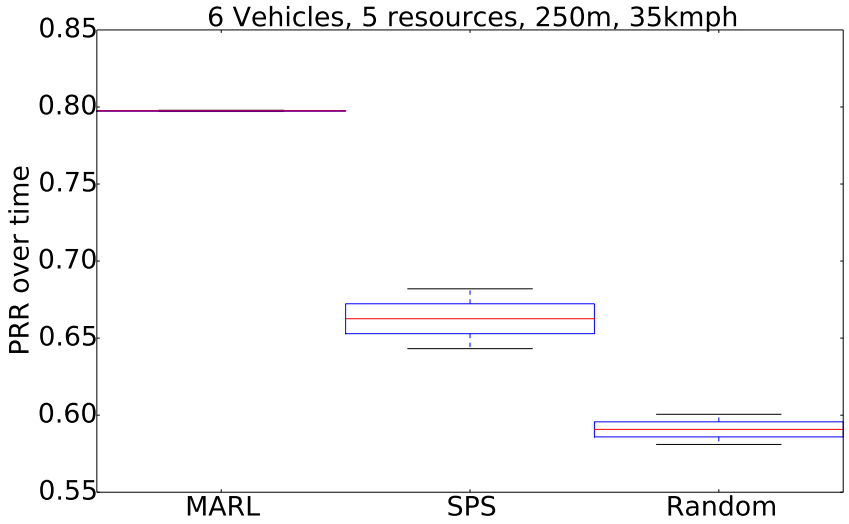
Table 5.4: 5G-V2X Numerology.

terms of overall PRR over time. The measurements are taken for every 100 transmissions over 1000 seconds.

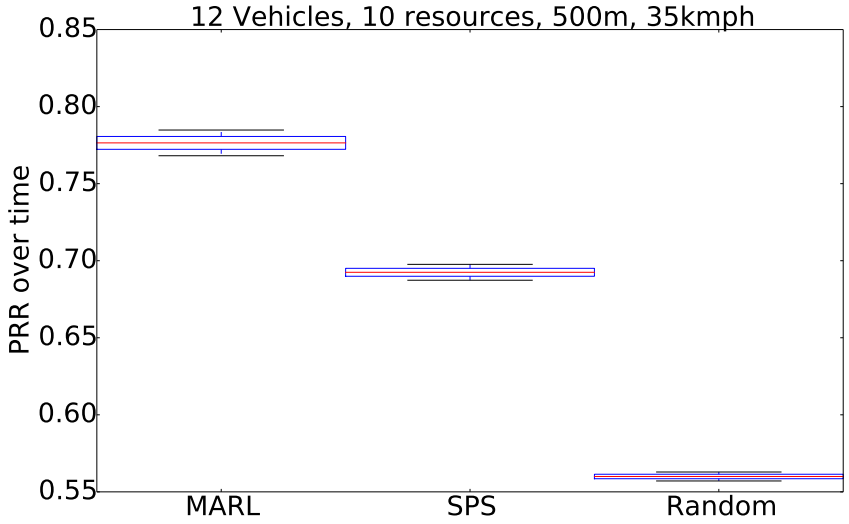
The Figure 5.12 shows the PRR results of the training scenarios 2, 3, and 4 in Table 5.1 where the number of resources is 10. The results indicates that PRR of SPS i.e. 0.9077 is very close to the PRR of MARL scheduling i.e. 0.9086. Although, SPS performs well without congestion case, it suffers from the congestion. We also evaluate the performance of the scenario 3 in Table 5.3. Although the training performance can be improved further, the PRR result in Figure 5.13 shows that the algorithm perform better than SPS. The Figure 5.14 shows the PRR results over distance for both scenarios. In the first scenario, the goal for near vehicles is to use separate resources whereas the farthest vehicles should transmit at the same resource. Therefore, the PRR as seen in Figure 5.14a is higher within near ranges and very low for far vehicles. This also depicts that the trained policy is able to perform the desired behavior. MARL algorithm is affected by the positional distribution of the vehicles on the road. This is the reason why there is a deviation of the result in Figure 5.14b around $300m$. Note that, both SPS and random scheduling have also deviation around $300m$. MARL algorithm is influenced more because it relies on positional distribution for the selection of resources.

5.3.4 Information Age

Additionally, we evaluate and compare the performance of the proposed approach in terms of information age metric. The information age is defined as the time passed since the last received update was generated to measure the timeliness of information updates in a

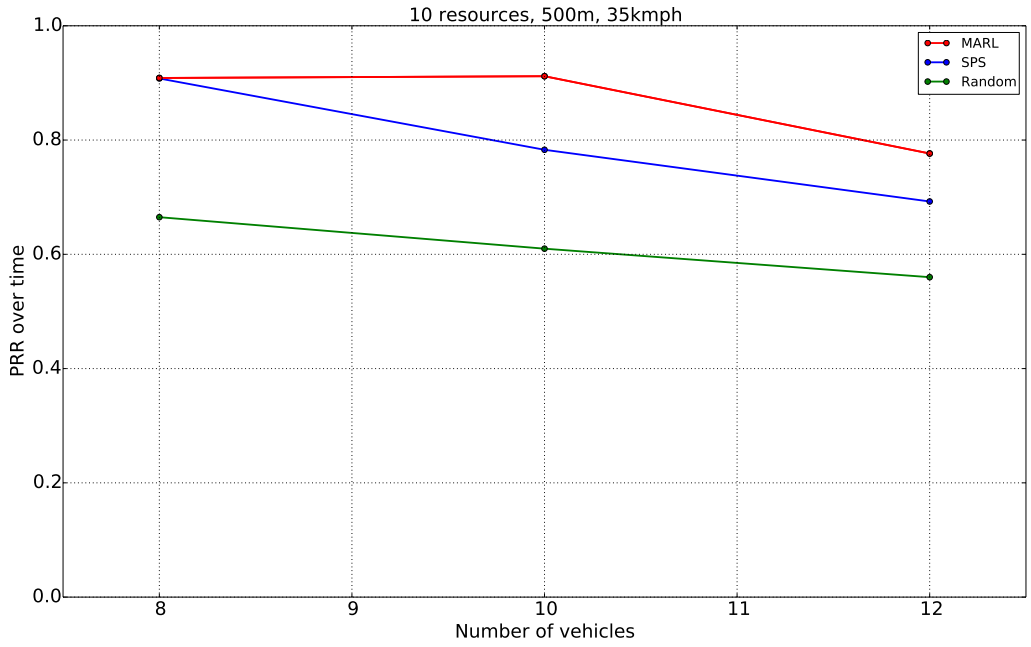


(a) Scenario 1, PRR

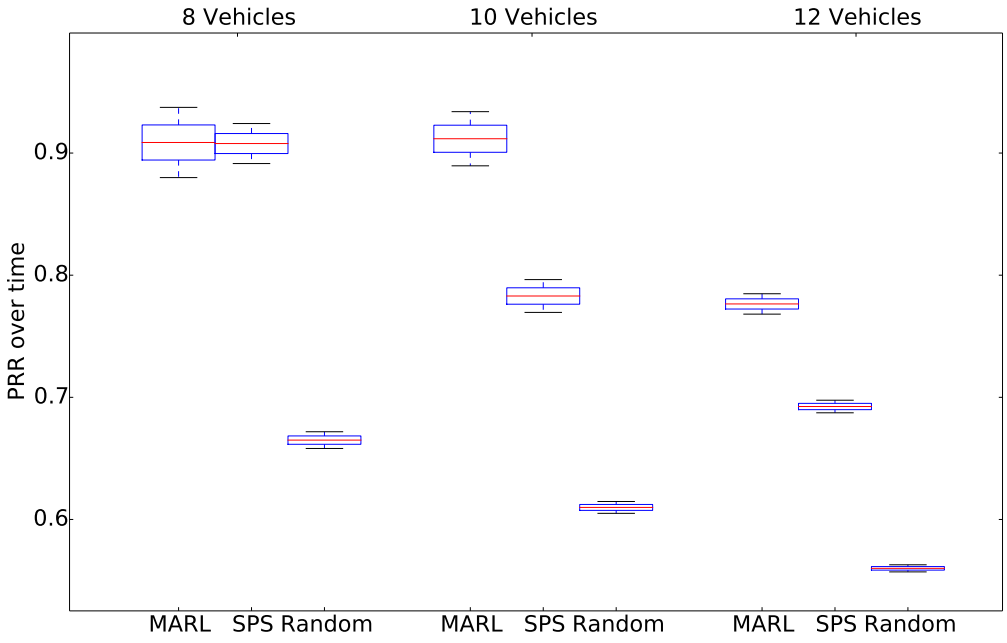


(b) Scenario 2, PRR

Figure 5.11: PRR over time



(a) Average PRR



(b) PRR with confidence

Figure 5.12: PRR of 10 resources with various vehicles

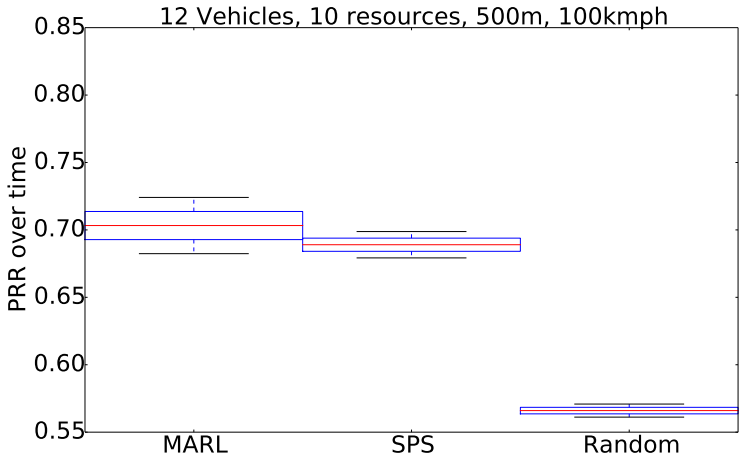
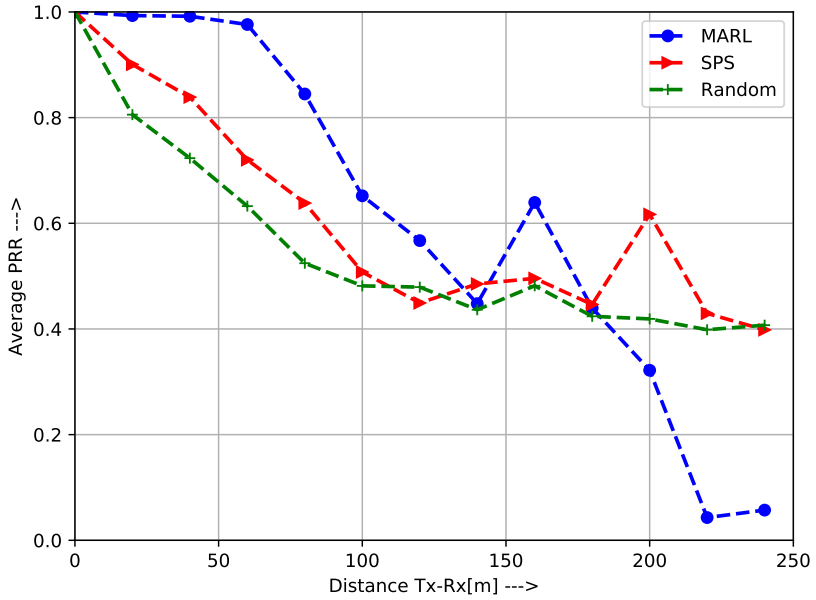
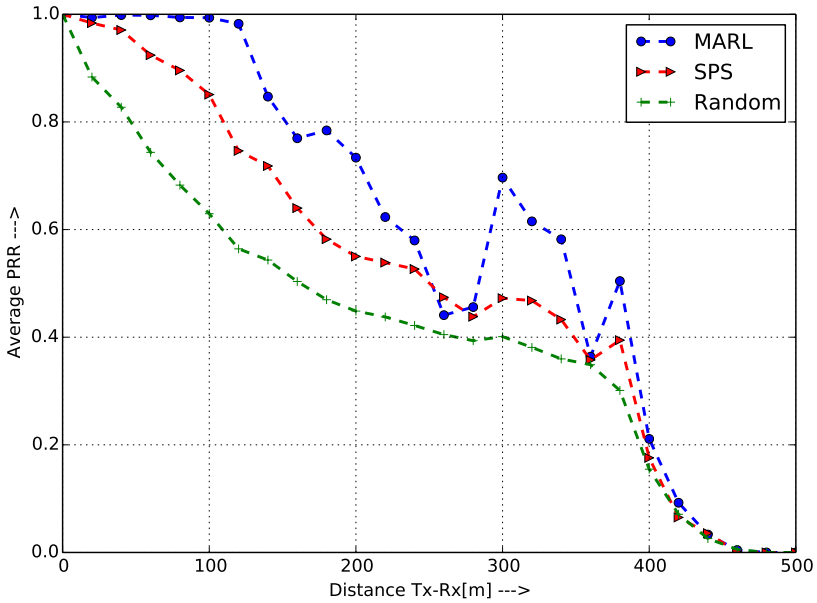


Figure 5.13: Scenario 3, PRR

network. Each vehicle holds a variable for other vehicles in the vicinity to store the time of the last received message. The difference between the current simulation time and the last received update time is calculated for every received message to plot the results. The variable is reset if the other vehicle is outside of the evaluation range. For the analyses, we used the evaluation range of $250m$. The Figure in 5.15 proves that information age is lower for near vehicles. Also, our method outperforms SPS in terms of information age for near vehicles. Simply, vehicles learn to communicate with near vehicles while sacrificing communication with far vehicles in congestion. The performance of the scenario 3 in Table 5.3 is also evaluated and compared with SPS and random scheduling. The results are depicted in Figure 5.16 show that SPS still suffers from the longer tail of information age. Although training performance can be improved further as explained in discussion Section 5.2.6, our proposed approach performs better than SPS algorithm in terms of the information age.

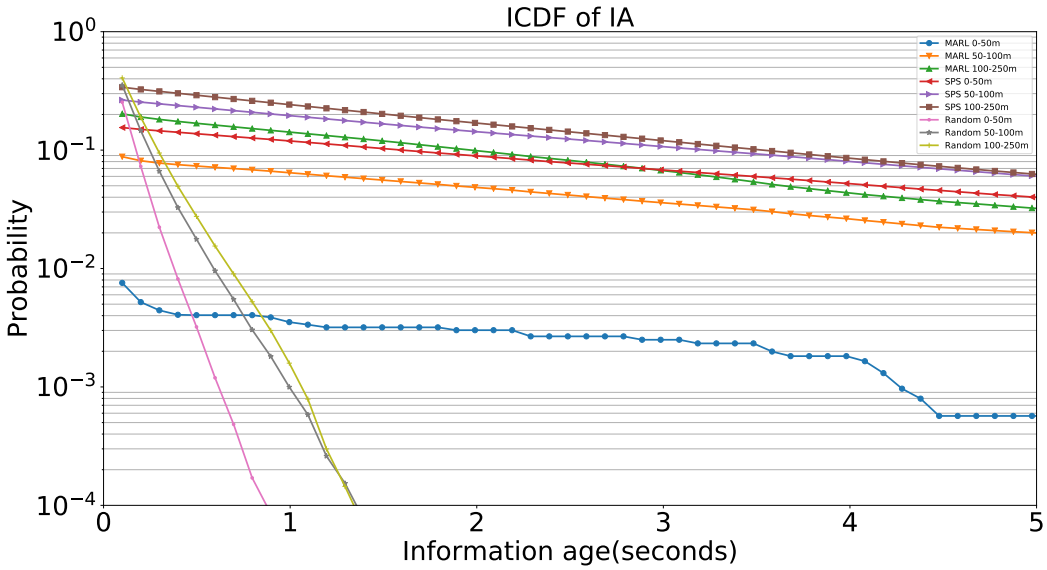


(a) Scenario 1

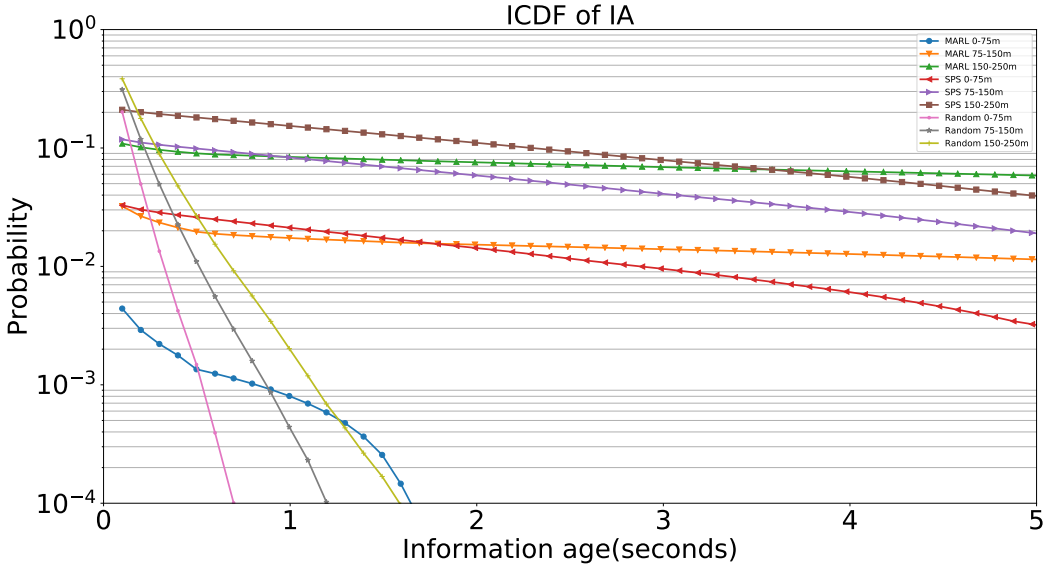


(b) Scenario 2

Figure 5.14: PRR over distance



(a) Scenario 1



(b) Scenario 2

Figure 5.15: ICDF of Information age

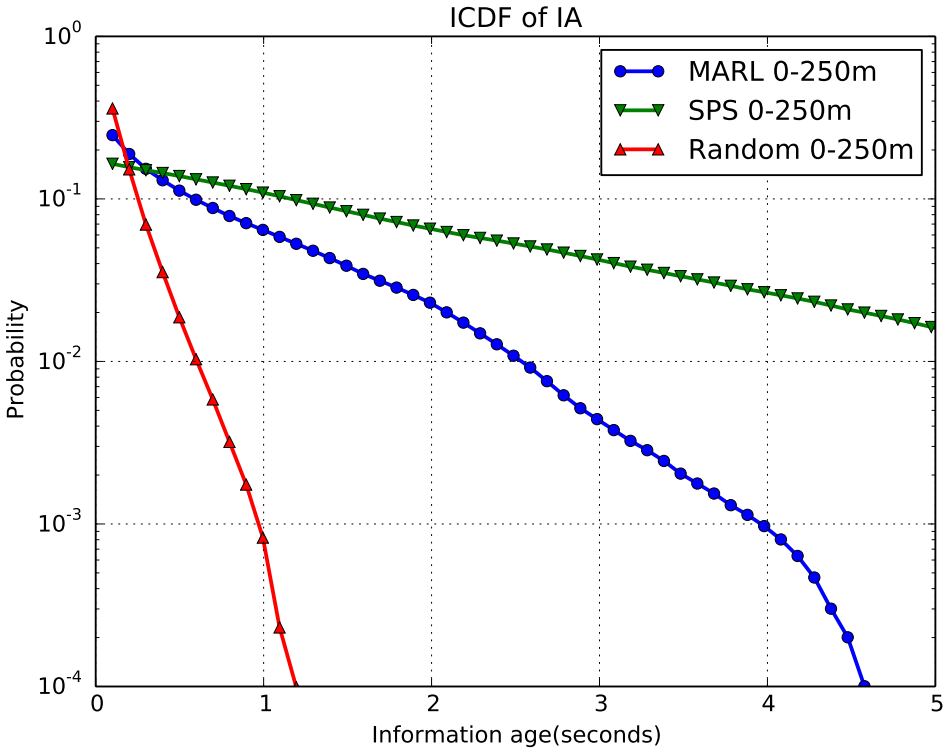


Figure 5.16: Scenario 3, ICDF

Chapter 6

Conclusions and Outlook

V2X communication is an emerging technology to provide a safer driving experience and even collaboration to utilize traffic flow. Also, It is expected to support advanced applications such as driving safety, remote control, platooning and new infotainment experiences for passengers in the new future. In vehicular safety communication, each vehicle sends periodic broadcast messages that contain the position, velocity and heading information of the transmitter. The neighboring vehicles can benefit from those messages to reduce road accidents. In the coverage area of a base station, resources e.g. frequency and time for transmission can be allocated by considering the interference among vehicles. However, the presence of a base station can not be always guaranteed, therefore distributed and autonomous resource allocation is required for vehicles to utilize safety communication.

There are two distributed radio access technologies for V2X communication; DSRC and C-V2X as explained in previous chapters. The studies showed that C-V2X outperforms DSRC technology in terms of its additional link budget, higher resilience to interference and better non-line-of-sight(NLOS) [GAA18]. Furthermore, 3GPP is working on 5G-V2X to support advanced applications. C-V2X adopts semi-persistent-scheduling for distributed resource allocation. Each vehicle senses the received signal strength for the last 1000 subframes, excludes the resources used by other vehicles and decides which resources to allocate within the selection window. Although SPS outperforms DSRC, it suffers from high information age within near vehicles. In SPS, near vehicles are likely to choose the same resources due to similar sensing results based on their positions. The collision persists for many transmission due to the nature of the algorithm(see Section 2.2.2 for details). Half-duplicity constraint does not allow vehicles to receive a message from the other vehicles that transmit at the same resources. Thus, near vehicles can not inform each other about their presence which can increase the probability of road accidents.

In this thesis, a new distributed resource allocation method with multi-agent reinforcement learning is proposed to overcome the main problem as explained above in SPS. The reward of the algorithm is designed to motivate far vehicles to exploit the same resources in case of congestion. We propose a unique state design based on view-based positional distribution. Each vehicle has a neighboring table that contains the positions of other vehicles. The neighboring table is piggybacked with safety messages to always provide updated position values. Then, vehicles construct a view-based positional distribution

vector and ask for an action from the MARL scheduler. After the training, the proposed algorithm converges to a policy such that each vehicle exploits a specific resource for transmission based on local view-based positional distribution vector.

We are inspired by the entity-based field-of-view common knowledge stated in paper [dWFF+18] for the design of view-based positional distribution. Simply, if all agents in the system can see each other, then an agent can estimate the observations of the others from its own observation and selects an action by estimating the actions of others to maximize the system objective. Each agent view is the group's joint view, joint actions are generated based on this joint view. Each agent performs its part from the joint action which is determined by the view of the agent.

The performance of the proposed MARL scheduler is compared with random and standardized SPS scheduling. in terms of packet reception ratio and information age. Our methods provides 12% higher packet reception ratio over time for scenario 4 in Table 5.1. Furthermore, PRR for near transmitters and receivers is higher than SPS and random scheduling. We also compared the results of the information age of MARK scheduling. The results reveal that the information age our method is much lower than SPS for near vehicles. This also proves that the near vehicles use separate resources for transmission so that they can inform each other. In other words, our method provides reliable transmission within near vehicles while sacrificing communication with far vehicles in the congestion case.

As a part of the future work, the performance of our method can be tested in bigger scenarios including more than 100 vehicles. The performance of DQN networks degrades in higher action space. Therefore, one solution to scale the algorithm is to use resource pools. So, the vehicles can be divided into the resource pools with e.g. 10 resources/actions. Each resource pool is scheduled at different times to avoid collision among resource pools. At the end, one policy or different policies for different resource pools can be generated for resource selection. Another solution to scale the algorithm for a large number of users and action space is to use *policy gradient* methods. In policy gradient method, the optimal behavior of the agent can be found by directly modeling and optimizing the policy instead of estimating the true value of each state-action pairs at first, and then obtaining policy by acting greedy. One possible *policy gradient* approach is Proximal Policy Optimization(PPO) which is easy to implement and outperforms other policy gradient methods [SWD+17]. Also, the design of the reward function can be adjusted for different system objectives. For example; information age can be part of the reward design in order to reduce the information age not only for the near vehicles but also for far vehicles.

List of Figures

2.1 Air interfaces [WSGY19] . . . 10
2.2 C-V2X resource allocation [Res19b] . . . 11
2.3 5G Frame Structure [FK17] . . . 12
2.4 Reselection in SPS; p is the resource keeping probability [JKK18] . . . 14
2.5 The agent–environment interaction in a Markov decision process [SB18b] 16
2.6 Comparison between simple RNN and LSTM[Ola15] . . . 21
2.7 Structure of one LSTM unit . . . 21
2.8 Comparison between tabular method and neural networks . . . 22
2.9 Single agent MDP vs Multi-agent Markov game . . . 25
2.10 The MDP, POMDP, and Dec-MDP are the special cases of Dec-POMDP [BZI13] . . . 26

3.1 Example of the output of $f(p_t^i, B, R)$ with $B = 10$ and $R = 100m$ . . . 34
3.2 Illustration of resource pool for action space $|\mathcal{A}| = 10$ . . . 36

4.1 Deep neural network structure . . . 39
4.2 Illustration of the proposed algorithm . . . 41
4.3 Visualization of the vehicles with mobility in RealNeS . . . 44
4.4 Illustration of communication through python interface . . . 45

5.1 N=4, K=3, t=0, moving right direction . . . 47
5.2 Toy example training results . . . 48
5.3 Training performance of different scenarios . . . 48
5.4 6 Vehicles and 5 resources scenario . . . 49
5.5 6 Vehicles, 5 resources, training performance . . . 50
5.6 Scenario 1 with various γ . . . 51
5.7 Training performance of various granularity . . . 51
5.8 Scenario 6, Vanilla DQN vs DQN with LSTM . . . 52
5.9 Scenario 1, random velocity . . . 53
5.10 Scenario 5: 12 Vehicles, 10 resources, 100kmph . . . 53
5.11 PRR over time . . . 58
5.12 PRR of 10 resources with various vehicles . . . 59
5.13 Scenario 3, PRR . . . 60
5.14 PRR over distance . . . 61
5.15 ICDF of Information age . . . 62
5.16 Scenario 3, ICDF . . . 63

List of Tables

2.1 V2X application requirements [3GP18a] 11
2.2 Summary of notations 17

3.1 Example of neighbor table 36

4.1 Python libraries for designing python agent 45

5.1 Scenario settings 46
5.2 Training Hyperparameters 47
5.3 Realness scenario settings 54
5.4 5G-V2X Numerology. 57

List of Abbreviations

CAM	Cooperative Awareness Message
C-V2X	Cellular - Vehicle-to-Everything
CNN	Convolutional Neural Networks
CSMA	Carrier Sense Multiple Access
D2D	device to device
Dec-POMDP	Decentralized Partially Observable Markov Decision Process
DL	Downlink
DNN	Deep Neural Networks
DQN	Deep Q Networks
DRL	Deep Reinforcement Learning
DRQN	Deep Recurrent Q Networks
DSDV	Destination Sequenced Distance Vector
DSRC	Dedicated Short Range Communications
eNodeB	evolved Node B or E-UTRAN Node B
IA	Information Age
IoT	Internet of Things
LSTM	Long Short Term Memory
LTE	Long Term Evolution
LVFA	Linear Value Function Approaximation
MAC	Medium Access Control
MARL	Multi-agent Reinforcement Learning
MCS	Modulation and Coding Scheme
MDP	Markov Decision Process
NLOS	Non-line-of-sight
NR	New Radio

OFDM	Orthogonal Frequency-Division Multiplexing
PDCP	Packet Data Convergence Protocol
PHY	Physical layer
POMDP	Partially Observable Markov Decision Process
PRR	Packet Reception Ratio
PSFCH	Physical Sidelink Feedback Channel
RAT	Radio Access Technology
RB	Resource Block
RealNeS	Realtime Network Simulator
RL	Reinforcement Learning
RLC	Radio Link Control
RNN	Recurrent Neural Networks
RRI	Resource Reservation Interval
RSSI	Received Signal Strength Indicator
SPS	Semi-Persistent Scheduling
TD	Temporal Difference
TPU	Tensor Processing Unit
UAV	Unmanned Aerial Vehicle
UL	Uplink
URLLC	Ultra-Reliable Low Latency Communications
V2I	Vehicle-to-Infrastructure
V2P	Vehicle-to-pedestrian
V2V	Vehicle-to-vehicle
V2X	Vehicle-to-Everything
VPD	View-based Positional Distribution
WHO	World Health Organization

Bibliography

[3GP17a] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) Protocol Specification . Technical Specification (TS) 36.321, 3rd Generation Partnership Project (3GPP), 08 2017. Version 14.3.0.

[3GP17b] 3GPP. Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures. Technical Specification (TS) 36.213, 3rd Generation Partnership Project (3GPP), 08 2017. Version 14.3.0.

[3GP17c] 3GPP. Study on New Radio Access Technology Physical Layer Aspects. 3GPP TR 38.802, 2017.

[3GP17d] 3GPP. Study on New Radio Access Technology: Radio Access Architecture and Interfaces. 3GPP TR 38.801, V14.0.0, 2017.

[3GP18a] 3GPP. 5G; Service requirements for enhanced V2X scenarios. 3GPP TS 22.186 version 15.3.0 Release 15, 2018.

[3GP18b] 3GPP. Study on enhancement of 3GPP Support for 5G V2X Services. 3GPP TS 22.886 version 16.2.0 Release 16, December 2018.

[AAB+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, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan 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. Software available from tensorflow.org.

[ABPS16] Muhammad Ikram Ashraf, Mehdi Bennis, Cristina Perfectod, and Walid Saad. Dynamic proximity-aware resource allocation in vehicle-to-vehicle (V2V) communications. *2016 IEEE Globecom Workshops, GC Wkshps 2016 - Proceedings*, 2016.

[Ame11] American Automobile Association(AAA). Crashes vs. Congestion – What's the Cost to Society? (November), 2011.

[BCM+19] Alessandro Bazzi, Giammarco Cecchini, Michele Menarini, Barbara M. Masini, and Alberto Zanella. Survey and perspectives of vehicular Wi-Fi

versus sidelink cellular-V2X in the 5G era. *Future Internet*, 11(6):1–20, 2019.

[BCV13] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, Aug 2013.

[Ber00] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000.

[BGIZ02] Daniel Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27, 12 2002.

[BKKF14] Mladen Botsov, Markus Klügel, Wolfgang Kellerer, and Peter Fertl. Location dependent resource allocation for mobile device-to-device communications. *IEEE Wireless Communications and Networking Conference, WCNC*, 2:1679–1684, 2014.

[BO99] Tamer Başar and G.J. Olsder. Dynamic non-cooperative game theory. 160, 01 1999.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.

[BZI13] Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. *CoRR*, abs/1301.3836, 2013.

[CLN19] J. Cui, Y. Liu, and A. Nallanathan. Multi-agent reinforcement learning based resource allocation for uav networks. *IEEE Transactions on Wireless Communications*, pages 1–1, 2019.

[CS18] Vehicular Communications and Awareness Basic Service. Draft ETSI Vehicular Communications ; Basic Set of Applications ; Part 2 : Specification of Cooperative. 0:1–45, 2018.

[DHK13] L. Deng, G. Hinton, and B. Kingsbury. New types of deep neural network learning for speech recognition and related applications: an overview. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8599–8603, May 2013.

[dWFF+18] Christian A. Schroeder de Witt, Jakob N. Foerster, Gregory Farquhar, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. Multi-Agent Common Knowledge Reinforcement Learning. 2018.

[FAdFW16] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016.

[FBZ19] S. FEKI, A. BELGHITH, and F. ZARAI. A reinforcement learning-based radio resource management algorithm for d2d-based v2v communication. In

2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), pages 1367–1372, June 2019.

[FFA+17] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017.

[FK17] Eiko Seidel Frank Kowalewski. 5g frame structure. Technical report, Nomor Research GmbH, Germany, White Paper, August 2017.

[FLZ+18] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. Deepmove: Predicting human mobility with attentional recurrent networks. In *Proceedings of the 2018 world wide web conference*, pages 1459–1468, 2018.

[FNF+17] Jakob N. Foerster, Nantas Nardelli, Gregory Farquhar, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. *CoRR*, abs/1702.08887, 2017.

[Foe18] Jakob N Foerster. *Deep multi-agent reinforcement learning.* PhD thesis, University of Oxford, 2018.

[GAA18] ("5GAA") 5G Automotive Association. V2X Technology Benchmark Testing. 2018.

[GE] Jayesh K Gupta and Maxim Egorov. Cooperative Multi-Agent Control Using Deep Reinforcement Learning.

[GEK17] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing.

[GZ19] Piotr Gawłowicz and Anatolij Zubow. ns-3 meets OpenAI Gym: The Playground for Machine Learning in Networking Research. In *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, November 2019.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS14] National Highway and Traffic Safety. Fact Sheet : Improving Safety and Mobility Through Connected Vehicle Technology. Technical report, 2014.

[HS15] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs. *AAAI Fall Symposium - Technical Report*, FS-15-06:29–37, 2015.

[HSN17] Maximilian Hüttenrauch, Adrian Sosic, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems. *CoRR*, abs/1709.06011, 2017.

[HVS11] M. I. Hassan, H. L. Vu, and T. Sakurai. Performance analysis of the ieee 802.11 mac protocol for dsrc safety applications. *IEEE Transactions on Vehicular Technology*, 60(8):3882–3896, Oct 2011.

[JKK18] Yongseok Jeon, Seungho Kuk, and Hyogon Kim. Reducing message collisions in sensing-based semi-persistent scheduling (sps) by using reselection lookaheads in cellular v2x. *Sensors*, 18(12):4388, 2018.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[KHFW02] Daniel Krajzewicz, Georg Hertkorn, Christian Feld, and Peter Wagner. Sumo (simulation of urban mobility); an open-source traffic simulation. pages 183–187, 01 2002.

[KMH+19] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. *CoRR*, abs/1902.01554, 2019.

[LBBH98] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[LBH15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[LHG+18] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *CoRR*, abs/1810.07862, 2018.

[Lit94] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 157–163, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[LL15] Dong Li and Yong Liu. In-band emission in LTE-A D2D: Impact and addressing schemes. *IEEE Vehicular Technology Conference*, 2015:1–5, 2015.

[LLZC18] Mingxiao Li, Feng Lu, Hengcai Zhang, and Jie Chen. Predicting future locations of moving objects with deep fuzzy-lstm networks. *Transportmetrica A: Transport Science*, pages 1–18, 2018.

[LMY+] Le Liang, Student Member, Hao Ye, Student Member, and Geoffrey Ye Li. Towards Intelligent Vehicular Networks : A Machine Learning Framework. pages 1–12.

[LYL19] Le Liang, Hao Ye, and Geoffrey Ye Li. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *CoRR*, abs/1905.02910, 2019.

[LZG+17] Marc Lanctot, Vinícius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *CoRR*, abs/1711.00832, 2017.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fid-

jeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[NC17] Oshri Naparstek and Kobi Cohen. Deep Multi-User Reinforcement Learning for Distributed Dynamic Spectrum Access. pages 1–14, 2017.

[NMCP19] Gaurang Naik, Student Member, Biplav Choudhury, and Jung-min Jerry Park. IEEE 802 . 11bd & 5G NR V2X : Evolution of Radio Access Technologies for V2X Communications. (March), 2019.

[NMGM] Yasar Sinan Nasir, Student Member, Dongning Guo, and Senior Member. Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks. pages 1–12.

[NVD12] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. *Game Theory and Multi-agent Reinforcement Learning*, pages 441–470. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[Ola15] Christopher Olah. Understanding lstm networks. *Colah's blog*, August 2015.

[OSV08] F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, May 2008.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM '94, page 234–244, New York, NY, USA, 1994. Association for Computing Machinery.

[R1-18a] R1-1810051: Final report of 3gpp tsg ran wg1 94 v1.0.0. Final report, 3GPP, October 2018.

[R1-18b] R1-1810977: Discussion on mode 2 resource allocation in nr-v2x. Technical report, Guangdong OPPO Mobile Telecom, October 2018.

[R1-18c] R1-1813230: Preliminary link level evaluations on reference signal for nr v2x. Final report, InterDigital Inc., November 2018.

[R1118] R11809867: Offline summary for nr-v2x agenda item - 7.2.4.1.4 resource allocation mechanism. Technical report, Intel Corportation, August 2018.

[Res19a] Nomor Research. Comparison of V2X based on 802.11p, LTE and 5G. 2019.

[Res19b] Nomor Research. Comparison of v2x based on 802.11p, lte and 5g. Technical report, Nomor Research GmbH, Germany, White Paper, April 2019.

[RN94] G. Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

[SB18a] T. Sahin and M. Boban. Radio resource allocation for reliable out-of-coverage v2v communications. In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pages 1–5, June 2018.

[SB18b] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[Sch14] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[SKS16] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level. In *IJCAI*, volume 16, pages 2618–2624, 2016.

[SSF16] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *CoRR*, abs/1605.07736, 2016.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[Tro16] John Tromp. Combinatorics of Go. pages 1–38, 2016.

[TW12] Karl Tuyls and Gerhard Weiss. Multiagent Learning: Basics, Challenges, and Prospects. *Ai Magazine*, 33:41–52, 2012.

[vHGS15] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015.

[WMG17] Xuyu Wang, Shiwen Mao, and Michelle X. Gong. An overview of 3gpp cellular vehicle-to-everything standards. *GetMobile: Mobile Comp. and Comm.*, 21(3):19–25, November 2017.

[Wor18] World Health Organization. Global status report on road safety 2018. Technical report, Genf, Schweiz, 2018.

[WSGY19] Jian Wang, Yameng Shao, Yuming Ge, and Rundong Yu. A survey of vehicle to everything (v2x) testing. In *Sensors*, 2019.

[YLJ19] H. Ye, G. Y. Li, and B. F. Juang. Deep reinforcement learning based resource allocation for v2v communications. *IEEE Transactions on Vehicular Technology*, 68(4):3163–3173, April 2019.

[ZLGV] Chen Zhong, Ziyang Lu, M Cenk Gursoy, and Senem Velipasalar. A Deep Actor-Critic Reinforcement Learning Framework for Dynamic Multichannel Access. 13244.

[ZYB19] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. *arXiv e-prints*, page arXiv:1911.10635, Nov 2019.

[ZYC+17] Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *CoRR*, abs/1712.00600, 2017.

[ZYL+18] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. Fully decentralized multi-agent reinforcement learning with networked agents. *CoRR*, abs/1802.08757, 2018.