Annals of Telecommunications
https://doi.org/10.1007/s12243-021-00872-w

Network slicing for vehicular communications: a multi-agent deep reinforcement learning approach

Zoubeir Mlika[1] · Soumaya Cherkaoui[1]

Received: 16 February 2021 / Accepted: 19 July 2021

Abstract
This paper studies the multi-agent resource allocation problem in vehicular networks using non-orthogonal multiple access (NOMA) and network slicing. Vehicles want to broadcast multiple packets with heterogeneous quality-of-service (QoS) requirements, such as safety-related packets (e.g., accident reports) that require very low latency communication, while raw sensor data sharing (e.g., high-definition map sharing) requires high-speed communication. To ensure heterogeneous service requirements for different packets, we propose a network slicing architecture. We focus on a non-cellular network scenario where vehicles communicate by the broadcast approach via the direct device-to-device interface (i.e., sidelink communication). In such a vehicular network, resource allocation among vehicles is very difficult, mainly due to (i) the rapid variation of wireless channels among highly mobile vehicles and (ii) the lack of a central coordination point. Thus, the possibility of acquiring instantaneous channel state information to perform centralized resource allocation is precluded. The resource allocation problem considered is therefore very complex. It includes not only the usual spectrum and power allocation, but also coverage selection (which target vehicles to broadcast to) and packet selection (which network slice to use). This problem must be solved jointly since selected packets can be overlaid using NOMA and therefore spectrum and power must be carefully allocated for better vehicle coverage. To do so, we first provide a mathematical programming formulation and a thorough NP-hardness analysis of the problem. Then, we model it as a multi-agent Markov decision process. Finally, to solve it efficiently, we use a deep reinforcement learning (DRL) approach and specifically propose a deep Q learning (DQL) algorithm. The proposed DQL algorithm is practical because it can be implemented in an online and distributed manner. It is based on a cooperative learning strategy in which all agents perceive a common reward and thus learn cooperatively and distributively to improve the resource allocation solution through offline training. We show that our approach is robust and efficient when faced with different variations of the network parameters and compared to centralized benchmarks.

Keywords Network slicing · Vehicle-to-vehicle · Non-orthogonal multiple access · Resource allocation · Deep Q learning · Deep reinforcement learning

1 Introduction

Vehicle-to-everything (V2X) communication is an integral part of future intelligent transportation systems (ITS) and has become a key communication paradigm in future wireless networks. Indeed, V2X communication is one of the key verticals in fifth generation (5G) networks [9, 28]. In 3rd generation partnership project (3GPP) release 16, the new radio (NR) cellular-V2X (C-V2X) communication supports two transmission modes: a direct mode with NR C-V2X sidelink and an over-the-cellular network mode that, together, provide seamless connectivity to vehicles through a unified radio. V2X communication enables information sharing through vehicle-infrastructure (V2I), vehicle-to-pedestrian, vehicle-to-vehicle (V2V), or vehicle-to-cellular network (V2N) communication. V2X communication covers a range of use cases that can broadly be divided into safety services, non-safety services, and infotainment services. Safety services are the main

✉ Zoubeir Mlika
zoubeir.mlika@usherbrooke.ca

Soumaya Cherkaoui
soumaya.cherkaoui@usherbrooke.ca

1 Department of Electrical and Computer Engineering, Université de Sherbrooke, Sherbrooke, Canada

use case of V2X communication, and aim to improve road safety by enabling road users and/or infrastructure to exchange information in a timely manner to avoid accidents. Non-safety services are offered by ITS to improve traffic management and efficiency. Infotainment services include other value-added services for vehicle users such as video streaming or content sharing [17, 27]. Despite the convenience of a unified radio, the diverse service requirements of vehicular applications make the problem of resource allocation in V2X communication very challenging. A key challenge is to support future 5G vehicular services with extremely diverse performance requirements using a unified radio interface. Network slicing is a potential solution to solve this key challenge [9, 17]. Network slicing is a tool that allows network operators to support virtualized end-to-end networks by creating different logical networks on top of a common, programmable physical infrastructure.

In this paper, we propose a novel solution to solve the challenging problem of resource allocation in a 5G vehicular network using network slicing. Specifically, we focus on V2V sidelink communication without the assistance of a base station such as a 5G-NR next generation NodeB (gNB). Network slicing in such scenarios is more challenging because the proposed solutions must be distributed and implemented independently in each vehicle. In V2V communication, 5G-NR has defined different ways to send information, including broadcast, unicast and groupcast communication techniques [3]. In this article, the broadcast technique is considered. On the other hand, non-orthogonal multiple access (NOMA) is a promising technique for increasing spectrum efficiency in wireless networks in general [11] and in vehicular networks in particular [12]. NOMA has been shown to be effective in providing broadband communications and massive connectivity because it allows multiple transmitting devices to share common spectrum resources (in time and/or frequency) [13].

In this paper, we propose network slicing–based resource allocation framework using NOMA for V2V broadcast communication. We answer the following important question: “how to distributively allocate the different resources in network slicing–based vehicular network in order to guarantee the latency, reliability and rate requirements of the different slices?”.

The problem considered, hereafter called vehicle resource allocation (VRA), is a four-dimensional problem that involves the allocation of four resources corresponding to the following four decisions: deciding which packet to send, deciding the broadcast range, deciding which resource blocks (RBs) to use for transmission, and deciding the transmission power. The use of NOMA is attractive for the development of distributed solutions because some RBs can be reused by multiple transmitting vehicles and thus there is no need to pay attention to collisions. By carefully allocating different vehicle resources (slice, coverage, RBs, and power) and applying successive interference cancellation (SIC) to the corresponding receivers, NOMA can help increase the capacity of various vehicle network applications. It should be noted that the use of NOMA in broadcast communications is different from the usual NOMA technique in wireless uplink and downlink networks and is more difficult to apply. This is due to the nature of broadcasting in vehicle networks: two transmitting vehicles broadcasting with two different transmission powers to the *same* group of receiving vehicles must carefully distribute their transmission powers so that the corresponding receivers can successfully apply SIC.

1.1 Contributions

To the best of our knowledge, this is the first work that distributively solves VRA in 5G NR C-V2X sidelink communication based on network slicing and NOMA. To do so, we apply deep reinforcement learning (DRL) [24]. In general, deep learning (DL) has had significant success in various disciplines and is applied differently to various problems in vehicular networks [36]. With the help of DRL, we provide a distributed and model-free solution to VRA. On the other hand, we propose a model-based formulation using integer programming and analyze the NP-hardness of VRA. The contributions of our work are summarized in the following list.

- We formulate VRA as a mixed integer non-linear program.
- We prove that VRA is NP-hard even in its simplest form.
- We model VRA as a multi-agent Markov decision process.
- We propose a deep Q learning (DQL) algorithm to provide an online and distributed resource allocation solution that incites, through proper reward function design, the agents (the vehicles) to work cooperatively.
- We compare DQL to optimal and centralized benchmark algorithms that we have adapted to solve VRA and we demonstrate the superior performance of DQL.

1.2 Related work

Di et al. [13] used NOMA to solve the RB and power allocation problem in a V2X communication network where the objective is to maximize the number of successfully decoded signals subject to scheduling and fairness constraints. The proposed solution is applied in two phases: a centralized phase and a distributed phase. In the centralized phase, the authors solved the RB allocation problem by transforming it into an assignment problem and propose a rotation-based algorithm. In the distributed phase, the authors proposed a distributed algorithm to find the transmission power of vehicles to improve the performance of NOMA. Liang et al. [21] focused on maximizing the sum rate of V2I links while guaranteeing minimum quality of service (QoS) for V2I and V2V links. The minimum QoS of V2V links ensured reliable communication by imposing a minimum threshold value for the outage probability of the achievable rate. The authors assumed that the channel state information (CSI) at the base station is delayed due to high mobility. The authors decoupled the problem by considering the case of a single V2I user and a single V2V user and derived the power allocation vector. Then, using the allocated power, they found the spectrum allocation by transforming the problem into an assignment problem and used the Hungarian algorithm to solve it. The proposed solution is based on a central coordination point and does not include network slicing or vehicle coverage optimization. Nasir et al. [29] solved the classical power allocation problem in an interference channel using model-free DRL. Particularly, the proposed solution can be summarized as follows. After collecting CSI and QoS information, each transmitter adapts its own transmit power accordingly. The objective is, as usual, to maximize the total weighted sum-rate. The proposed method used DQL to develop a distributed solution. Liang et al. [23] proposed a multi-agent DRL framework to solve the challenging problem of resource allocation in a V2X communication network. In particular, the problem involved spectrum sharing between V2I links and V2V links and the objective was to ensure ultra-low latency communication for V2V links and high throughput for V2I links. The proposed solution is based on DQL to provide a distributed solution for the RB and power allocation problem. Campolo et al. [10] used the LTE V2X Mode 4 scenario to improve the decoding probability of V2V links while considering full duplex radios. The overall objective was to improve the existing approach used in LTE V2X Mode 4 and specifically the work aims to improve resource allocation and reduce collisions and packet decoding. However, the resource allocation should be updated every period based on the collision history which may increase the overhead. Liang et al. [22] examined resource allocation in vehicular device-to-device communications. In particular, the authors solved the problem of spectrum and power allocation while maximizing the throughput of V2I links and ensuring the reliability of V2V links. Zhang et al. [38] studied a vehicle network using multi-access edge computing (MEC) with backup cloud servers. They proposed a DQL algorithm to offload vehicle tasks to the MEC servers to minimize the total processing time of vehicle tasks. The results showed superiority in terms of processing time, especially when backup cloud servers are considered. Sharif et al. [4] studied the problem of clustering vehicles in Internet of vehicles (IoV) networks. They proposed a DRL-based approach to select cluster heads for resource allocation among vehicles. Their approach is based on the DRL actor-critic method where the policy and the value function are separated. The policy is called the actor because it selects actions while the value function is called the critic because it criticizes the actions chosen by the actor. Their approach was compared to static and DQN-based approaches and found to outperform them, especially in terms of convergence time, i.e., the actor-critic approach required fewer iterations to achieve better throughput. Zhang et al. [39] studied the problem of mode selection (V2V mode or V2I mode) and resource allocation (RB and power) in V2X communication networks. The problem is formulated to maximize the total capacity of V2I users and guarantee the latency and reliability of V2V users. The authors proposed a multi-agent DRL approach in a two-time scale architecture. In a large time scale, a graph-theoretic based vehicle clustering algorithm is proposed. In a small time scale, vehicles in the same group cooperate to form a single DRL model based on federated learning. Ye et al. [35] considered RB and power allocation in V2X communication networks. The aim was to design a multi-agent DRL algorithm where V2V agents learn to reduce interference with all V2I links and with other V2V links while meeting their latency requirements. A DQL algorithm is proposed with low communication overhead.

In the above previous work, the integration of network slicing and NOMA in V2V communication networks with the consideration of broadcast communication in 5G-NR C-V2X-assisted side-link communication scenario is not well analyzed and solved. The challenging multidimensional problem of coverage, slice, RB, and power optimization is not solved distributively and without the help of cellular network in previous work. In this work, we fill this research gap by first proposing a mathematical programming model and then studying its NP-hardness. Then, we use DRL to propose a distributed multi-agent solution to this multidimentional problem.

1.3 Organization

The paper is organized as follows. Section 2 presents the model including the system assumptions and the mathematical programming model and it studies the NP-hardness of the problem. Section 3 presents the multi-agent deep reinforcement learning framework and describes the proposed algorithm. Section 4 presents centralized benchmark algorithmic solutions and gives some simulation results. Finally, Section 5 draws some conclusions.

2 Model

2.1 System model

We consider a vehicular network composed of $n + m$ vehicles. A set $\mathcal{V}$ of m vehicles generates traffic and is called transmitting vehicles. The remaining vehicles form a set $\mathcal{W}$ of n receiving vehicles. All vehicles operate in the 5G-NR C-V2X-assisted sidelink communication scenario [3] and thus communicate with each others via the direct device-to-device interface. The transmitting vehicles transmit according to the broadcast communication approach [6]. Since transmitting vehicles are not able to broadcast to all other vehicles, coverage selection must be optimized. In other words, a transmitting vehicle $v \in \mathcal{V}$ can broadcast only to a subset $\mathcal{W}_v \subseteq \mathcal{W}$ of the receiving vehicles. The proposed resource allocation is carried out autonomously by the vehicles according to the autonomous mode of 5G-NR [6]. The total time duration is divided into a set $\mathcal{T}$ of T time-slots of length τ seconds each. The total transmission bandwidth is divided into a set $\mathcal{F}$ of F frequency-slots. A resource block (RB) is given by the pair (f, t) for each frequency-slot f and time-slot t [1].

Several use cases can be supported by our model, including cooperative communication using extended sensors [6]. To be able to provide guaranteed heterogeneous service requirements of these use cases, we propose a communication architecture based on network slicing. Network slicing is an effective solution to satisfy the requirements of various use cases of wireless networks in general and vehicular networks in particular [17]. This is done primarily by creating logical networks on top of a common, programmable infrastructure [17]. It is known that network slicing involves the core network (CN) as well as the radio access network (RAN). In this paper, we consider network slicing in the RAN only. In our system model, we create two slices. The first slice is reserved for non-safety applications and is called the non-safety slice (or slice 1). It is designed primarily to support non-safety related traffic that is characterized by high-throughput transmissions, e.g., video streaming. On the other hand, the second slice is reserved for safety applications and is referred to as the safety slice (or slice 2), which is designed primarily for safety-related traffic that is characterized by extremely low latency requirements [8], e.g., emergency alerts and accident reports. An example of our system model is given in Fig. 1.

Each transmitting vehicle $v \in \mathcal{V}$ generates two packets of size [in bits] σ_v^{n} and σ_v^{s}. Without loss of generality, the packet sizes are called packet requirements. Each packet corresponds to a specific traffic (either non-safety related traffic or safety related traffic). Based on our network slicing architecture, a non-safety related packet is transmitted on the logical non-safety network slice and requires a high throughput. On the contrary, a safety-related packet is transmitted on the logical safety network slice and thus has strict latency requirements. We call these two packets "non-safety packet" and "safety packet" and refer to them as $\mathsf{pkt}_v^{\mathsf{n}}$ and $\mathsf{pkt}_v^{\mathsf{s}}$ respectively. For a packet to be successfully transmitted, the corresponding transmitting vehicle must be assigned a number of RBs such that the number of achievable bits on those assigned RBs is at least equal to the requirements of the corresponding packet. Note that the non-safety packet $\mathsf{pkt}_v^{\mathsf{n}}$ corresponding to the vehicle v can be assigned any of $\mathcal{F} \times \mathcal{T}$ RBs so that its requirements are met. On the other hand, the safety packet $\mathsf{pkt}_v^{\mathsf{s}}$ arrives during the considered time horizon at a specific time-slot s_v. Due to the strict latency requirements of $\mathsf{pkt}_v^{\mathsf{s}}$, we associate with it a positive integer e_v, where $s_v \leq e_v \leq T$, that denotes its deadline. Thus, $\mathsf{pkt}_v^{\mathsf{s}}$ must meet its requirements by scheduling it on any frequency-slot f such that for the RB (f, t), the time-slot t belongs to $\{s_v..e_v\}$[1], i.e., the admissible set of RBs belongs to $\mathcal{F} \times \{s_v..e_v\}$.

The normalized power gain of the wireless channel between $v \in \mathcal{V}$ and $w \in \mathcal{W}$ on the RB (f, t) is given by $g_{v,w,f}(t) = |h_{v,w,f}(t)|^2/N_0$, where N_0 is the noise power and $h_{v,w,f}(t)$ is the channel coefficient that includes fast and slow fading. Each $v \in \mathcal{V}$ transmits pkt_v^{i} (for $i \in \{\mathsf{s}, \mathsf{n}\}$) with a transmission power $p_{v,f}^{i}(t)$ through RB (f, t) that belongs to $[0, p_v^{\mathsf{max}}]$.

In the considered system model, the selection of vehicle coverage, i.e., the selection of the receiving vehicles targeted for broadcast, is a significant challenge. According to [2], a transmitting vehicle transmits to all intended receiving vehicles within a certain distance, and this distance must be carefully designed to improve the performance of vehicular networks. The selection of vehicle coverage can be seen as vehicle clustering that was recently discussed in [30]. Thus, for successful packet transmission, a transmitting vehicle $v \in \mathcal{V}$ must: (i) decide whether to send both packets or only the safety packet, (ii) decide which receiving vehicles to broadcast to, (iii) allocate the necessary RBs, and (iv) allocate the transmission powers $p_{v,f}^{\mathsf{s}}(t)$ and $p_{v,f}^{\mathsf{n}}(t)$ for each allocated RB (f, t). To improve the spectral efficiency of the system, non-orthogonal multiple access (NOMA) can be used to overlay transmissions from the transmitting vehicles to a common receiving vehicle. The common receiving vehicle then uses successive interference cancellation (SIC) to decode the superimposed transmissions. For simplicity, if a transmitting vehicle transmits both packets, it uses orthogonal RBs, i.e., NOMA is applied in different transmitting vehicles and not in the transmission of the same transmitting vehicle.

[1]The notation $\{a..b\}$ with $b \geq a$ denotes the integer interval between a and b (inclusive).

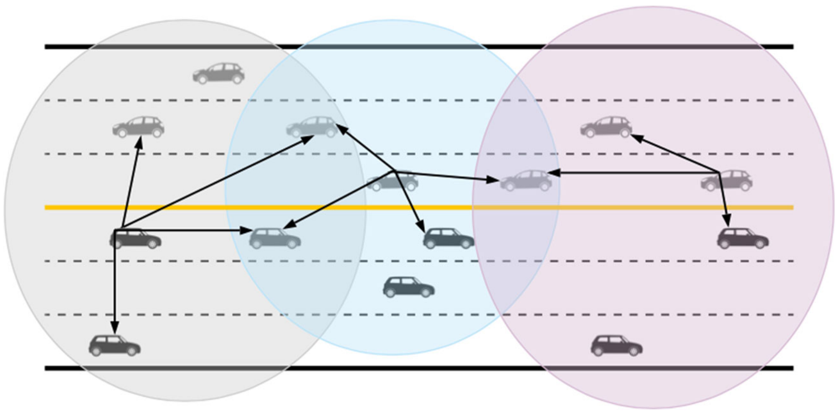

Fig. 1 An example of the system model with three transmitting vehicles

2.2 VRA optimization model

Let $x^{\mathsf{n}}_{v,w,f}(t) = 1$ (resp. $x^{\mathsf{s}}_{v,w,f}(t) = 1$) if and only if vehicle v transmits to vehicle w the non-safety packet (resp. the safety packet) on the RB (f, t) and $y^{\mathsf{n}}_{v,w} = 1$ (resp. $y^{\mathsf{s}}_{v,w} = 1$) if and only if vehicle v transmits the non-safety packet (resp. the safety packet) to vehicle w.

The signal-to-interference-plus-noise ratio (SINR) of the ith packet (i denotes either the non-safety packet or the safety packet, i.e., $i \in \mathsf{n}, \mathsf{s}\}$) between vehicle $v \in \mathcal{V}$ and vehicle $w \in \mathcal{W}_v$ on the RB (f, t) is given by :

$$\mathsf{sinr}^i_{v,w,f}(t) := \frac{p^i_{v,f}(t) g_{v,w,f}(t)}{1 + I_{v,w,f}(t)}, \tag{1}$$

where $I_{v,w,f}(t)$ is the interference generated by other transmitting vehicles broadcasting on RB (f, t), which is given by:

$$I_{v,w,f}(t) = \sum_{v' \in \mathcal{V}_{v,w,f}(t)} g_{v',w,f}(t)\big(p^{\mathsf{n}}_{v',f}(t) + p^{\mathsf{s}}_{v',f}(t)\big), \tag{2}$$

and $\mathcal{V}_{v,w,f}(t) = \{v' \in \mathcal{V} : g_{v,w,f}(t) > g_{v',w,f}(t)\}$. In Eq. 2, we used $p^{\mathsf{n}}_{v',f}(t) + p^{\mathsf{s}}_{v',f}(t)$ because we assumed that a transmitting vehicle sends its packets on orthogonal RBs.

Our main focus is on optimizing the packet reception ratio (PRR) in V2X communication networks. Technically, it is called Type 2 PRR and is defined in [2] as follows: for a packet and a transmitting vehicle, the PRR is given by the percentage of vehicles with successful reception among the total number of receiving vehicles. For this purpose, we optimize the total number of vehicles with successful reception. Therefore, the objective function is given by :

$$\sum_{v \in \mathcal{V}} \sum_{w \in \mathcal{W}_v} \big(y^{\mathsf{n}}_{v,w} + y^{\mathsf{s}}_{v,w}\big). \tag{3}$$

We note that the objective function is an unweighted sum of the number of successfully delivered packets, which means that safety and non-safety packets are treated the same. Nevertheless, since safety packets are generally lighter in bit size, they should be delivered more often than non-safety packets as we show in the simulation results. Adding the weights λ_{s} and λ_{n} to the objective function is a straightforward approach to prioritize each type of packet, but further analysis of fairness is needed, which will be left for our future work.

The vehicular resource allocation (VRA) problem is formulated as an optimization program as follows.

$$\text{maximize} \quad \sum_{v \in \mathcal{V}} \sum_{w \in \mathcal{W}_v} \big(y^{\mathsf{n}}_{v,w} + y^{\mathsf{s}}_{v,w}\big), \tag{4a}$$

$$\text{subjto} \quad x^{\mathsf{n}}_{v,w,f}(t), x^{\mathsf{s}}_{v,w,f}(t), y^{\mathsf{n}}_{v,w}, y^{\mathsf{s}}_{v,w} \in \{0, 1\}, \forall v, w, f, t, \tag{4b}$$

$$p^i_{v,f}(t) \in [0, p^{\mathsf{max}}_v], \forall v, f, t, \tag{4c}$$

$$\sum_{f \in \mathcal{F}} \sum_{t \in \mathcal{T}} \beta\tau \lg\big(1 + \mathsf{sinr}^i_{v,w,f}(t)\big) \geq \sigma^i_v y^i_{v,w}, \forall v, w, i, \tag{4d}$$

$$x^i_{v,w,f}(t) + x^i_{v,w',f'}(t) \leq 1, \forall v, t, f \neq f', w, w', i, \tag{4e}$$

$$x^i_{v,w,f}(t) \leq y^i_{v,w}, \forall v, w, f, t, i, \tag{4f}$$

$$y^i_{v,w} \leq \sum_{f \in \mathcal{F}} \sum_{t \in \mathcal{T}} x^i_{v,w,f}(t), \forall v, w, i, \tag{4g}$$

$$x^{\mathsf{n}}_{v,w,f}(t) + x^{\mathsf{s}}_{v,w,f}(t) \leq 1, \forall v, w, f, t, \tag{4h}$$

$$x^{\mathsf{s}}_{v,w,f}(t) = 0, \forall v, w, f, t \notin \{s_v..e_v\}, \tag{4i}$$

$$p^i_{v,f}(t) \leq p^{\mathsf{max}}_v x^i_{v,w,f}(t), \forall v, w, f, t, i, \tag{4j}$$

The objective function and the constraints are explained in the following list.

- The objective function in Eq. 4a represents the number of successfully received packets.
- Constraints Eqs. 4b and 4c represent the optimization variables.
- Constraints Eq. 4d force a packet to be successfully received only if its achievable bits (where β is the bandwidth and τ is the time-slot duration) on the allocated RBs are greater than the minimum required bits.

- For the sake of fairness, constraints Eq. 4e require that transmitting vehicle v at time-slot t should not use more than one frequency-slot to send packet i.
- Constraints Eqs. 4f and 4g relate the variables $x^{i}_{v,w,f}(t)$ and $y^{i}_{v,w}$.
- Constraints Eq. 4h state that transmitting vehicle v should use orthogonal RBs to send both the safety and the non-safety packets.
- Constraints Eq. 4i guarantee that vehicle v should transmit its safety packet only in the range $\{s_v..e_v\}$ of time-slots.
- Constraints Eq. 4j guarantee that the transmission power of vehicle v should be zero if v does not transmit any packet.

Problem Eq. 4 is non-linear and non-convex because of the constraints Eq. 4c. We show in the next subsection that it is in addition NP-hard even for a special case that can be formulated as an integer linear program. Thus, Eq. 4 is very challenging to solve in an optimal way and it is even more difficult to solve it in a distributed manner without assuming the existence of a central coordination point.

2.3 NP-hardness

Lemma 1 *VRA is NP-hard.*

Proof We prove the lemma by restriction. We consider the following restricted version of VRA:

1. there is only the safety slice;
2. there is a single frequency-slot;
3. there are m transmitting vehicles and a single receiving vehicle w;
4. the arrival and deadline of the safety packet of vehicle v are $s_v = 1$ and $e_v = T$, respectively;
5. OMA technique is assumed; and
6. all transmitting vehicles are allocated their maximum power.

Under this restriction, VRA becomes equivalent to the following problem: maximize the number of safety packets successfully received by w while allocating time-slots to the transmitting vehicles such that (i) the minimum required rate of vehicle v is met, and (ii) the transmitting vehicles do not use the same time-slots.

We prove NP-hardness of this restricted version of VRA by reducing the maximum independent set (MIS) problem [15] to it in polynomial time.

Let an instance of MIS be given by a graph. Vertices represent transmitting vehicles while edges represent time-slots. An edge exists between two vertices if and only if one of the corresponding vehicles can be scheduled in the corresponding time-slot. We can construct the channel coefficients between transmitting vehicle v and w at time-slot t as follows: if the edge corresponding to t is incident to the vertex corresponding to vehicle v, then the channel coefficient is set to 1; otherwise, it is set to 0. The transmission power and the noise power are set to 1. The minimum number of bits required by vehicle v, $\sigma_v^{textsfs}$, is chosen equal to the degree of vertex v.

If there exists an independent set $\mathcal{IS}$ in the given graph of maximum size, then by scheduling vehicle $v \in \mathcal{IS}$ at time-slot t (for each t incident to v), we have a maximum number of scheduled vehicles. Since we have an independent set, it is true that each time-slot is used by at most one vehicle. Moreover, by construction of the channel coefficients, the rate reached by vehicle v is equal to the degree of vertex v. On the other hand, if there is a solution to the restricted version of VRA of maximum size, then, to meet the minimum number of bits required, vehicle v must be scheduled in all time-slots incident to it. The set of scheduled vehicles forms an independent set of maximum size since each time-slot is used by at most one vehicle.

In summary, MIS reduces to the restricted version of VRA in polynomial time and the latter is solved if and only if the former is solved. The proof of the lemma follows since MIS is a well known NP-hard problem [15]. □

3 Multi-agent deep reinforcement learning–based resource allocation

When vehicles operate in 5G-NR C-V2X assisted sidelink scenario, the resource allocation problem becomes very difficult to solve due to the lack of a central coordination central point acting as a network manager. Therefore, it is necessary for vehicles to solve the resource allocation problem by themselves. To this end, we use multi-agent DRL to develop an online distributed algorithm to solve VRA. The proposed algorithm is based on the well-known deep Q learning (DQL) approach [26]. DQL operates in two phases: the learning (or the training) phase and the inference (or the testing) phase. In the training phase, each agent trains a deep neural network, often referred to as deep Q network (DQN). In the inference phase, each agent, based on its observations, takes actions based on its trained DQN.

DQL is a major improvement of the table-based method called Q-learning. Q-learning works by creating a table of state-action pairs and finding the best action given a certain state. It often uses an exploration policy called the ϵ-greedy method in which an action is chosen at random with probability ϵ and is chosen to give the best reward so far otherwise. Q-learning can solve an interesting set of RL problems. However, when the state and action spaces become large (which is often the case in resource allocation problems in wireless networks), creating a Q-table and finding the best policy becomes a prohibitively complex task. In addition, many states will be very rarely visited. Moreover, the task of learning tabular Q becomes even more complex when the learning is multi-agent.

DQL is a promising approach that can be used to solve the curse of dimensionality in RL [26] by approximating the Q function instead of using a Q table. One way to solve the multi-agent problem in RL is to combine DQN with independent Q learning for each agent. In other words, each agent tries to learn its own policy based on its own observations and actions while treating all other agents as part of the environment. However, this will strongly influence the outcome of the training phase as it will create a non-stationary environment. Therefore, we will also discuss how to remove the non-stationarity problem by creating a specific state space.

Before describing the proposed DQL algorithm, we first model VRA as a multi-agent Markov decision process (MDP) defined by a state space, an action space, a reward function, and a probability transition function. Each transmitting vehicle, as an independent agent in a given state, performs an action, receives a reward, and transitions to the next state based on its interaction with the vehicle environment. This interaction with the unknown vehicle environment allows each agent to gain experiences and increase its accumulated rewards. The interaction of the agents with the environment is illustrated in Fig. 2. To maximize system performance, i.e., the objective function of VRA in Eq. 4, the agents must act cooperatively. By specifically designing the reward function, we are able to construct a multi-agent RL framework in which agents cooperatively maximize the objective function of VRA.

Mathematically, at each time-slot t, given that the environment is in the state $\mathcal{S}(t)$, each v transmitting vehicle successively (i) receives an observation $\mathcal{O}_v(t)$ from the environment, (ii) takes an action $a_v(t)$, and (iii) receives a reward $r(t+1)$. Then, the environment evolves to the next state $\mathcal{S}(t+1)$. The reward function $r(t+1)$ is independent of agent v, meaning that agents receive a common reward signal. This is designed to encourage cooperative behavior among agents. The reward function $r(t+1)$ is computed as soon as all agents take their actions, forming a common action $\mathbf{a}(t) = (a_v(t), \mathbf{a}_{-v}(t))$, where $\mathbf{a}_{-v}(t)$ denotes the actions of all vehicles but v. In the following, we describe the key elements of the multi-agent MDP.

3.1 The state space

The state $\mathcal{S}(t)$ of the vehicle environment at time-slot t is not known directly by each transmitting vehicle (called agent from now on). Instead, given the state $\mathcal{S}(t)$, each agent v receives an observation $\mathcal{O}_v(t)$. State $\mathcal{S}(t)$, which is hard to acquire by each agent due to its high mobility, includes accurate and global information about CSI of all agents and their transmission behaviors (e.g., the remaining data bits to be sent in future time-slots). However, each agent v can gain knowledge only by observing the environment using $\mathcal{O}_v(t)$. In our model, $\mathcal{O}_v(t)$ includes local CSIs and transmission behavior of v alone. Specifically, $\mathcal{O}_v(t)$ is given by the following tuple:

$$\mathcal{O}_v(t) = (\mathbf{g}_v(t), \mathbf{d}'_v(t), \mathbf{z}(t), \mathbf{l}_v(t), s_v, e_v, k, \epsilon), \tag{5}$$

where $\mathbf{g}_v(t) := (d_{v,w}(t) : w \in \mathcal{W}_v)$ represents the large-scale and small-scale fading between agent v and all other receiving vehicles $\mathcal{W}_v$. The value $\mathbf{g}_v(t)$ can be accurately estimated by the receiving vehicles and returned to each agent without significant delay [29]. Similarly, $\mathbf{d}'_v(t)$ is the distance between agent v and all other agents. The distances $\mathbf{d}'_v(t)$ help agent v observe the location of other agents (who are likely to interfere with it) and thus contribute to the required cooperative behavior. The distance $\mathbf{d}'_v(t)$ is location-based channel information that

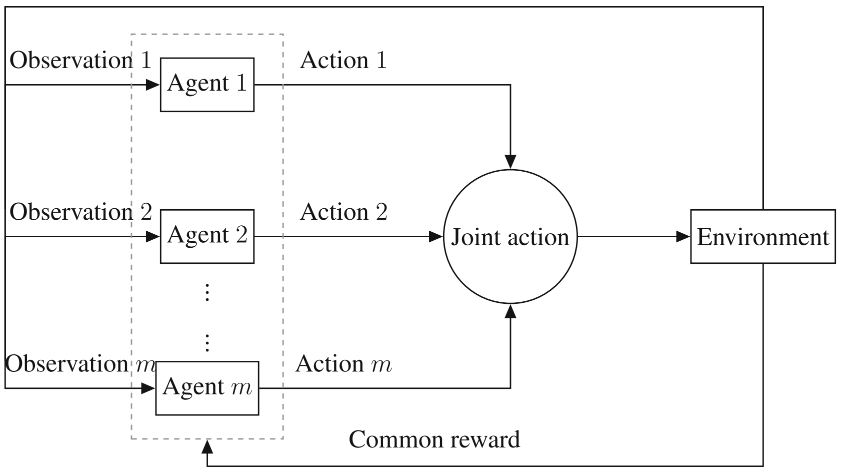

Fig. 2 The multi-agent interaction with the vehicle environment

can be accurately estimated by agent v' and can be fed back to v with a delay-free communication [29]. Therefore, it can be assumed that $\mathbf{g}_v(t)$ and $\mathbf{d}'_v(t)$ are instantly available to agent v. In addition, $\mathcal{O}_v(t)$ includes a decision variable $\mathbf{z}(t)$ that indicates whether or not the agents were transmitting in previous time-slots and if so, which packet did they transmit. The fourth observation that agent v acquires is $\mathbf{l}_v(t)$ which indicates the remaining bits of the two packets that agent v should send (e.g., initially, $l^i_v(t) = \sigma^i_v$). The observation also includes the global start time s_v and end time e_v to primarily capture the safety packet's deadline. The last two parameters that $\mathcal{O}_v(t)$ includes are the learning episode index k and the exploration rate of exploration policy ϵ-greedy. These last two parameters are used mainly to remove the non-stationarity problem that arises in multi-agent DQL [14]. In fact, DQL relies on an experience replay memory that is used to better train the DQN by creating a data set and applying batch sampling from time to time. As discussed earlier, if DQL is applied to each agent independently, non-stationarity occurs and batch sampling no longer reflects the current dynamics of the environment. In [14], the authors solved the non-stationarity problem by conditioning each agent's Q-function on a fingerprint that tracks the policies of the other agents. They showed that the agent's policy is strongly correlated with the training episode index k and the exploration rate ϵ.

3.2 The action space

The VRA problem is solved online, i.e., on a time-slot basis. At each time-slot, the agent has to make four decisions which are given as follows: (i) coverage broadcast selection, (ii) packet selection, (iii) RB allocation, and (iv) power allocation. The first decision is the selection of the set of receiving vehicles to broadcast to. The second decision is the selection of the packet to broadcast (safety or non-safety). The third decision is the selection of the frequency-slot and time-slot to be used for transmission. Finally, the fourth decision is the allocation of transmission power for transmitting vehicles. For (i), we define a set of coverages $\mathcal{C} := \{c_1, c_2, \ldots, c_\delta\} \cup \{0\}$ from which the coverage of each transmitting vehicle could be selected. Thus, if an agent chooses $c_i\mathcal{C}$ for a certain $i \in \{1, 2, \ldots, \delta\}$, then it will broadcast to all the receiving vehicles that are present in the circle of radius c_i (note that when $c_i = 0$, the transmitting vehicle will not broadcast any packet). For (ii), we define the set $\mathcal{B} := \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$ that indicates which packet of which slice the agent will transmit. The agent has four possible choices: it does not transmit, it transmits packet 1 (from slice 1), it transmits packet 2 (from slice 2), or it transmits both packets (from slices 1 and 2). For (iii), the RB allocation consists in choosing the frequency-slot f to use in the current time-slot t. The last decision is (iv) which represents the power allocation. Normally, power should be allocated from a continuous interval $[0, p^{\max}_v]$ which makes DQL implementation more complex. Nonetheless, discrete power allocation is a realistic assumption in many real systems [5, 25, 32]. Thus, to keep things simple and realistic and following the assumptions of previous work [23], we define a set $\mathcal{P} = \{p_1, p_2, \ldots, p_\ell\}$ of discrete power levels that the agent can use to transmit its packets on the allocated RB.

In summary, the action space of agent v at time-slot t is given by the set $\mathcal{A}_v(t) := \mathcal{C} \times \mathcal{B} \times \mathcal{F} \times \mathcal{P}$.

3.3 The reward function

As discussed in 5G-NR C-V2X, an important optimization objective in vehicular broadcast communication is the PRR, which is directly related to the number of successfully received packets. In VRA, we correlate the objective function in Eq. 4 with the reward function. The flexibility in the design of reward functions is what makes RL a particularly attractive approach to solving NP-hard problems like VRA.

The reward of agent v at time-slot t depends on whether the agent successfully transmits the packet or not. As long as the packet is not successfully transmitted, the reward is set to the normalized achievable rate between agent v and each of its selected receiving vehicles. (Normalization is used to make the reward less than one.) As soon as the agent successfully transmits its packet, the reward is set to 1 (the highest possible reward for agent v). Setting the reward to a number less than 1 when the agent has failed to transmit its packet helps the agent to acquire useful informations for its future decisions. Furthermore, by setting the maximum possible reward once the packet has been successfully transmitted, the agent learns the best possible outcome. Agent's v individual reward at time-slot t when transmitting packet i to w using frequency-slot f is given by the following formula:

$$\begin{cases} \frac{\beta\tau}{\Gamma} \lg(1 + \mathsf{sinr}^i_{v,w,f}(t)), & \text{if } v \text{ does not finish its transmission,} \\ 1, & \text{otherwise,} \end{cases} \tag{6}$$

where Γ is a normalization constant to make the reward less than 1.

The reward function $r(t)$ is thus the sum of all individual rewards of each agent; creating a common reward for all agents. This will encourage agents to play cooperatively while learning, in order to maximize their rewards.

The goal of the DQL is to maximize the cumulative (discounted) reward in the long run, given an initial state of the environment $\mathcal{S}(0)$. This cumulative reward is given mathematically by:

$$\sum_{\iota=0}^{\infty} \gamma^{k} r(t+\iota+1), \quad 0 \leq \gamma \leq 1, \tag{7}$$

where γ is called the discount factor.

3.4 The transition probability function

The evolution of the environment from one state to another is often modeled by a probability distribution function. It is given mathematically by the probability of moving to state $\mathcal{S}(t+1)$ and obtaining reward $r(t+1)$ given that the environment was in the state $\mathcal{S}(t)$ and that the action taken was $a(t)$, i.e., $\Pr[\mathcal{S}(t+1), r(t+1)|\mathcal{S}(t), \mathbf{a}(t)]$. This probability function depends on the highly dynamic vehicular environment (depends on channel coefficients and vehicle motion) and cannot be computed explicitly due to the complex nature of the vehicular environment.

3.5 The training phase of DQL

As discussed previously, DQL is composed of two phases: the learning phase and the inference phase. The learning phase lasts a number of episodes where each episode spans a time horizon of T time-slots. DQL uses deep neural networks to approximate the Q function. We leverage DQL with prioritized replay memory [31] and a dueling [34]. In general, experience replay memory helps agents remember and use past experiences. Standard replay memory is used to sample experience transitions uniformly, without paying attention to the importance of the sampled experiences. Note, however, that prioritized experience replay memory is proposed to pay more attention to important experiences. This indeed allows agents to learn better [31]. On the other hand, dueling [34] is proposed as a new neural network architecture that represents two estimators for the Q-function.

The proposed DQL algorithm uses a DQN with a weight vector $\mathbf{w}_v$ to represent the Q-function of each agent v. In other words, we create m DQNs; one for each agent. The input of DQN v is given by the observation $\mathcal{O}_v(t)$ obtained by observing the state of the vehicle environment $\mathcal{S}(t)$. The output of the DQN v is the value of the Q function which is given by the appropriate action taken by agent v among the set of possible actions $\mathcal{A}_v(t)$. All DQNs are trained simultaneously. Note that we assume some level of synchronization between agents, which can be achieved by a central controller (or a roadside unit) that collects agents' actions and forms the joint action to calculate the common reward function. In fact, once each agent chooses its action, it sends it to the roadside unit (RSU). Next, the RSU forms a joint action composed of agents' individual actions. It can then check if the joint action is feasible or not according to the problem constraints and can calculate the common reward function.

The learning phase lasts H episodes. In each episode, agents explore the action space using the ϵ-greedy policy. Each episode k covers a time horizon of T time-slots. At the beginning of the first time-slot, the initial state of the vehicular environment (initial distances of vehicles, etc.) is revealed to all agents. For each time-slot $t \geq 1$, each agent chooses an action, that is a tuple $a_v(t) := (c_v, b_v, f_v, p_v) \in \mathcal{A}_v(t)$, according to ϵ-greedy, where c_v is the coverage area, b_v is the packet(s) to be sent, f_v is the frequency-slot and p_v is the transmission power. Once all agents have chosen their actions, a joint action $\mathbf{a}(t)$ is formed and the reward function $r(t+1)$ is calculated at the next time-slot $t+1$. Each agent moves to the next state due to the evolution of the channel coefficients and vehicle mobility. The resulting tuple $\text{Exp}_v := (\mathcal{O}_v(t), a_v(t), r(t+1), \mathcal{O}_v(t+1))$ is called the experience of agent v and is stored in its prioritized replay memory with some associated priority. After a few episodes, a mini-batch $\mathcal{M}_v$ is sampled according to the priorities from the prioritized replay memory. This mini-batch is used to update the DQN weight parameter using a variant of the stochastic gradient descent algorithm to minimize the loss function. The loss function is given by the mean square error as follows:

$$\sum_{\text{Exp}_v \in \mathcal{M}_v} \Bigg[r(t+1) + \gamma \max_{a_v} \{ Q(\mathcal{O}_v(t+1), a_v; \mathbf{w}_v^{-}) \} - Q(\mathcal{O}_v(t), a_v(t); \mathbf{w}_v) \Bigg]^2, \tag{8}$$

where each DQN is represented mathematically by the Q function $Q(\mathcal{O}_v(t), a_v(t); \mathbf{w}_v)$ (that DQL tries to approximate) and $\mathbf{w}_v^{-}$ is the weight parameter of a duplicate copy of the original DQN (the target DQN) that is created in order to update the original DQN from time to time. The creation of a target DQN is suggested by the quasi-static target network method [26] to set the targets of the Q values. The pseudocode for the learning phase of the DQL algorithm is given in Algorithm 1.

Algorithm 1 The training phase of DQL.

```
Require: Agents and environment.
Ensure: Trained DQNs.
1: Start simulator: generate vehicles and network parameters.
2: Initialize the DQN of each agent $v$.
3: for each episode $k \leftarrow 1$ to $H$ do
4:     Reset and build the environment.    //Generate vehicles, their speeds and directions, generate network parameters and QoS requirements, etc.
5:     Anneal the exploration rate $\epsilon$ for each agent.    //Use the decayed $\epsilon$-greedy approach in Eq. 9.
6:     for each time-slot $t \leftarrow 1$ to $T$ do
7:         $t_k \leftarrow (k-1)T + t$
8:         for each agent $v$ do
9:             Observe $\mathcal{O}_v(t)$.    //The observation includes all vehicular network parameters according to Eq. 5.
10:            Choose an action $a_v(t)$ using $\epsilon$-greedy.
11:        end for
12:        Obtain the joint action $\mathbf{a}(t)$.    //The action of all agents.
13:        Find the common reward $r(t+1)$.    //The reward is designed as in Eq. 6.
14:        for each agent $v$ do
15:            Obtain the next observation $\mathcal{O}_v(t+1)$.
16:            Prioritize the experience $\mathrm{Exp}_v$.    //For any agent, its $t$th experience is assigned a priority $\pi_t$ as given in Eq. 10.
17:            Store $\mathrm{Exp}_v$ in $\mathcal{M}_v$.
18:            if $t_k$ mod $T_{\text{train}} = 0$ then
19:                Sample a mini-batch from $\mathcal{M}_v$.    //For any agent, its $t$th experience is sampled according to its probability $\Pi_t$ given in Eq. 10. We sample according to these probabilities $N_{samples}$.
20:                Do a mini-batch training.    //After constructing the dataset of samples, each agent starts the training by applying the stochastic gradient descent algorithm to minimize its loss function according to Eq. 8.
21:                Update the priorities.    //The priority of every experience $t$ is updated according to the temporal difference $\Delta_t$ as in Eq. 11.
22:            end if
23:            if $t_k$ mod $T_{\text{target}} = 0$ then
24:                Update the target DQN of $v$.
25:            end if
26:        end for
27:    end for
28: end for
```

The training phase of DQL requires as input the vehicular environment which includes vehicles, channel coefficients, packet requirements, available RBs and any other relevant network parameters. It returns as output trained DQN of each agent. First, DQL generates all the vehicular network parameters, and then it initializes the weights of each DQN. Then, it iterates the episodes. For each episode k, the vehicular environment is constructed by (i) updating the network parameters, e.g., the remaining bits of each agent are updated based on the previous episodes, and (ii) moving the vehicles according to the mobility model. Then, the exploration rate ϵ is annealed based on the episode index. The annealing of the exploration rate over time is a technique used in RL to solve the dilemma between exploration and exploitation, i.e., over time we decrease ϵ to increase the probability of exploitation when the agent starts to learn something useful. To perform annealing, we use the decayed ϵ-greedy algorithm which works as follows. Let ϵ_{start} be the initial value of ϵ (before the first episode) and let ϵ_{end} be the final value of ϵ. Let H_k be the number of learning episodes after which the annealing stops. Thus, the value of ϵ is updated (annealed) according to:

$$\begin{cases} \epsilon \leftarrow (\epsilon_{\text{start}} - \epsilon_{\text{end}})(1 - k/H_k) + \epsilon_{\text{end}}, & \text{if } k < H_k \\ \epsilon \leftarrow \epsilon_{\text{end}}, & \text{otherwise.} \end{cases} \tag{9}$$

After H_k episodes, the value of ϵ is no longer decreased and is set to ϵ_{end}.

Once all agents have chosen their actions based on the annealed ϵ as in Eq. 9, the joint reward is computed by each receiving vehicle through the physical sidelink feedback channel (PSFCH) [3]. Specifically, a receiving vehicle calculates the received sinr (and thus the achievable data rate) and finds out how many bits a particular agent is transmitting. Then, it broadcasts this information. In this way, all agents can acquire the joint action $\mathbf{a}(t)$ at any time t. To guarantee some level of synchronization, the joint action can be obtained by an RSU that collects all individual actions. Once the reward signal is received by all agents, learning begins. Then, the environment moves to the next state according to vehicle mobility and channel variations. Each agent then adds its experience $\mathrm{Exp}_v(t)$ to its prioritized replay memory. Initially, agents assign random priorities to their experiences, but the priorities change as agents begin to learn and update the parameters of their DQNs. Specifically, each agent's t experience is assigned a priority π_t (the agent index is omitted for simplicity). Then, experience t is sampled from each agent's replay buffer with a probability Π_t given by [31]:

$$\Pi_t := \frac{\pi_t^\alpha}{\sum_{t'} \pi_{t'}^\alpha}. \tag{10}$$

We use the proportional prioritization approach where $\pi_t = |\Delta_t| + \mu$, with Δ_t represents the time difference error of experience t and μ is a small positive constant used to avoid the limiting case of transitions that are not revisited once their error is zero. Finally, importance sampling is used to eliminate any possible bias introduced by prioritization. This is achieved by introducing a weight called the importance sampling weight of experience t given by $w_t := (E \cdot \Pi_t)^{-\theta}$ with E being the number of experiences in the replay buffer.

Once every T_{train}, each agent samples a mini-batch of size N_{samples} from its prioritized replay memory. This prioritized replay memory forms a dataset that each agent uses to perform learning. Indeed, each agent uses a well-known variant of stochastic gradient descent to minimize the mean square error (or loss) which is given by Eq. 8. In Eq. 8, the term $r(t+1) + \gamma \max_{a_v}\{Q(\mathcal{O}_v(t+1), a_v; \mathbf{w}_v^-)\}$ denotes the target value that each DQN v tries to reach or adjust. Then, each agent updates the priorities of the sampled experiences in proportion to the value of the loss. In other words, the priority of experience t is updated as $\pi_t = |\Delta_t| + \mu$ where Δ_t is the time difference error and is defined as follows:

$$\Delta_t = r(t) + \gamma \max_a\{Q(\mathcal{O}(t), a; \mathbf{w}^-)\} - Q(\mathcal{O}(t-1), a(t-1); \mathbf{w}), \tag{11}$$

where agent's index is omitted. Finally, once every T_{target} each trained DQN is copied into the target DQN.

3.6 The inference phase of DQL

The inference phase of DQL is as follows. First, the trained DQNs (their weight parameters) are loaded. Similarly, the annealed ϵ is loaded from the last training episode (the episode index is also revealed). Then, for each episode, which now represents a random realization of the channel, the environment is reset and built—initializing the network parameters and transmission behaviors of each agent. Then, for each time-slot, each agent v, after observing the environment, chooses the best action which is given by the maximum value of its Q function approximated by its DQN. Once all agents have chosen their actions, a common action is formed by the feedback from the receiving vehicles to the transmitting vehicles. Then, the common reward is obtained and the next episode begins.

The inference phase of DQL is an online distributed algorithm that is run in each time-slot to find the best possible action to select without knowing the future observation. The learning phase is the most computationally intensive task in DQL. It is executed for a large number of episodes and can be performed offline with different channel conditions and network topologies. The pseudo-code for the inference phase of DQL is given in Algorithm 2.

Algorithm 2 The inference phase of DQL.

```
Require: The trained DQNs.
Ensure: Resource allocation solution of VRA.
 1: Load the DQNs—one DQN per agent.   //After training and saving DQNs' weights, they are loaded to use and apply to learned policy.
 2: for each episode $k \leftarrow 1$ to $H$ do
 3:     Reset and build the environment.   //Similar to the training phase, we generate vehicles, their speeds and directions, and we generate network parameters and QoS requirements, etc.
 4:     Get the annealed $\epsilon$ for each agent.   //Obtain the saved decayed $\epsilon$ value.
 5:     for each time-slot $t \leftarrow 1$ to $T$ do
 6:         for each agent $v$ do
 7:             Observe $\mathcal{O}_v(t)$.   //The observation includes all vehicular network parameters according to Eq. 5.
 8:             Choose $a_v(t)$ that maximizes the Q function.   //Choose the action $a_v = \arg\max_a Q(\mathcal{O}_v, a; \mathbf{w}_v)$.
 9:         end for
10:         Obtain the joint action $\mathbf{a}(t)$.
11:         Find a solution to VRA.   //Based on the chosen action, calculate the objective function in Eq. 4a.
12:     end for
13: end for
```

Note that the complexity of DQL is dominated by training. This is due to the computationally intensive tasks that must be performed offline to learn the optimal policy. Nevertheless, once learning is done offline, DQL can be used online to act suboptimally according to the learned policy. The complexity of DQL depends on many parameters, namely the learning time of DQNs, the computation time to obtain the reward of each agent, the buffer sampling time, etc. To quantify the worst-case time complexity, we proceed as follows. The most computationally expensive instructions in DQL are written in lines 13, 19, and 20 of Algorithm 1. Let T_{13}, T_{19}, and T_{20} be the worst-case computation time of these lines, respectively. The worst-case time T_{13} corresponds to the worst-case time to compute the reward, which mainly depends on the SINR computation. Thus, at each time-slot t, $T_{13} = \mathcal{O}(m^2nF)$ according to the SINR equation in (1). The worst-case time T_{19} is related to the sampling of a set of experiences in the mini-batch. It is mainly proportional to the size of the sampled mini-batch. Thus, $T_{19} = \mathcal{O}(N_{\text{samples}})$, where N_{samples} represents the number of sampled experiences. Finally, the time complexity of line 20 in the worst case, T_{20}, depends mainly on the training performed. We have a mini-batch of size N_{samples} that is trained using the ADAM optimizer. Thus, T_{20} depends mainly on the ADAM optimizer, the training rate used, and the number of samples N_{samples}. Nevertheless, the value of T_{20} cannot be given explicitly. Overall, DQL has a complexity of $\mathcal{O}(HT(T_{13} + mN_{\text{samples}}T_{20})) = \mathcal{O}(HT(m^2nF + mN_{\text{samples}}T_{20}))$, where H and T are respectively the number of episodes and the number of time-slots.

4 Simulation results

In this section, we validate the proposed resource allocation method. We present simulation results to illustrate the performance of the proposed DQL algorithm. The simulation setup is based on the road configuration for the highway scenario detailed in 3GPP TR 37.885 [2] and also used, to name a few, in [22]. We consider a multilane highway with a total length of 2 km where each lane is 4 m wide. There are a total of six lanes—three for the forward direction (vehicles move from right to left) and three for the reverse direction (vehicles move from left to right). Transmitting and receiving vehicles are generated in the vehicular environment by a spatial Poisson process. The vehicle speed determines the vehicle density and we consider an average inter-vehicle distance (in the same lane) of $2.5 \times V$ [13] where V is the absolute vehicle speed. The speed of a vehicle depends on the lane it is in: the ith forward lane (top to bottom with $i \in \{1, 2, 3\}$) is characterized by the speed of $60 + 2(i-1) \times 10$ km/h, while the ith backward lane (top to bottom with $i \in \{1, 2, 3\}$) is characterized by the speed of $100 - 2(i-1) \times 10$ km/h. The number of transmitting vehicles m and receiving vehicles n is chosen randomly from the generated vehicles. The important simulation parameters are given in Table 1. In Fig. 3, we take a snapshot of the initial vehicle topology at the beginning of the first time-slot where the vehicles are randomly chosen according to the spatial Poisson process.

Unless otherwise specified, the packet requirements of the non-safety slice are generated uniformly at random in $\{0.1..1\}$ Mbit. The packet requirements of the safety slice are equal to 1200 bytes. Each agent can choose a coverage (in m) from the set $\mathcal{C} = \{100, 200, 400, 800, 1000, 1200, 1400\} \cup \{0\}$. The power levels (in dBm) are given by the set $\mathcal{P} = \{5, 10, 15, 20, 23, 27, 30\} \cup \{-100\}$ where -100 dBm is used to indicate that the corresponding agent will not transmit its packets (similar to the zero coverage).

We train three different vehicular networks for a total time horizon of 100 ms. The first one is called (2,2,1,5)-network and consists of $m = 2$ agents (transmitting vehicles), $n = 2$ receiving vehicles, $F = 1$ frequency-slots and $T = 5$ time-slots, each with a duration of 20 ms. The second is called (5,4,2,10)-network and consists of $m = 5$ agents, $n = 4$ receiving vehicles, $F = 2$ frequency-slots and $T = 10$ time-slots, each with a duration of 10 ms. The third is called (6,4,4,20)-network and consists of $m = 6$ agents, $n = 4$ receiving vehicles, $F = 4$ frequency-slots and $T = 20$ time-slots, each with a duration of 5 ms. We use the first small network for optimal comparison purposes. We implement the mixed integer non-linear programming (MINLP) formulation given in Eq. 4 in the Julia programming language [7] using Juniper [19] package. Juniper is a MINLP solver written in Julia that solves MINLPs using NLP solvers and then branch-and-bound

Table 1 Vehicular network parameters

Parameter	Value
Carrier frequency	2 GHz
Bandwidth per RB	1 MHz
Vehicle antenna height	1.5 m
Vehicle antenna gain	3 dBi
Vehicle receiver noise figure	9 dB
Shadowing distribution	Log-normal
Fast fading	Rayleigh fading
Pathloss model	LOS in WINNER + B1 [20]
Shadowing standard deviation	3 dB
Road configuration	Highway road configuration [2]
Vehicle drop model	Spatial Poisson process
Noise power N_0	−114 dBm

(BnB) based solvers (MIP solvers). In our implementation, we used Ipopt as the NLP solver and CPLEX as the MIP solver. DQNs are created and trained in Julia using Flux.jl [16] machine learning library. Each DQN consists of an input layer and an output layer and three fully connected hidden layers containing 500, 350, and 260 neurons respectively. The rectified linear unit activation function (ReLU) given by $f(x) = \max\{0, x\}$ is used in each layer. Each DQN is trained with the ADAM optimizer [18] with a learning rate of 10^{-5}. The training lasts $H = 3000$ episodes with an exploration rate starting at $\epsilon_{\text{start}} = 1$ and annealed to $\epsilon_{\text{end}} = 0.02$ for the 80% of the episodes (i.e., from episode $k > 2400$, the exploration rate is fixed at 0.02). The target update frequency and training frequency (T_{target} and T_{train}) are set according to the trained network, so that every 10 episodes, DQL performs a mini-batch training and every 100 episodes, it performs a target DQN update. The size of the mini-batch is chosen to be large and equal to 2000 [33] and the experiment priority update parameters are set to $\alpha = 0.6$, $\theta = 0.4$, and $\mu = 0.001$ [31]. The channel coefficients (including slow and fast fading) change over time at each stage of the learning phase. The requirements and deadlines of the safety packets as well as the requirements of the non-safety packets are fixed in the training phase and change in the inference phase to validate the robustness of our proposed approach.

As discussed earlier, to the best of our knowledge, there is no current research work that solves VRA while considering network slicing selection, coverage selection, RB and power allocation. Due to the lack of comparisons, we adopt the following benchmarks to compare our proposed algorithm. We implement three benchmarks: two are based on NOMA technique and one is based on OMA technique. The partial idea of all benchmarks comes from [13, 37] which is based on transforming the RB allocation problem in VRA into a matching problem. Then, a matching (or rotation) algorithm is adopted. All benchmarks are centralized and use randomization and offline decisions contrary to the distributed and online nature of DQL. They are called OMA-MP, NOMA-MP, and NOMA-RP. In the case of OMA-MP, each RB is used by no more than one vehicle to meet OMA constraints and the vehicles transmit with their maximum transmission power. In the NOMA-MP and NOMA-RP, each RB can be used by any vehicle and the vehicles transmit with their maximum transmission power and with random transmission power, respectively. Coverage and slot selection are performed randomly at the beginning of the time horizon. The allocation of RBs to vehicles is done in a similar way in all benchmarks [13, 37]. First, an initial RB allocation is performed that finds the RB that yields the highest sum of channel power gain between a transmitting vehicle v and its corresponding target receiving vehicle in $\mathcal{W}_v$. The sum criterion on receiver vehicles is adapted to our V2V broadcasting case and is used to maximize the number of receiving vehicles that have high channel power gains. Once the initial allocation is obtained, a swap match is performed to improve the number of successfully received packets. If no swap improves the matching, the algorithm terminates; otherwise, the algorithm continues swapping operations (see [37] for more details).

Figure 4 shows the convergence of DQL as a function of learning episodes. It shows the cumulative average rewards per episode where the average is taken over the last 200 episodes. It is clear that the average rewards improve as the training episodes increase. This shows the effectiveness of the proposed algorithm. We can see that DQL converges gradually, starting at episode number 2700. Note that the convergence of the algorithm is not smooth and contains some fluctuations which are mainly due to the high mobility of the vehicular environment. Based on

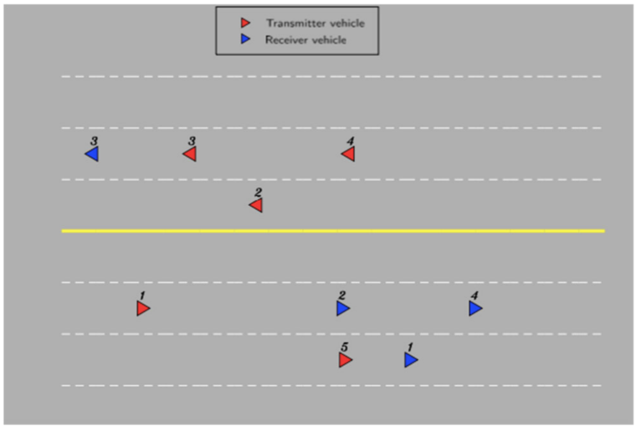

Fig. 3 A snapshot of the vehicle topology at the beginning of the first time-slot with five transmitter vehicles and four receiver vehicles

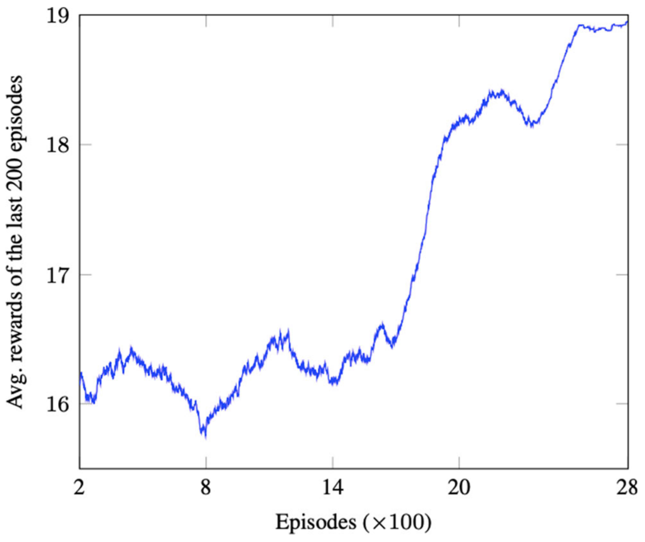

Fig. 4 Training rewards

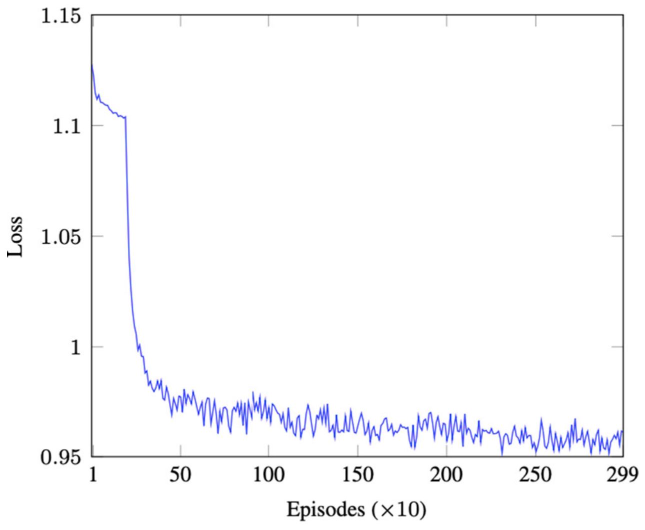

Fig. 5 Training loss

Fig. 4, we concluded that the DQNs should be trained offline for 3000 episodes, as discussed earlier, to provide some convergence guarantees.

In Fig. 5, we show the convergence of the loss function used to train each DQN network as shown in Eq. 8. It is clear that the loss decreases to a small value, indicating that the training improves with episode iteration. This makes the proposed multi-agent DQL algorithm important from a practical point of view as it provides some convergence guarantees.

Figure 6 shows the performance of DQL against the optimal algorithm OPT. OPT is obtained with the Julia-based Juniper package using the Ipopt NLP and CPLEX MIP solvers. It is clear that OPT outperforms all the algorithms since it is able to obtain the global optimal solution. However, due to the high complexity of OPT, its implementation in real vehicular network scenarios would be very impractical and that is why Fig. 6 is only obtained for the small network instance of (2,2,1,5)-network. Fortunately, DQL achieves the best performance compared to other heuristics while being online and distributed. We can also see that DQL is more than 60% close to OPT, which illustrates its outstanding performance.

Figure 7 shows the performance of DQL compared to the benchmarks by varying the packet size of slice 2. We can see that DQL manages to deliver more packets despite the fact that it is distributed and does not have the full and future CSI as in the benchmarks. We can see that NOMA-MP also achieves good performance and has the highest performance among all benchmarks. NOMA-RP achieves the lowest performance, as expected. In addition, DQL achieves a higher number of successfully delivered safety packets compared to non-safety packets. This is especially important in V2V communication because safety packets must have a higher priority to be delivered. We conclude that DQL achieves good performance compared to the benchmarks despite being distributed and runs online without seeing the future.

Figure 8 shows the performance of DQL compared to the benchmarks when varying the deadlines of slice 2 packets. DQL always achieves the best performance when the safety packet deadlines increase. The gap between DQL and the other benchmarks further improves as the deadlines

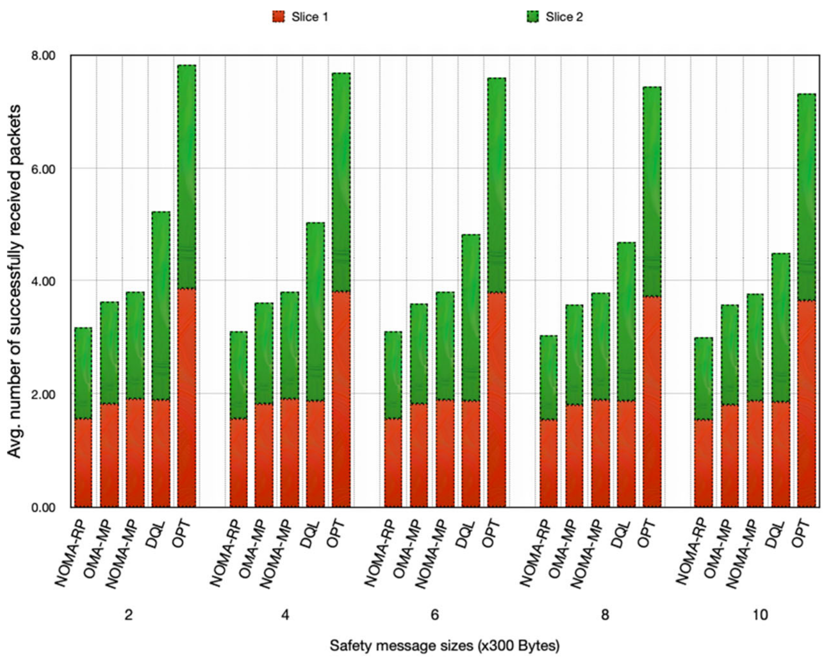

Fig. 6 Impact of safety message sizes for the (2,2,1,5)-network

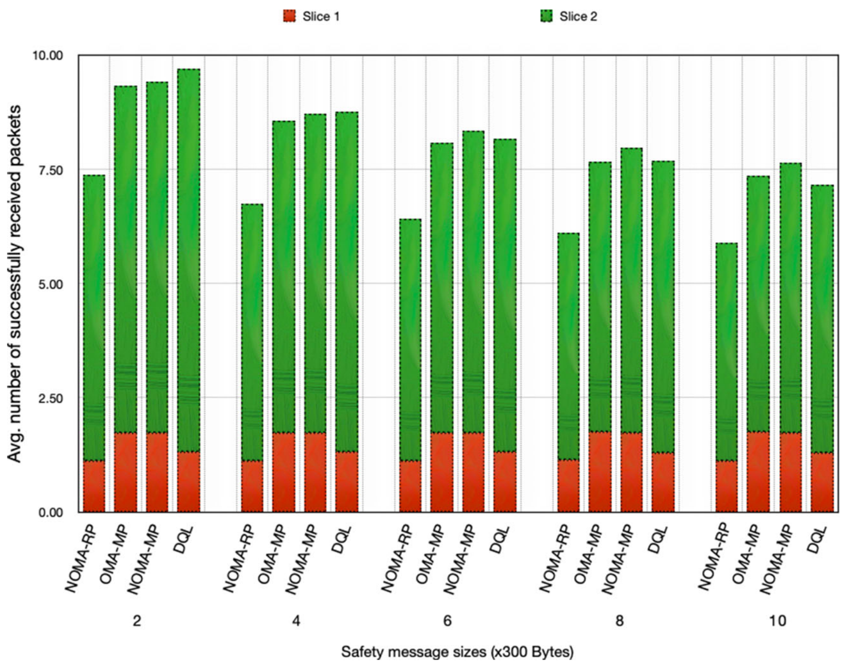

Fig. 7 Impact of safety message sizes for the (5,4,2,10)-network

increase. We also notice that NOMA-RP has the worst performance among all algorithms, indicating the need for an appropriate power allocation method in the considered NOMA resource allocation problem.

In Fig. 9, we present the effect of the size of non-safety messages on the performance of DQL. We can see that even if the size of the non-safety packets changes, DQL still performs better compared to other algorithms. This illustrates the effectiveness of the proposed multi-agent method and shows its robustness. Of course, if the size of the non-safety packets increases further, the number of successfully delivered packets will decrease since the safety packets must also be delivered. We conclude that there exists some form of fairness between the safety and non-safety slices that should be carefully studied, which we will do in our future work.

In Fig. 10, we show the performance of DQL in the case of a larger vehicle network, i.e., the (6,4,4,20)-network, with stricter latency requirements (more agents, more RBs and shorter time-slot). Indeed, this figure shows that even

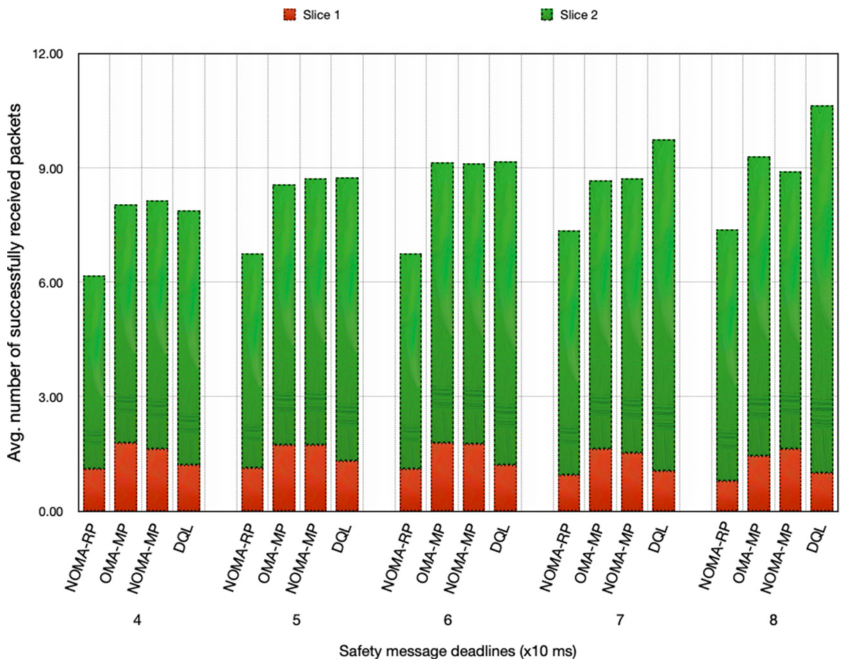

Fig. 8 Impact of safety message deadlines for the (5,4,2,10)-network

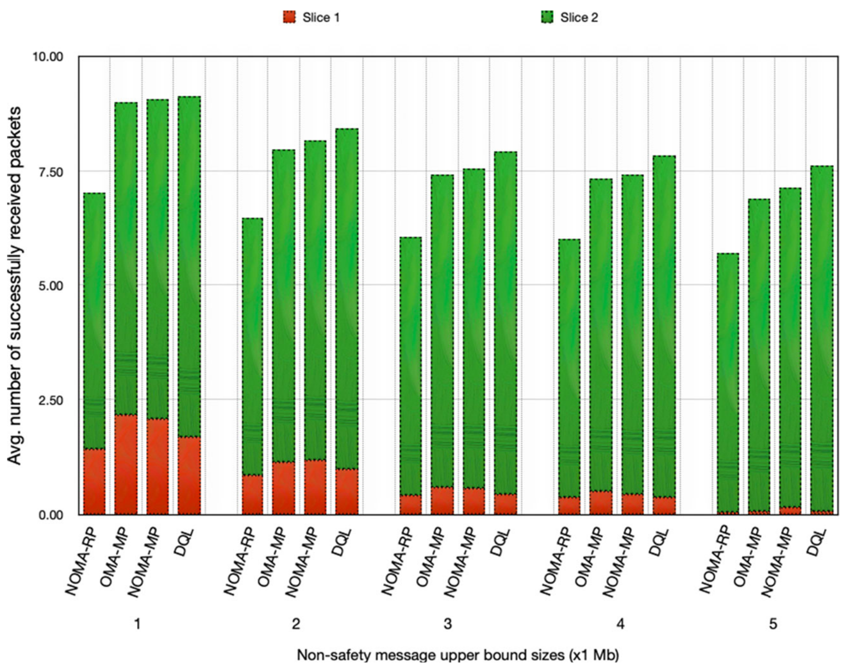

Fig. 9 Impact of non-safety message sizes for the (5,4,2,10)-network

after significant system modifications, the proposed DQL multi-agent method is robust and able to guarantee better performance compared to unrealistic centralized benchmarks. We note that a larger vehicle network allows the non-safety slice to deliver more non-safety packets because more resources are available. Thus, DQL better optimizes the utilization of individual RBs to deliver more packets.

Figure 11 shows the average number of packets delivered by each agent when the size of the non-safety packets varies. Since the non-safety packet requirements are low, each agent delivers almost the same number of packets (except for vehicle v_1, all other vehicles deliver almost 3 packets). This is beneficial since all agents will have almost the same load and thus DQL provides a load balanced solution. As the size of the non-safety packets increases, the requirements are more stringent, and thus to maximize the total number of packets successfully delivered, each DQL agent will deliver a different number of packets, thus unbalancing their loads.

In Fig. 12, we present the average DQL sojourn time obtained by each receiving vehicle under the large vehicle

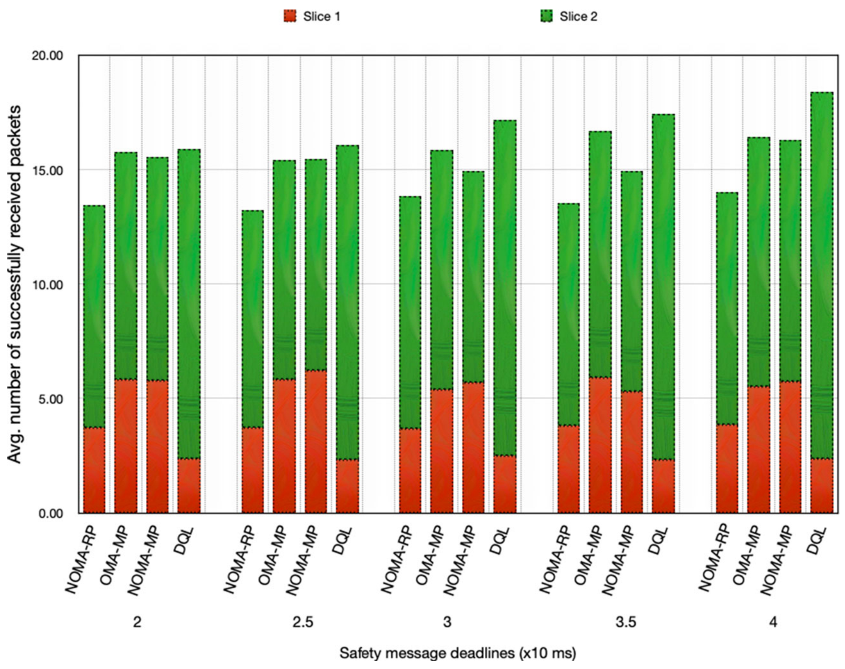

Fig. 10 Impact of safety message deadlines for the (6,4,4,20)-network

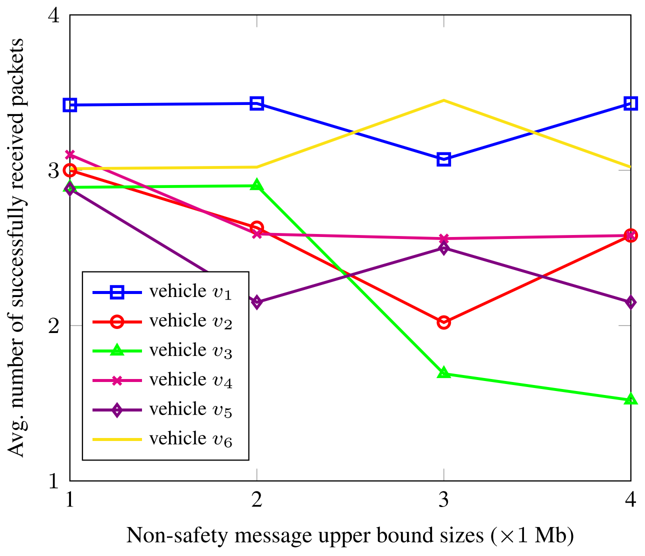

Fig. 11 Individual number of packets for the (6,4,4,20)-network

network scenario (i.e., the (6,4,4,20)-network). Sojourn time is a metric used in mobile networks to measure "the expected duration of stay of a mobile node in a particular serving cell". In our system model, we compute the sojourn time with respect to each receiving vehicle w as the expected time that w remains in the coverage of a particular transmitting vehicle. In other words, once an agent chooses a coverage area in which it communicates with w, the time will be incremented until the first time w is no longer served by that agent. This provides insight into the handoff that receiving vehicles experience. We see in Fig. 12 that three out of four receiving vehicles have an average sojourn time of about 20%. Over a time horizon of 100 ms, each receiving

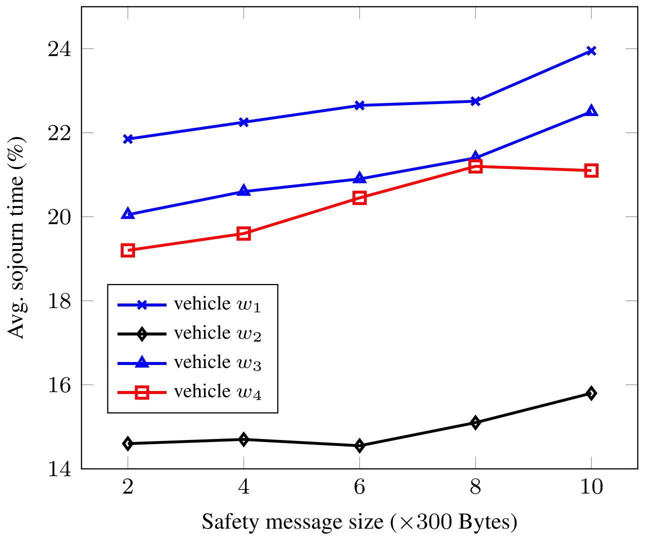

Fig. 12 Impact of safety message sizes on sojourn time for the (6,4,4,20)-network

vehicle stays in communication with the same transmitting vehicle on average 20 ms. This is due to the high mobility scenario considered.

In order to increase the sojourn time of each receiving vehicle, it is necessary for our proposed DRL framework to undergo major changes: to develop a two-time-scale DQL algorithm that acts successively in two different time scales. That is, in the first time scale (of the order of 100 ms–time horizon), the agents must choose their coverage communication ranges. Then, they must choose the slice selection decision, RB and power allocation in a shorter time-slice scale (on the order of 5 ms). This can be done by implementing two DQNs (for each agent) that are connected in series. The study of this alternative is left for our future work.

5 Conclusions

In this paper, we have developed an online distributed scheme to solve the problem of network slicing, coverage selection, resource block and power allocation in vehicular networks using non-orthogonal multiple access. We provided a mathematical programming formulation of the problem and an explicit NP-hardness analysis. We then modeled the problem as a multi-agent Markov decision process to develop a multi-agent deep reinforcement learning algorithm that is based on the well-known fingerprinting method known as deep Q learning (DQL). Our DQL algorithm utilized the most recent advances in deep reinforcement learning, including the duel architecture and prioritized experience replay memory. The proposed DQL algorithm has been shown to be robust and effective against various system parameters, including the high mobility characteristics of vehicular networks. It also outperformed baseline benchmark algorithms that are based on global and centralized decisions. In our future work, we will investigate the hierarchical two-time scale approach to solve the problem by proposing a different deep network architecture. In addition, we will consider the case of multiple slices rather than two and study the fairness between different slices.

References

1. 3GPP (2017) Evolved universal terrestrial radio access (E-UTRA); physical channels and modulation. technical specification (TS) 36.211, 3rd generation partnership project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2425. Version 14.2.0
2. 3GPP (2019) Study on evaluation methodology of new vehicle-to-everything (V2X) use cases for LTE and NR. technical report (TR) 37.885, 3rd generation partnership project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3209. Version 15.3.0

3. 3GPP (2019) Study on NR vehicle-to-everything (V2X). technical report (TR) 38.885, 3rd generation partnership project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3497. Version 16.0.0
4. Abida S, Jian PL, Muhammad AS, Gunasekaran M, Seifedine K, Abdul B, Muhammad AK (2021) A dynamic clustering technique based on deep reinforcement learning for internet of vehicles. J Intell Manuf 32:757–768. https://doi.org/10.1007/s10845-020-01722-7
5. Altman E, Avrachenkov K, Miller G, Prabhu B (2007) Discrete power control: cooperative and non-cooperative optimization. In: Proc. IEEE INFOCOM, Anchorage, AK, USA, pp 37–45. https://doi.org/10.1109/INFCOM.2007.13
6. Ashraf SA, Blasco R, Do H, Fodor G, Zhang C, Sun W (2020) Supporting vehicle-to-everything services by 5G new radio release-16 systems. IEEE Commun Standards Mag 4(1):26–32. https://doi.org/10.1109/MCOMSTD.001.1900047
7. Bezanson J, Edelman A, Karpinski S, Shah VB (2017) Julia: a fresh approach to numerical computing. SIAM Rev 59(1):65–98. https://doi.org/10.1137/141000671
8. Boukhalfa F, Hadded M, Muhlethaler P, Shagdar O (2021) Performance Evaluation of an Active Sig- naling based Time-Slot Scheduling Scheme for Connected Vehicles. Springer Ann. Telecommun. 76:363–374. https://doi.org/10.1007/s12243-020-00818-8
9. Campolo C, Molinaro A, Iera A, Fontes RR, Rothenberg CE (2018) Towards 5G network slicing for the V2X ecosystem. In: Proc. IEEE NetSoft, pp 400–405. https://doi.org/10.1109/NETSOFT.2018.8459911
10. Campolo C, Molinaro A, Romeo F, Bazzi A, Berthet AO (2019) Full duplex-aided sensing and scheduling in cellular-v2x mode 4. In: Proc. ACM mobihoc workshop on TOP-cars, pp 19–24. Catania, Italy. https://doi.org/10.1145/3331054.3331549
11. Dai L, Wang B, Ding Z, Wang Z, Chen S, Hanzo L (2018) A Survey of non-orthogonal multiple access for 5G. IEEE Commun Surveys Tuts 20(3):2294–2323. https://doi.org/10.1109/COMST.2018.2835558
12. Di B, Song L, Li Y, Han Z (2017) V2X Meets NOMA: non-orthogonal multiple access for 5G-enabled vehicular networks. IEEE Wireless Commun 24(6):14–21. https://doi.org/10.1109/MWC.2017.1600414
13. Di B, Song L, Li Y, Li GY (2017) Non-orthogonal multiple access for high-reliable and low-latency V2X communications in 5G systems. IEEE J Sel Areas Commun 35(10):2383–2397. https://doi.org/10.1109/JSAC.2017.2726018
14. Foerster J, Nardelli N, Farquhar G, Afouras T, Torr PHS, Kohli P, Whiteson S (2017) Stabilising experience replay for deep multi-agent reinforcement learning. In: Precup D, Teh YW (eds) Proc. PMLR ICML, vol. 70, pp 1146–1155, Sydney, Australia
15. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman & Co., New York
16. Innes M (2018) Flux: elegant machine learning with julia. J Open Source Softw. https://doi.org/10.21105/joss.00602
17. Khan H, Luoto P, Samarakoon S, Bennis M, Latva-Aho M (2019) Network slicing for vehicular communication. Trans Emerg Telecommun Technol e3652. https://doi.org/10.1002/ett.3652
18. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: Proc. ACM ICLR, San Diego, CA, USA, pp 1–15.
19. Kröger O., Coffrin C, Hijazi H, Nagarajan H (2018) Juniper: an open-source nonlinear branch-and-bound solver in Julia. In: Integration of constraint programming, artificial intelligence, and operations research. Springer International Publishing, pp 377–386
20. Kyosti P et al (2007) WINNER II channel models: document IST-4-027756 WINNER II D1.1.2 V1.2. Tech. rep. https://pdfs.semanticscholar.org/dd24/b09f0f95367ce73a2a1445445a802556ac5e.pdf
21. Liang L, Kim J, Jha SC, Sivanesan K, Li GY (2017) Spectrum and power allocation for vehicular communications with delayed CSI feedback. IEEE Wireless Commun Lett 6(4):458–461. https://doi.org/10.1109/LWC.2017.2702747
22. Liang L, Li GY, Xu W (2017) Resource allocation for D2D-enabled vehicular communications. IEEE Trans Commun 65(7):3186–3197. https://doi.org/10.1109/TCOMM.2017.2699194
23. Liang L, Ye H, Li GY (2019) Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. IEEE J Sel Areas Commun 37(10):2282–2292. https://doi.org/10.1109/JSAC.2019.2933962
24. Liang L, Ye H, Yu G, Li GY (2020) Deep-learning-based wireless resource allocation with application to vehicular networks. Proc IEEE 108(2):341–356. https://doi.org/10.1109/JPROC.2019.2957798
25. Liu Y, Fang X, Xiao M (2018) Discrete power control and transmission duration allocation for self-backhauling dense mmwave cellular networks. IEEE Trans Commun 66(1):432–447. https://doi.org/10.1109/TCOMM.2017.2757017
26. Mnih V et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236
27. Naik G, Choudhury B, Park J (2019) IEEE 802.11bd & 5G NR V2X: Evolution of radio access technologies for V2X communications. IEEE Access 7:70169–70184. https://doi.org/10.1109/ACCESS.2019.2919489
28. Narasimha M, Pinheiro AL (2020) Connected automobiles, chap. 9. Wiley, New York, pp 235–262. https://doi.org/10.1002/9781119514848.ch9. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119514848.ch9
29. Nasir YS, Guo D (2019) Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. IEEE J Sel Areas Commun 37(10):2239–2250. https://doi.org/10.1109/JSAC.2019.2933973
30. Ren M, Zhang J, Khoukhi L, Labiod H, Vèque V (2021) A Review of Clustering Algorithms in VANETs. Springer Ann. Telecommun. https://doi.org/10.1007/s12243-020-00831-x
31. Schaul T, Quan J, Antonoglou I, Silver D (2015) Prioritized experience replay. arXiv:1511.05952
32. Shan F, Luo J, Wu W, Li M, Shen X (2015) Discrete rate scheduling for packets with individual deadlines in energy harvesting systems. IEEE J Sel Areas Commun 33(3):438–451
33. Smith SL, Kindermans PJ, Le QV (2018) Don't decay the learning rate, increase the batch size. In: ICLR. https://openreview.net/forum?id=B1Yy1BxCZ
34. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N (2016) Dueling network architectures for deep reinforcement learning. In: Balcan MF, Weinberger KQ (eds) Proc. PMLR ICML, vol. 48, pp. 1995–2003, New York, New York, USA
35. Ye H, Li GY, Juang BHF (2019) Deep reinforcement learning based resource allocation for V2V communications. IEEE Trans Veh Technol 68(4):3163–3173. https://doi.org/10.1109/TVT.2019.2897134
36. Ye H, Liang L, Ye Li G, Kim J, Lu L, Wu M (2018) Machine learning for vehicular networks: Recent advances and application examples. IEEE Veh Technol Mag 13(2):94–101. https://doi.org/10.1109/MVT.2018.2811185
37. Zeng M, Yadav A, Dobre OA, Poor HV (2019) Energy-efficient joint user-RB association and power allocation for uplink

hybrid NOMA-OMA. IEEE Internet Things J 6(3):5119–5131. https://doi.org/10.1109/JIOT.2019.2896946

38. Zhang J, Guo H, Liu J (2019) A reinforcement learning based task offloading scheme for vehicular edge computing network. In: Han S., Ye L., Meng W. (eds) Artificial intelligence for communications and networks, pp 438–449. Springer International Publishing. https://doi.org/10.1007/978-3-030-22971-9_38
39. Zhang X, Peng M, Yan S, Sun Y (2020) Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications. IEEE Internet Things J 7(7):6380–6391. https://doi.org/10.1109/JIOT.2019.2962715