



A Reinforcement Q-Learning-based Resource Sharing Mechanism for V2X slicing Networks

Anas Nawfel SAIDI¹ and Mohamed LEHSAINI²

¹ STIC Laboratory, Tlemcen University, Algeria
anasnawfel.saidi@univ-tlemcen.dz

² STIC Laboratory, Computer Science Department
Tlemcen University, Algeria

Abstract. Network slicing has emerged as a transformative technology, offering the possibility of coexisting with multiple services with different Quality of Service (QoS) requirements within the same infrastructure. The main challenge of vehicle-to-everything (V2X) network slicing lies in developing an effective resource management approach. This approach should provide an adequate balance between optimizing the use of resources and maintaining isolation between slices. One of the benchmark approaches used in the network slicing environment is strict slicing, which allocates a fixed proportion of the whole resource pool to each slice throughout its lifetime. However, one of the limitations of this approach is the inefficiency of resource utilization, as each slice may not utilize its resources 100% during its lifetime. In this paper, we propose a flexible resource sharing mechanism based on deep reinforcement Q-learning (QDRL-based resource sharing). This mechanism triggers sharing between slices when there is an overloaded slice in the system while maintaining high isolation. Experimental results show that our solution is effective in terms of improving resource utilization and minimizing the blocking probability of new calls and the handover dropping probability.

Keywords: Deep reinforcement learning, Resource sharing, V2X, Networks slicing, 5G

1 Introduction

Since the dot-com bubble burst in 2000, technology has evolved dramatically and had a huge impact on our daily lives and civilizations. The widespread adoption of broadband internet has ushered in a new era in personal technology, with more than half the world's population now enjoying internet access. Over the past two decades, technological advances have included the introduction of WiFi 6, the rollout of 5G networks and the increasing prevalence of Internet of Things (IoT) devices.

Global efforts to move beyond 4G have been driven by the explosive growth of data-intensive mobile applications (such as online gaming and live streaming) and the IoT. The fifth-generation network, or 5G, is being promoted as a leading

© The Author(s) 2024

C. A. Kerrache et al. (eds.), *Proceedings of the International Conference on Emerging Intelligent Systems for Sustainable Development (ICEIS 2024)*, Advances in Intelligent Systems Research 184,

https://doi.org/10.2991/978-94-6463-496-9_23

technology capable of meeting the throughput, latency, reliability, capacity and mobility needs of a wide range of next-generation vertical applications, while delivering high QoS [1].

Network slicing (NS) is seen as a paradigm that can enable 5G technology to handle the heterogeneity of different 5G services. Network slicing involves dividing a single 5G physical network into multiple isolated logical networks of different sizes and configurations suitable for different types of services [2]. This allows 5G operators to create different service levels for different verticals, enabling them to customize their operations accordingly [3]. Each individual network slice has the ability to manage resources for its users independently of other slices [4].

The emergence of V2X services (e.g. traffic safety and efficiency, autonomous driving, remote diagnostics and management) has led academia and industry to adopt network slicing as a leading solution to address the diversity of vehicle-to-everything requirements. Indeed, the wide range of vehicle use cases raises the need for four types of V2X network slices dedicated to each vehicle use case [5]. These four types include: the autonomous driving slice, the remotely controlled driving slice, the vehicle infotainment slice, and the vehicle remote diagnosis and management slice.

One of the challenging tasks in the network slicing environment is the management of resources between slices, which still attracts the attention of researchers in the literature [6]. Moreover, fixed allocation is considered an appropriate solution to provide complete isolation in the slicing environment, as this solution does not allow resource sharing between slices. However, due to the high mobility of the vehicular environment, static allocation becomes inefficient to provide high performance in terms of resource utilization. In this context, many studies on resource sharing have been published in recent years, and few of them have focused on resource sharing in the vehicular environment. Furthermore, it is a real challenge to establish an efficient mechanism for resource sharing while maintaining a high level of isolation.

In this paper, we propose a novel V2X sharing mechanism that overcomes the limitations and weaknesses of static allocation. To this end, we propose a strategy based on deep Q-learning that aims to accept new calls and handovers that have been rejected by the admission control of overloaded slices, using the sharing mechanism.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3, the main part of the paper, is devoted to a detailed explanation of our solution. Section 4 is designed to evaluate the performance of the proposed solution. Section 5 concludes the paper and summarizes the strengths of our proposal.

2 Related work

In this section, we present various studies that have addressed the issue of resource management in the context of network slicing.

The work in [7] proposes an inter-slice resource management approach based on a real-encoded genetic algorithm. The goal of this approach is to maximize the number of users who obtain their required bitrate, while ensuring a high isolation between the slices.

In [8], the authors have proposed a strict slicing strategy that prohibits any sharing of resources between slices. This strategy allows complete isolation for each slice. However, all sharing requests received under this strategy will be rejected. This approach can result in inefficient resource utilization when some slices have excess capacity while others face shortages.

In [9], the authors presented a solution called SoftSlice to facilitate the sharing of radio resources. This solution consists of modelling user traffic within network slices as a Markov process. It allows overloaded slices to borrow unused radio resources from underloaded slices based on a predefined sharing agreement. This solution may not be suitable for scenarios where maintaining high isolation is a critical requirement.

The authors of [10] proposed two V2X resource borrowing schemes that use negotiation games to reallocate resources from an overloaded slice when a sharing request is received. The first scheme called NBS1 (Nash Bargaining solution for game I), is based on a bargaining game which is a special form of cooperative games. In this strategy, slices can engage in the bargaining game by offering resources to overburdened slices, provided that they adhere to certain conditions. It aims only to maximize the utility of the overloaded slice while respecting the QoS of the handover request. In this scheme, each lending slice offers a significant proportion of its resources to supply the entire amount requested for the handover call. The second scheme aims to maximize the sum of the utilities of the lending slices and the overloaded slice. This scheme seeks to establish a balance between the resources required for the handover call in the overloaded slice and the resources available in the lending slice. To solve the first problem, the authors proposed a low-complexity heuristic algorithm, while the second problem was addressed using the particle swarm optimisation (PSO) algorithm.

In [11], a resource scheduling algorithm based on dynamic scheduling is introduced for V2X communications. This algorithm mainly improves the user experience in three typical V2X services: basic safety services, advanced safety services and traffic efficiency services.

The article of [12] proposes a joint video quality selection and resource allocation technique aimed at maximizing the quality of experience (QoE). This is achieved by taking advantage of the queue dynamics and channel state of embedded devices, ensuring seamless video playback for these devices.

A mechanism for addressing communication in a V2X slicing environment is proposed in [13]. The objective of the proposed mechanism is to improve the packet reception ratio (PRR) in two logical slices: the autonomous driving slice (exchange of safety messages) and the infotainment slice (provision of a video stream).

The authors of [14] proposed a novel radio resource slicing framework to offload the tasks of Ultra-Reliable Low Latency Communication (URLLC) service

vehicles via a 5G slicing network. The optimization problem is formulated as a Markov decision process, with deep reinforcement learning used to manage the resource slicing while meeting the deadline requirements.

A reinforcement learning (RL) framework is proposed in [15] to dynamically allocate computing and radio resources for Internet of Vehicles (IoV) services that require different QoS requirements.

A dynamic resource allocation method is proposed in [16]. The proposed method aims to gradually share the unused radio resources from the underloaded slices to the overloaded slices until satisfying their needs or reaching the non-sharing threshold. This approach effectively balances resource utilization but may face challenges under highly dynamic network conditions.

3 QDRL-based resource sharing

The simplest solution for taking advantage of underloaded slices is to trigger the resource sharing mechanism with overloaded slices in order to maximize the use of resources in the system. Indeed, this solution improves the performance of the vehicular network. Nevertheless, it can potentially lead to a strong degradation of the isolation of the slicing environment, which means that the performance of the underloaded slices will be strongly influenced by the overloaded slices. To address this issue, we propose in this paper a flexible resource sharing mechanism, based on deep reinforcement learning, called DRL-based sharing resource. The purpose is to enable the resource sharing mechanism while maintaining high isolation between all slices. We present the correspondence between the key elements of reinforcement learning and the resource sharing problem. Furthermore, we detail and explain our solution based on deep Q-learning (DQL).

3.1 Background on reinforcement learning approach

Reinforcement learning (RL) is considered as a general class of algorithms in the field of machine learning, whose objective is to allow an agent to learn to behave in an environment, where the only feedback consists of a scalar reward signal. The agent's main purpose is to take actions that maximize long-term cumulative rewards, which is called the decision policy π . A Markov decision process (MDP) is commonly used to model the interactions between an agent and its environment. In general, a MDP model is formulated as follows: S represents the finite space of states, A denotes the finite space of actions, $P(s'|s, a)$ is the probability that action ($a \in A$) in state ($s \in S$) at slot " t " leads to state ($s' \in S$) at slot ($t + 1$), $R(s, a)$ is an immediate reward after performing action a in state s , and $\gamma \in [0, 1]$ is a discount factor to reflect the decreasing importance of the current reward on future rewards. Moreover, the objective of the MDP model is to find a policy $a = \pi(s)$ which selects an action " a " in state " s " in order to maximize the value function, which is defined as the discounted cumulative reward expected by the Bellman equation as expressed by equation 1 [17].

$$Q(s_t, q_t) = (1 - \alpha)Q_{old}(s_t, q_t) + \alpha(r_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a)) \quad (1)$$

where :

- $Q_{old}(s_t, a_t)$ is the old Q-value,
- $Q(s_t, a_t)$ is the new value obtained after updating the old value of Q,
- α is the learning rate,
- γ is the discount factor that indicates the importance of future rewards,
- r_t is the reward received as a result of the action a_t ,
- $\max_{a \in A} Q(s_{t+1}, a)$ is the estimate of the optimal future action.

The Q-learning algorithm is one of the free-model reinforcement learning algorithms that stores the learned results of each state and action pair (also known as the Q-value) in a Q-table [17]. However, one of the limitations of the Q-learning algorithm arises when the state space is large, requiring significant storage for the Q-table. As a result, the time required to explore each state to fill the Q-table becomes unrealistic. Function approximation presents a more attractive alternative, as it gets around this issue by approximating Q-values, making the task more manageable and scalable. In deep Q-learning, neural networks are used to approximate the Q-function in the form $Q(s, a; \theta)$, where θ represents a vector containing all the weights and biases of the neural network. This vector θ parameterises the Q-function, and is the parameter that undergoes updates during the learning process [18]. Therefore, the main part of the DQL solution is the training process that optimizes the whole network by minimizing the loss function. To this end, we intend to use the DRL method instead of using other approaches with high computational complexity.

3.2 Mapping with the key elements of the MDP model

The mapping to the key elements of the MDP model is presented as follows:

- *Station*: In our problem, a state is composed of six elements, noted s_t where :
 - $W_{autonomous}$, $W_{tele-operated}$, $W_{management}$ and $W_{infotainment}$ represent the sum of all the percentages of resources shared by each respective V2X slice.
 - "b" indicates the type of slice of the last accepted request, which may take the values 1, 2, 3 or 4, respectively the autonomous driving slice, the tele-operated driving slice, the remote vehicle diagnostics and management slice or the vehicle infotainment slice, of the last accepted request.
 - "m" indicates the type of request, where "1" corresponds to a handover request and "2" to a new call request.
- *Action*: Each action a_i is represented by a set of percentages: $a_i = [w_{i,1}\%, w_{i,2}\%, w_{i,3}\%, \dots, w_{i,K}\%]$, where $w_{i,k}$ is the percentage taken from the dedicated resources of the k^{th} slice, offered to accept a request from an overloaded slice. For example, $a_0 = [0\%, 0\%, 0\%, \dots, 0\%]$ represents an action which means rejecting the request.

- *Reward*: The determination of the reward is based on factors such as the type of sharing request, the type of overloaded slice and the isolation interference. If the type of action is a_0 , which is interpreted as a direct rejection action, a reward $R = 0$ is generated. The second part concerns the verification of the satisfaction of the quantity of resources requested by the sharing request, which checks whether the action has provided sufficient resources to accept sharing as presented in equation 2.

$$x_i = \sum_{k=1}^K \min(B_k * w_{i,k}, B_k - U_{k,t} - (B_k * W_k)) \quad (2)$$

where B_k is the number of resources dedicated to slice k , $U_{k,t}$ is the number of resources used by slice k at time t . However, if the resource offered is less than the resource requested, the action is interpreted as a rejection and a negative reward is generated. Otherwise, the sharing request is accepted and the environment moves to the next state. In this new state, we will examine whether each rejection in each slice that has shared part of its resources $w_{i,k} > 0$ by the last action is not a consequence of this shared part. Then, if we find that the action does not lead to rejection in the lending slices, a positive reward is generated, otherwise, a penalty is assigned. Therefore, we assume that each received sharing request has a slice priority denoted by α_k and a request type priority denoted by p_k , here $\alpha_k, p_k \in [0, 1]$. The reward is defined in equation 3.

$$R = \begin{cases} 0 & \text{if } a_x = a_0 \\ R_{max} * p_j * \alpha_k & \text{if } (M_k = x_i \wedge \text{No rejection is caused in the lender slices by } a_x) \\ C & \text{Otherwise} \end{cases} \quad (3)$$

R_{max} represents the maximum reward, and C indicates the penalty.

3.3 Reducing action space

In this subsection, if we denote the set of possible actions by A , where the size of A (i.e. the number of possible actions) is N^K , this means that each action corresponds to a combination of N possible percentages that a slice can offer for sharing, and that there are K slices in the network. The main purpose of reducing the action space is to eliminate infeasible sharing decisions. As a result, we can improve the chances of making the optimal sharing decision. The proposed method for reducing the action space consists of two parts:

1. Classification of prior actions

In this step, we classify the entire action space into a separate action space for each slice. We check whether an action can satisfy the amount of resources requested by a slice. If so, we add it to its action space. Moreover, in this

part we calculate the amount of resources offered by each action in the entire action space according to the equation 4.

$$X_{offered} = \sum_{k=1}^K B_k * w_{i,k} \quad (4)$$

In the second part, we evaluate the relevance of each action created to be added to the action space of a slice. To do this, we consider two types of slice:

- *First case*: when a slice reaches the maximum number of connections N_{max} that it can accept simultaneously and has no resources left, as expressed by the equation 5.

$$B_k - \sum_{n=1}^{N_{max}} b_k = 0 \quad (5)$$

where

$$N_{max} = INT\left(\frac{B_k}{b_k}\right)$$

b_k represents the data rate dedicated to every connection accepted in slice k .

In this scenario, when a slice k is overloaded, it no longer has any resources to offer itself. Therefore, we only include in the action space of slice k those actions which suggest that the overloaded slice offers itself a percentage of resources, where $w_{x,k} = 0\%$ as expressed in equation 6.

$$A_k = \{a_x \in A_k \mid w_{i,k} = 0 \wedge X_{offered} = X_{requested}\} \quad (6)$$

- *Second case*: when a slice reaches the maximum number of connections it can accept simultaneously, but still has a certain amount of resources, which are not sufficient to accept another connection as shown in equation 7.

$$B_k - \sum_{n=1}^{N_{max}} b_k > 0 \quad (7)$$

In this scenario, to use the remaining available resources in the overloaded slice and avoid wasting them, we not only include actions suggesting that the overloaded slice offers itself 0% of the resources, but we also add actions suggesting that it offers itself all the available resources. These available resources represent a percentage of all the resources dedicated to this slice, the number of which is calculated according to the equation 8.

$$Res_{available} = B_k - \sum_{n=1}^{N_{max}} b_k \quad (8)$$

We can then calculate the percentage of resources available using the equation 9.

$$Per_{available} = \frac{Res_{available} * 100}{B_k} \quad (9)$$

We then use the equation 10 to perform the classification.

$$A_k = \{a_x \in A_k \mid (w_{x,k} = 0 \vee w_{x,k} = Per_{available}) \wedge (X_{offered} = X_{requested})\} \quad (10)$$

2. Elimination of unfeasible actions

In this step, we aim to reduce the action space by removing sharing decisions that are not feasible, meaning that not all actions can be applied in each state. The proposed action elimination method consists of the following phases:

- (a) Let's consider $A_k = \{a_1, \dots, a_n\}$ as the action space of the k^{th} slice, which represents the overloaded slice for which we have received the sharing request.
- (b) We examine the feasibility of a sharing decision ($a_x \in A_k$) by checking whether $w_{x,k}$ exceeds the threshold of $(100 - W_k)$ as expressed in equation 11.

$$A_k = \{a_x \in A_k \mid w_{x,k} \leq (100 - W_k)\} \quad (11)$$

- (c) Then, if slice k is of type 2, is overloaded and requests sharing, and its $r_k > 0$ indicates that it has already shared the available resource, we eliminate all the actions in its action space which propose a $w_{x,k}$ greater than 0%. Otherwise, we eliminate the actions whose $w_{x,k} = 0$. Equation 12 illustrates the formal passage through this phase.

$$A_k = \begin{cases} \{a_x \in A_k \mid w_{x,k} = 0\} & \text{if } W_k > 0 \\ \{a_x \in A_k \mid w_{x,k} = Per_{available}\} & \text{Otherwise} \end{cases} \quad (12)$$

4 Performance evaluation

To evaluate the proposed resource sharing solution in a V2X environment, we use the OMNET++ 6.0 simulator with Simu5G [19] for 5G network communications and Sumo [20] for vehicle mobility on roads. We consider a map area of $3 \times 2.5 \text{ km}^2$, and cover it with 3 cells each with a transmission range of 1 km. The amount of dedicated resources for the 4 V2X slices in each cell is $B = 150 \text{ Mbps}$. We implement our DQL model using the Keras open source library (Python). Therefore, to facilitate the interaction between the DQL model and the simulation of vehicular communications, we used text files to implement a communication mechanism between the OMNeT++ (C++) simulator and the Keras (Python) library.

To illustrate the performance of our contribution, we compared it with other approaches described in the literature: strict slicing strategy [8], NBS1 [10], and SoftSLICE [9] in terms of resource utilization, handover dropping probability, and new Call blocking probability.

4.1 Simulation environment

We carried out a series of simulations to evaluate each performance criterion and calculated the average value of the results obtained. We varied the vehicle arrival rate (λ) to create different scenarios. We considered only the tele-operated driving slice and the autonomous driving slice that have the ability to request sharing in case of overload. It is essential to note that the vehicle can have several admitted flows simultaneously, each with its own distinct slices. Therefore, each cell will have its own deep Q-learning agent responsible for the sharing process. It is important to mention that we used an agent in the simulations, which underwent a training phase until it demonstrated relatively consistent rewards. The parameters used for these simulations are shown in Table 1.

Table 1. Simulation parameters

Parameters	Values		
	Slice capacity B_k	Required data rate	Priority α_k
Slice k			
Autonomous driving slice	0.25*B	10 Mbps	$\alpha_1 = 0.5$
Tele-operated driving slice	0.4*B	20 Mbps	$\alpha_2 = 1$
Vehicle remote diagnostics and management slice	0.1*B	1 Mbps	///
Vehicular infotainment slice	0.25*B	5 Mbps (web browsing)	///
R_{max}	10		
C	$-1 \times R_{max}$		
Percentage of values that the slice can offer via an action	0%, 5%, 10%, 15%, <i>cdots</i> , 40%		
p_h	1		
p_n	0.6		

4.2 Simulation results

In this section, we present the results obtained after a series of simulations and compare the results obtained by our proposal with those provided by strict slicing strategy [8], NBS1 [10], and SoftSLICE [9]. In this work, we define overloaded slices as those unable to accept additional new calls or handover requests using their dedicated resources, and have the permission to request sharing from other slices within the same cell, referred to as lender slices.

Resource utilization

Figure 1 shows that the SoftSlice solution has the highest resource utilisation, as it only evaluates the availability of resources within the same cell when processing a sharing request, without taking into account the importance of maintaining a high level of isolation between different slices. When the arrival rate increases, the method we propose comes close to the results of the SoftSlice scheme. This is because a higher arrival rate leads to a greater number of sharing requests in the network, allowing our solution to accept requests that do not compromise the performance of the lender slices. In addition, our solution outperforms the NBS1 solution because it allows the exchange of resources between slices for both the new calls and handover requests when slices are overloaded. In contrast, NBS1 only allows resource sharing for handover requests. On the other hand, the strict slicing scheme gives the lowest results in terms of resource utilisation due to its global strategy of isolating resources between slices, which prohibits the sharing of resources between slices.

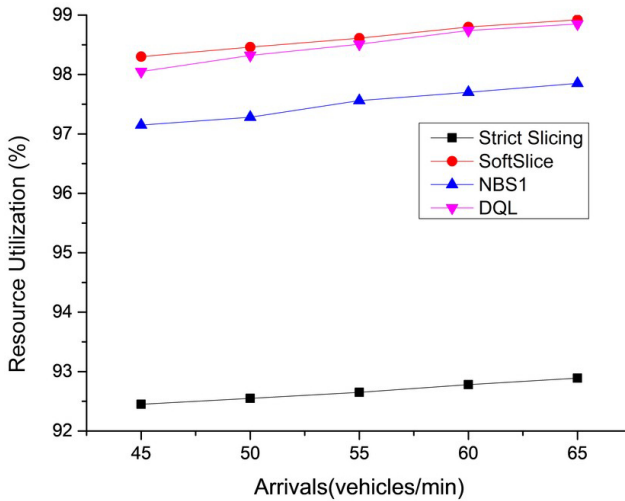


Fig. 1. Resource utilization vs. Vehicle arrival rate

Handover dropping probability

Figures 2 and 3 show that our solution outperforms SoftSlice in reducing the probability of dropping handover calls in overloaded slices due to the fact that SoftSlice does not prioritise handover requests. In addition, we note that the NBS1 solution achieves the lowest probability of handover dropping out in overloaded slices thanks to the sharing made from lending slices only for handover requests, which makes it possible to accept a greater number of these requests. However, when the probability of handover dropping is reduced in overloaded

slices, the performance of the lending slices degrades considerably in NBS1. In contrast, our approach minimizes the probability of handover dropping in overloaded slices while causing only a slight degradation in the performance of lending slices, thus maintaining a high level of isolation between slices. Furthermore, we can conclude that our solution achieves the lowest handover dropping probability in lending slices compared to other strategies that propose the sharing resources between slices. This is due to the deep learning agent, which has learned during the training process to avoid sharing actions that result in a higher probability of handover dropping out in lending slices.

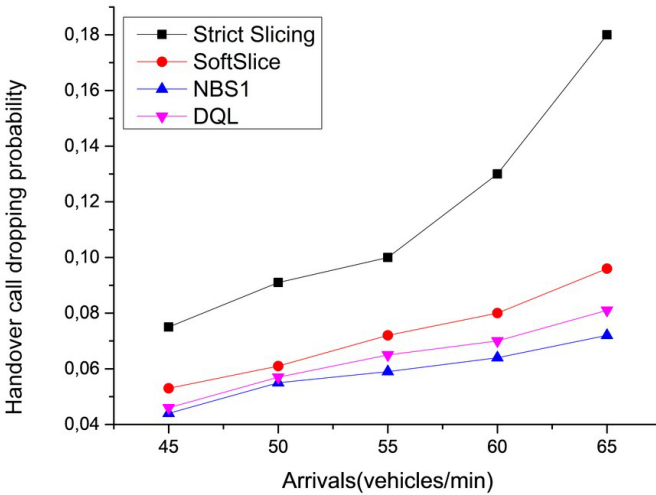


Fig. 2. Handover dropping probability in the overloaded slices

New Call blocking probability

Figures 4 and 5 illustrate that the SoftSlice scheme has the lowest probability of blocking new calls in overloaded slices compared to all other schemes. This is because our solution prioritizes handover requests over new calls. In addition, the NBS1 solution does not allow the sharing of resources for the new calls in overloaded slices. Furthermore, while our approach significantly reduces the probability of blocking new calls in overloaded slices, there is a slight deterioration in the probability of blocking new calls in lending slices. On the other hand, when the probability of blocking new calls in lender slices increases, the SoftSlice scheme has a substantial impact on the isolation between slices. For the strict slicing and NBS1 schemes, both exhibit the highest new call blocking probability in overloaded slices. This can be explained by the fact that these schemes deny any sharing of resources for the new calls of the overloaded slices. In addition, NBS1 still results in a significant degradation of the new call block-

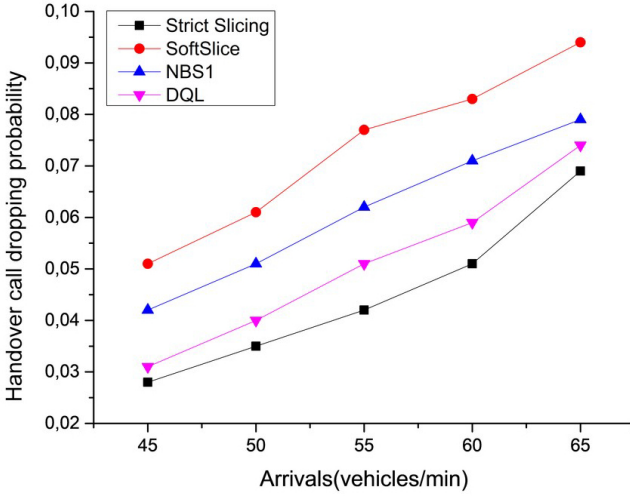


Fig. 3. Handover dropping probability in the lender slices

ing probability in lending slices due to the fact that resources are shared with handover requests from overloaded slices.

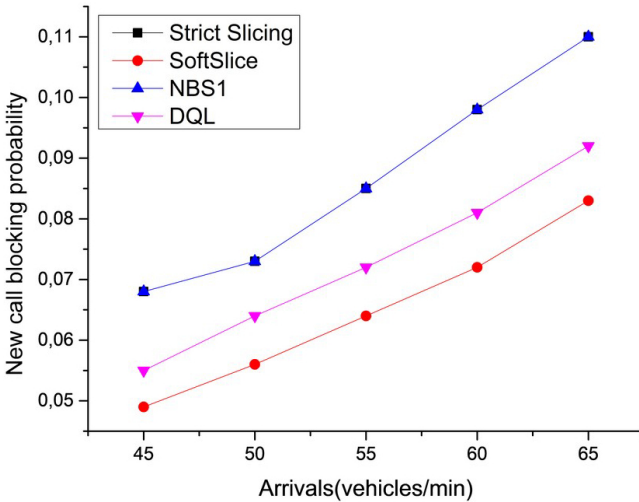


Fig. 4. New call blocking probability in overloaded slices

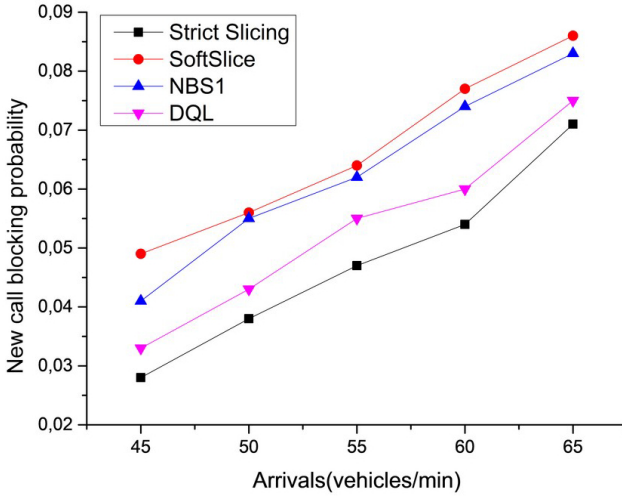


Fig. 5. New call blocking probability in lender slices

5 conclusion

This article focuses on the 5G V2X slicing environment. Specifically, to meet the challenge of managing bandwidth-constrained resources, an effective dynamic resource management policy is essential for 5G V2X slicing. This policy must take into account a number of factors, including efficiency, isolation and flexibility. Our deep reinforcement learning solution triggers the sharing process for overloaded slices, in order to reduce the probability of blocking new calls and the probability of dropping handover calls, while taking into account the maintenance of high isolation between slices. To do this, we modelled the problem as a Markov decision process. Next, we introduced a new resource sharing strategy using deep Q-learning. This approach is used to develop a policy for handling sharing requests from overloaded slices.

The obtained results showed that the proposed approach achieves a balance between increasing resource utilization and maintaining a high level of isolation, compared to the strict slicing, NBS1, and SoftSlice schemes. Future perspectives of this work include considering the sharing between slices of neighboring cells, to further enhance resource utilization.

References

1. Martin-Sacristàn, D., Michal, M., El-Ayoubi, S. E., Fallgren, M., Spapis, P. :5G PPP use cases and performance evaluation models. Technical report, 5G-Infrastructure Association, (2016)
2. Barakabitze, A. A., Ahmad, A., Mijumbi, R., Hines, A. :5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167, 106984 (2020). doi:10.1016/j.comnet.2019.106984

3. Campolo, C., Fontes, R. D. R., Molinaro, A., Rothenberg, C. E., Iera, A.: Slicing on the Road: Enabling the Automotive Vertical through 5G Network Softwarization. *Sensors*, 18(12), 4435 (2018). doi:10.3390/s18124435
4. Seremet, I., Causevič, S.: Benefits of using 5G Network Slicing to implement Vehicle-to-Everything (V2X) technology. In: 18th IEEE International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 176. IEEE, East Sarajevo, Bosnia and Herzegovina (2019). doi:10.1109/INFOTEH.2019.8717780
5. Campolo, C., Molinaro, A., Iera, A., Menichella, F.: 5G Network Slicing for Vehicle-to-Everything Services. *IEEE Wireless Communications*, 24(6), 38745 (2017). doi:10.1109/MWC.2017.1600408
6. Singh, P. K., Nandi, S. K., Nandi, S.: A tutorial survey on vehicular communication state of the art, and future research directions. *Vehicular Communications*, 18, 100164 (2019). doi:10.1016/j.vehcom.2019.100164
7. Yang, X., Liu, Y., Wong, I.C., Wang, Y., Cuthbert, L.: Genetic Algorithm for Inter-Slice Resource Management in 5G Network with Isolation. In: International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 176. IEEE, Split, Croatia (2020). doi:10.23919/SoftCOM50211.2020.9238298
8. Rodrigues, N. F., Barbosa, L. S.: Higher-Order Lazy Functional Slicing. *Journal of Universal Computer Science*, 13(6), 8547873 (2007).
9. Gebremariam, A. A., Chowdhury, M., Usman, M., Goldsmith, A., Granelli, F.: Soft-SLICE: Policy-Based Dynamic Spectrum Slicing in 5G Cellular Networks. In: International Conference on Communications (ICC), pp. 176. IEEE, Kansas City, USA (2018). doi:10.1109/ICC.2018.8422148
10. Mouawad, N., Naja, R., Tohme, S.: Inter-slice handover management in a V2X slicing environment using bargaining games. *Wireless Networks*, 26, 388373903 (2020). doi:10.1007/s11276-020-02292-5
11. Cai, Y., Zhang, Q., Feng, Z.: QoS-Guaranteed Radio Resource Scheduling in 5G V2X Heterogeneous Systems. In: IEEE Globecom Workshops (GC Wkshps), pp. 176. IEEE, Waikoloa, USA (2019). doi:10.1109/GCWkshps45667.2019.9024700
12. Khan, H., Samarakoon, S., Bennis, M.: Enhancing Video Streaming in Vehicular Networks via Resource Slicing. *IEEE Transactions on Vehicular Technology*, 69(4), 351373522 (2020). doi:10.1109/TVT.2020.2975068
13. Khan, H., Luoto, P., Bennis, M., Latva-aho, M.: On the Application of Network Slicing for 5G-V2X. In: 24th European Wireless Conference, pp. 176. IEEE, Catania, Italy (2018).
14. Hao, M., Ye, D., Wang, S., Tan, B., Yu, R.: Urrlc resource slicing and scheduling in 5G vehicular edge computing. In: 93rd Vehicular Technology Conference (VTC2021-Spring), pp. 175. IEEE, Helsinki, Finland (2021). doi:10.1109/VTC2021-Spring51267.2021.9448805
15. Wu, W., Chen, N., Zhou, C., Li, M., Shen, X., Zhuang, W., Li, X.: Dynamic RAN Slicing for Service-Oriented Vehicular Networks via Constrained Learning. *IEEE Journal on Selected Areas in Communications*, 39(7), 207672089 (2021). doi:10.1109/JSAC.2020.3041405
16. Maule, M., Mekikis, P. V., Ramantas, K., Vardakas, J., Verikoukis, C.: Dynamic partitioning of radio resources based on 5g ran slicing. In: Proceedings of Global Communications Conference, pp. 176. IEEE, Taipei, Taiwan (2020). doi:10.1109/GLOBECOM42002.2020.9322385
17. Van Otterlo, M., Wiering, M.: Reinforcement Learning and Markov Decision Processes. In: Wiering, M., Van Otterlo, M. (eds.), pp. 3-42. Berlin, Heidelberg: Springer Berlin Heidelberg (2012). doi:10.1007/978-3-642-27645-3_1

18. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Legg, D.W.S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature*, 518, 529?533 (2015). doi:10.1038/nature14236
19. Nardini, G., Sabella, D., Stea, G., Thakkar, P., Virdis, A.: Simu5G-An OMNeT++ Library for End-to-End Performance Evaluation of 5G Networks. *IEEE Access*, 8, 181176?181191 (2020). doi:10.1109/ACCESS.2020.3028550
20. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: Sumo-simulation of urban mobility: An overview. In: 3rd International Conference on Advances in System Simulation (SIMUL 2011), pp. 63?68. ARIA, Barcelona, Spain (2011).

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

