Received <day> <Month>, <year>; | Revised <day> <Month>, <year>; | Accepted <day> <Month>, <year>

DOI: xxx/xxxx

LETTER

Deep Reinforcement Learning-Aided RAN Slicing Enforcement Supporting Latency Sensitive Services in B5G networks

Sergio Martiradonna*[1,4] | Andrea Abrardo[2,4] | Marco Moretti[3,4] | Giuseppe Piro[1,4] | Gennaro Boggia[1,4]

[1]Dept. of Electrical and Information Engineering, Politecnico di Bari, Bari, Italy
[2]Dept. of Information Engineering and Applied Mathematics, Università degli Studi di Siena, Siena, Italy
[3]Dept. of Information Engineering, Università degli Studi di Pisa, Pisa, Italy
[4]CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni

Correspondence
*Sergio Martiradonna, Dept. of Electrical and Information Engineering, Politecnico di Bari, Bari, 70125, Italy. Email: sergio.martiradonna@poliba.it

Abstract

Edge computing and artificial intelligence promise to turn future mobile networks into service- and radio-aware infrastructures, able to address the requirements of upcoming latency-sensitive applications. For instance, they can be used to dynamically and optimally manage the Radio Access Network Slicing. However, this is a challenging goal, due to the mostly unpredictably nature of the wireless channel. This paper presents a novel architecture using Deep Reinforcement Learning at the network edge for addressing Radio Access Network Slicing and Radio Resource Management. By considering the autonomous-driving use-case, computer simulations demonstrate the effectiveness of our proposal against baseline methodologies.

KEYWORDS:
Network Slicing; Radio Access Network; Deep Reinforcement Learning; Edge Intelligence; B5G

1 | INTRODUCTION

Network slicing emerged as an effective design paradigm for current and future mobile architectures. It allows the creation of core network segments, dedicated to the provisioning of specific services with their own Service Level Agreement (SLA) and Quality of Service (QoS) requirements [1]. While the infrastructure provider (IP) still represents the owner of the resources employed for each slice, the slice tenant (TNT), that is the customer from vertical industries, can use those resources, install its applications, hold its data, and enable its preferred security policies. Thus, the slice appears as a virtualized and independent portion of the overall network, configurable through a service-based approach [1,2]. The idea to support orthogonal logical segments also at the radio interface of Beyond 5G (B5G) deployments recently gained momentum. Unlike the conventional network slicing concept, Radio Access Network (RAN) slicing is less mature and more challenging [3], because of the intrinsically shared and unpredictable nature of wireless resources and the need of novel Radio Resource Management (RRM) functionalities [4]. Edge Intelligence (EI) is considered as the most powerful enabling technology for RAN slicing enforcement. By leveraging the native capabilities of both Edge Computing and Artificial Intelligence, it promises to simplify the large-scale data acquisition, predict the incoming agglomerated per-slice traffic, and efficiently support resource allocation, management, orchestration, and network automation [5].

At the time of this writing, several AI-based solutions to anticipate future offered loads in mobile networks have been extensively studied [6,7,8]. However, estimating the traffic only represents a partial step for the optimal slice resource allocation problem. Even in the presence of perfect traffic estimation, in fact, evaluating the optimal RRM is a very difficult task owing to the random nature of the radio conditions and, as a matter of fact, the problem of optimal RRM generally requires unmanageable computational complexity. To make matters worse, the inherent requirements of latency sensitive services, put further constraints on this problem and call for unconventional, distributed, and scalable slicing approaches [9]. For these reasons, Reinforcement Learning (RL) has been recently investigated as a low-complexity and effective solution for RRM in communication and computing systems [10,11]. Here, an RL agent can generate (near-) optimal control actions on the basis of the reward feedback from interactions

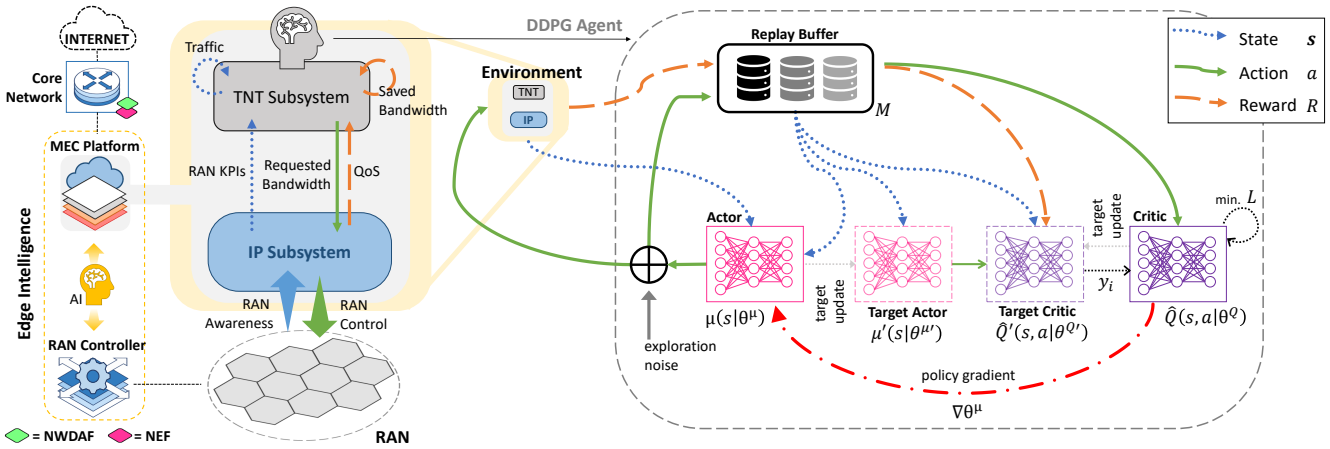

FIGURE 1 Reference architecture with block diagram of the DDPG algorithm.

with the environment. Instead of simply optimizing the current reward in a greedy manner, the RL agent can take a long-term goal into account. Thus, RL appears to be particularly suited for RRM problems when the optimum is very difficult to know, only a reward associated with a given policy is available, and the reward/loss cannot be related, through a closed-form expression, to the actions [12].

In line with these premises, we present in Section 2 a novel architecture for RAN slicing that exploits Deep RL (DRL)-based EI to serve latency sensitive applications. It considers a virtualized control platform in which both the IP and TNTs interact for enforcing Network Slices in the RAN. Note that unlike other state-of-the-art studies [6,7,10,11,13,14], the conceived methodology considers the openness of the network to third parties, hence encouraging TNTs to directly take control of the resources and drive the slice enforcement without relying on an IP-centralized solution. Besides, this architecture ensures that the roles of the different involved stakeholders are maintained: the IP is not aware of TNTs' most valuable information; the latters, in turn, will have an only partial understanding of the underlying RAN information [9]. Section 3 considers a use-case based on autonomous-driving to show the effectiveness of our proposal against baseline methodologies. Remaining open issues are finally drawn in Section 4 to sketch future research directions.

2 | THE PROPOSED ARCHITECTURE FOR RAN SLICING WITH EI

We consider herein a scenario envisaging a single IP that leases part of its network resources to create and manage specific slices for a set of independent TNTs, or mobile network operators, to realize advanced network services [15]. The IP determines the amount of resources that can be used by the TNTs for each slice. The TNTs, in turn, should adapt in real-time their requests according to their own users' requirements, avoiding expenses due to the issue of resources overbuying. As a consequence, the generation of *slice requests*, i.e., when a TNT defines its needed slice configuration to the IP, and the *slice dynamic enforcement*, i.e., the adaptation of the slice allocation policy to the time-varying RAN environment, are the solutions of local optimization problems, which have to be solved in real-time and whose decisions have to be executed instantaneously to reduce any latency of the system [16]. The reference scenario, shown in Fig. 1, consists of a controller that dynamically performs slicing operations at the RAN layer for a cluster of cells, i.e., 1) decides which slice creation requests can be admitted, 2) computes a slicing policy to allocate the available resources to the admitted slices, and 3) enforces the slicing policy on the underlying physical RAN. The network slices are instantiated by the interactions of two different entities, the *IP subsystem* and the *TNT subsystem*, both virtualized on the Multi-access Edge Computing (MEC) platform, which is co-located to the same cluster of cells [9].

Specifically, the IP subsystem is in charge of creating different RAN slices leveraging on the RAN controller and according to the information coming from both the TNT subsystems and the NetWork Data Analytics Function (NWDAF). In essence, the IP subsystem is the RAN counterpart of the Network Exposure Function (NEF) in the Core Network. On the other hand, the TNT subsystem essentially generates the slice requests by processing general information (e.g., type of services to be provided, the duration in time of the slice), as well as high-level control information for successfully addressing the requirements of the

related slice. Hence, the IP must decide in advance the number of resources that will remain assigned to a slice until the next reallocation takes place.

As for the specific RAN slicing enforcement strategy, we propose a dynamical RAN slicing where each slice is assigned a given radio resource pool across a cluster of interfering cells in a given service area[4]. The number of RBs is dynamically determined and requested by the TNT on the base of the pieces of information it has access to.

This scenario can be described as a discrete-time stochastic control process modelling a classical Markov decision process (MDP), where the cellular system is the *environment*, whose *state* $\mathcal{S}$ is represented by the radio conditions of the nodes and the amount of incoming traffic, the *reward* R is the efficiency of resource utilization subject to QoS constraints and the *set of actions* $\mathcal{A}$ is the bandwidth allocated to mission critical slices. However, the optimal RRM solution cannot be known because of the non-convex nature of the problem and for the fact that the TNTs have only partial knowledge of the underlying RAN information. Indeed, it is worth noting that the details of the radio interface, e.g., the adopted numerology, the scheduling policy, the packet fragmentation rules, and so on, are fully in charge of the IP and are not known by the TNT agents, which have only a limited knowledge of the radio link conditions of their users. Besides, the reward that is of interest for the TNT is often a QoS parameter, e.g., the latency and the packet loss ratio for mission-critical users, whose relationship with the allocation decision, e.g., the amount of allocated spectrum, is very hard to establish. Moreover, $P^a_{s,s'}$, which is the transition probability from the state s to the state s' given the action a and $R(s,a)$, i.e., the average reward R in the state s given the action a, are unknown since they depend on the cellular environment, whose dynamic is not predictable. In this context, RL emerges as the perfect tool to address the RAN slicing problem. Nevertheless, the dimensions of states and actions are huge or possibly infinite, therefore Q-learning approaches are uneffective. As a consequence, one of the most effective way to deal with the problem is through model free RL and, in particular, with function approximation of the action value function $Q(s,a)$ given by neural networks[12].

As illustrated in Fig. 1, we trained the TNT agent with the Deep Deterministic Policy Gradient (DDPG) algorithm, that is an off-policy model-free algorithm dealing with continuous states and actions[17]. Thanks to an actor-critic method, a DDPG agent concurrently learns a Q-function and an optimal policy that maximizes the long-term reward. The idea is to evaluate the Q function through an approximation $\hat{Q}(s,a|\theta^Q)$ and to represent the policy through another approximation $\mu(s|\theta^\mu)$. In particular, θ^μ and θ^Q are the parameters of the actor and critic neural networks, respectively. In addition, two copies of the actor and critic, that is the target networks, are used to improve the stability of learning the action-value function, since target values are constrained to change slowly. The target critic is identified by $\hat{Q}'(s,a)$ and $\theta^{Q'}$, while $\mu'(s)$ and $\theta^{\mu'}$ are related to the target actor.

The update of the actor and critic networks occurs with the gradient descent method. In particular, the critic parameter θ^Q is updated by minimizing the loss $L = \frac{1}{M}\sum_{i=1}^{M}\left(y_i - \hat{Q}(s_i,a_i|\theta^Q)\right)^2$, where M is the number of experiences sampled from the replay buffer, $y_i = R_i + \gamma\hat{Q}'(s_i,\mu'(s'_i|\theta^\mu)|\theta^{Q'})$, and γ is the future reward discount factor.

Let J be the environment start distribution. Indeed, the actor policy θ^μ is updated by following the sampled policy gradient to maximize the expected discounted reward: $\nabla_{\theta^\mu}J \approx \frac{1}{M}\sum_{i=1}^{M}\nabla_{\mu(s_i)}\hat{Q}\left(s_i,\mu(s_i)|\theta^Q\right)\nabla_{\theta^\mu}\mu(s_i|\theta^\mu)$.

The *action* $a \in \mathcal{A}$ is the amount of bandwidth requested every allocation period to the IP. The *state* $\mathbf{s} \in \mathcal{S}$ is a vector of some Key Performance Indicators related to the RAN and traffic information. The state can be either directly computed by the TNT (e.g., the agglomerated slice traffic) or communicated by the IP and is used to determine the amount of bandwidth requested for the next period. The *reward* $R(s,a)$ takes into account the amount of bandwidth the TNT saves with respect to the maximum bandwidth as well as some other QoS indicators. The TNT action is dynamically chosen on the base of the available observations (state) with the goal of maximizing a discounted average future reward. We consider $a \in [0.1, 0.9]$, i.e., the *action* is a continuous value between 10% and 90% of the maximum bandwidth allocated to the TNT. The *state* is defined as $\mathbf{s} = (l, r, d, o)$, where l is the total per-slice agglomerated traffic to be sent, r is the average rate of the user experiencing the worst channel conditions (averaged over the allocation period), d is the maximum delay experienced by the users of the slice, and o represents the number of QoS outages happened in the episode. If the QoS requirement is satisfied, the *reward* is set to $R = 1 - a$. Otherwise: $R = -1$. The less the bandwidth requested by the TNT while satisfying the target QoS requirement, the higher the reward.

3 | PERFORMANCE EVALUATION

To evaluate the performance of the proposed model in a realistic environment, we consider a specific use-case based on autonomous-driving. Since we focus on a latency-sensitive scenario, we assume that the TNT slice requests must be always accepted by the IP, i.e., neither an admission control nor a resource allocation negotiation policy is enforced (for instance, see[18] for further details on DRL-based admission control.) The service level agreement between the latency-sensitive TNT and the IP

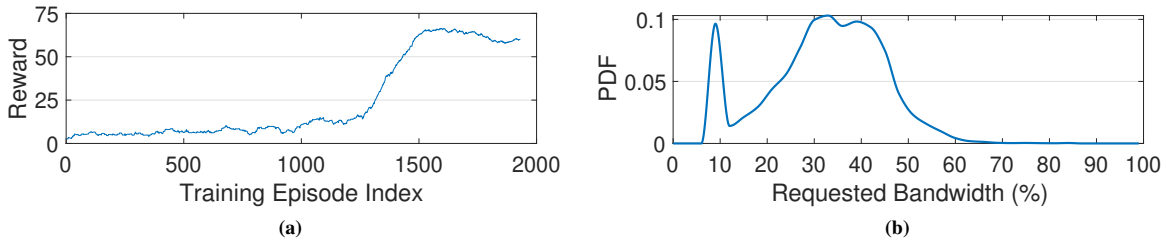

FIGURE 2 (a) Reward during the training phase of the agent. (b) Probability Density Function of the actions taken by the agent.

sets a maximum amount of bandwidth to be used in each cell and, by providing for a unitary cost associated with each bandwidth resource, enforces *pay for what you get* mechanisms to prevent from over-provisioning the TNT. Note that, if the TNT does not require the whole bandwidth, the unexploited bandwidth may be safely reused by other TNTs. Indeed, the TNT subsystem should dynamically determines and requests only the resources it needs.

Without loss of generality, we focus on a single cell scenario, i.e., we do not consider the effect of inter-cell interference. As a matter of fact, the proposed framework leverages on the capability of the DRL agent to predict the mutual interactions of the involved nodes in determining the actual system performance. As a consequence, it is naturally suitable to encompass a multi-cell scenario, provided that the state variables include some interference related parameters, e.g., the mutual position of the nodes. Channel modeling considers the 3GPP UMa scenario [19], whose path loss and lognormal shadowing are implemented. The data rate of each active link is then derived based on the Shannon capacity formula. In the following, we focus on the downlink case. Similar considerations and results can be obtained for the uplink case.

Our scenario contains one single macro Base Station and a single TNT subsystem, which is assumed to provide autonomous driving services. In the considered setting, vehicles use their own sensors (e.g., HD camera, LiDAR), as well as sensor information from other vehicles, to perceive the environment and obtain a 3D model of the world around them. The main QoS requirement of the slice is a maximum experienced packet delay of 5 ms, which is half the maximum value of latency envisioned for the High Definition Sensor Sharing, which is one of the main Autonomous Driving use cases [20]. The packet length is assumed to be fixed and equal to 32 bytes (as per the ITU-IMT2020 Urban Macro–URLLC usage scenario). The number of slice subscribers, i.e. the autonomous vehicles, is modeled according to real mobility traces: CRAWDAD dataset epfl/mobility (v. 2009-02-24), https://doi.org/10.15783/C7J010. The TNT is allocated a maximum bandwidth of 10 MHz, organized into slots of 1 ms, according to the 5G NR numerology with $\Delta f = 15$ KHz. The MAC scheduling strategy enforced by the IP is the Throughput to Average scheduling, in order to guarantee a minimum level of service to every user, hence reaching a high fairness index. The DDPG agent performs its actions every allocation period of 1 s. As for the actor and critic neural networks, they are composed of two fully connected layers, each with 2000 and 1500 neurons, respectively. The learning rate is set equal to 0.001 for the actor and 0.0001 for the critic.

Fig. 2 (a) shows the running average (with a window length of 100 episodes) of the reward during the training process of the agent. The figure shows that the proposed DRL approach allows to converge to a bandwidth occupancy of around 35% (65% of bandwidth left to other usages). During an initial exploration phase, the agent is not able to address the QoS requirement, hence the low reward. After approximately 1200 training episodes, rewards begin to grow, since the algorithm successfully learned how to satisfy the latency constraint. Fig. 2 (b) shows the probability density function of the bandwidth requested by the DRL agent of the TNT. Samples are related to 10000 independent simulations. On the one hand, it demonstrates how the agent effectively learned to perform a variety of actions, i.e., it learned to dynamically adapt to the environment. On the other hand, it shows that the agent usually tries to request as low bandwidth as possible, hence indicating a well-engineered reward function. Please note that this figure and the following are obtained by running the agent obtained at the end of the learning phase over the dataset considered for the testing phase. In this way, it is possible to assess the capability of the proposed DRL approach to generalize the proposed control strategy to every possible data traffic and radio channel conditions.

To better demonstrate the importance of DRL, we compare the simulation results with the following methods: Fixed Allocation, in which the TNT requests always the same amount of bandwidth; *Heuristic strategy*, characterized by a perfect prediction (i.e., ideal) of the incoming traffic and a bandwidth request that is directly proportional to the incoming traffic at each step; *Optimum allocation*, in which at each step the minimum bandwidth allowing to fulfill the slice QoS requirements is determined through iterative adjustment. Clearly, this last approach is unfeasible in a real system, although it can be easily simulated. Fig.

3 (a) shows the bandwidth requested by the TNT during a representative test episode. It clearly illustrates how the agent learned to request an amount of bandwidth close to the optimum, by taking into account only the state variables. In other words, the agent is dynamically adapting to the changes occurring in the environment. Furthermore, it is of the utmost importance to highlight how the proposed DRL solution outperforms the heuristic approach. In other words, even though the prediction of the incoming traffic is accurate, it is not sufficient to guarantee an optimal bandwidth request. As a matter of fact, it is necessary to take into account what actually happens in the RAN to accomplish such a decision. For instance, it is clear that the incoming traffic grows substantially after 60 s. However, it is reasonable to assume that general radio channel conditions improve as well, therefore it is not strictly necessary to claim more bandwidth. Fig. 3 (b) shows the bandwidth requested by the TNT to ensure a certain level of QoS availability, i.e., the probability associated with the main QoS requirement being satisfied. Specifically, the actions taken by both the trained DRL agent and the heuristic are successively weighted to obtained different behaviors. The results are then averaged over 10000 independent simulations. The most noticeable feature is that the proposed DRL mechanism always outperforms the other strategies. Even though requested bandwidth always grows with more stringent requirements on the QoS Availability probability, the DRL agent requests up to 50% less bandwidth compared to the fixed allocation. Moreover, the variation of the bandwidths requested by the TNT DRL agent are incredibly smaller, confirming how the agent learned a near-optimal allocation strategy starting from the limited information available.

4 | CONCLUSIONS AND OPEN CHALLENGES

We presented a novel architecture in which both the Infrastructure Provider and tenants interact for enforcing Network Slices in the next-generation RAN. It exploits Deep Reinforcement Learning at the edge for supporting effective enforcing of RAN slicing, where tenants are encouraged to take control having an only partial understanding of the underlying RAN status. Focusing on the autonomous-driving use case, our proposal's effectiveness against baseline methodologies is investigated through computer simulation. Results confirm that the prediction of the incoming agglomerated per-slice traffic is not sufficient for an effective RAN slice enforcement strategy and resource over-provisioning is remarkably inefficient. Even though the proposed solution proves its success, there are still different open challenges to deal with in future works. First, the tradeoff between the definition of the DRL state and the overhead in the communication between the Infrastructure provider and Tenants subsystems appears crucial to improve performance further. Second, it is important to use cutting-edge methodologies (e.g., transfer learning) for developing flexible and interoperable software agents, in order to guarantee reconfigurability and continuous deployment. Moreover, how to provide sufficient and powerful resources for running AI at the edge in an economically sustainable way, is another important aspect to address for properly preparing for the advent of EI.

ACKNOWLEDGMENT

This work was supported by the Italian MIUR PRIN project no. 2017NS9FEY entitled "Realtime Control of 5G Wireless Networks: Taming the Complexity of Future Transmission and Computation Challenges".

References

1. Foukas X, Patounas G, Elmokashfi A, Marina MK. Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine* 2017; 55(5): 94–100.

2. Zhou X, Li R, Chen T, Zhang H. Network slicing as a service: enabling enterprises' own software-defined cellular networks. *IEEE Communications Magazine* 2016; 54(7): 146–153.

3. Elayoubi SE, Jemaa SB, Altman Z, Galindo-Serrano A. 5G RAN slicing for verticals: Enablers and challenges. *IEEE Communications Magazine* 2019; 57(1): 28–34.

4. Sallent O, Perez-Romero J, Ferrus R, Agusti R. On radio access network slicing from a radio resource management perspective. *IEEE Wireless Communications* 2017; 24(5): 166–174.

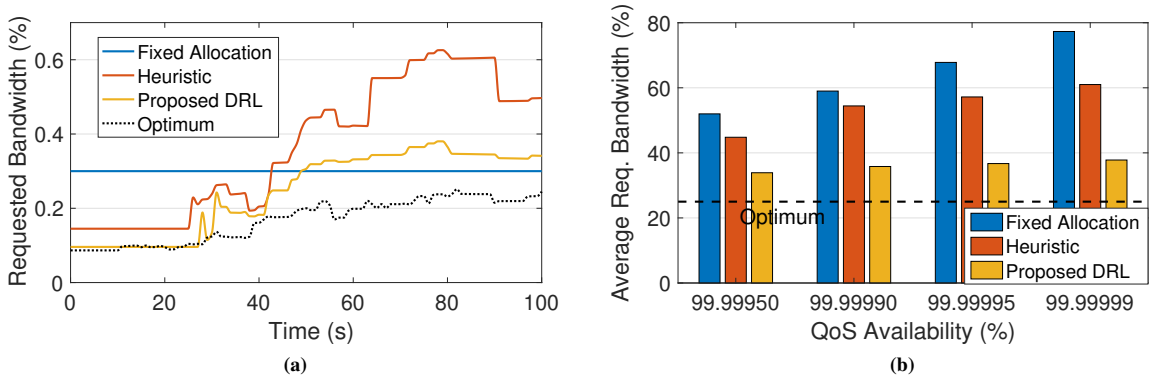

FIGURE 3 (a) Bandwidth requests in a representative test episode. (b) Average bandwidth satisfying a given QoS availability.

5. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE* 2019; 107(8): 1738–1762.

6. Nikravesh A, Ajila S, Lung C, Ding W. An experimental investigation of mobile network traffic prediction accuracy. *Services Transactions on Big Data* 2016; 3(1): 1–16.

7. Bega D, Gramaglia M, Fiore M, Banchs A, Costa-Perez X. DeepCog: Optimizing resource provisioning in network slicing with AI-based capacity forecasting. *IEEE Journal on Selected Areas in Communications* 2019; 38(2): 361–376.

8. Gutterman C, Grinshpun E, Sharma S, Zussman G. RAN resource usage prediction for a 5G slice broker. In: ACM. ; 2019: 231–240.

9. Martiradonna S, Abrardo A, Moretti M, Piro G, Boggia G. Architecting RAN slicing for URLLC: Design decisions and open issues. In: IEEE. ; 2019: 1–4.

10. Ye H, Li GY, Juang BHF. Deep reinforcement learning based resource allocation for V2V communications. *IEEE Transactions on Vehicular Technology* 2019; 68(4): 3163–3173.

11. Mismar FB, Evans BL, Alkhateeb A. Deep reinforcement learning for 5G networks: Joint beamforming, power control, and interference coordination. *IEEE Transactions on Communications* 2019; 68(3): 1581–1592.

12. Sutton RS, Barto AG. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press . 2011.

13. Chen X, Zhao Z, Wu C, et al. Multi-tenant cross-slice resource orchestration: A deep reinforcement learning approach. *IEEE Journal on Selected Areas in Communications* 2019; 37(10): 2377–2392.

14. Abiko Y, Saito T, Ikeda D, Ohta K, Mizuno T, Mineno H. Flexible resource block allocation to multiple slices for radio access network slicing using deep reinforcement learning. *IEEE Access* 2020; 8: 68183–68198.

15. Akgül ÖU, Malanchini I, Capone A. Dynamic resource trading in sliced mobile networks. *IEEE Transactions on Network and Service Management* 2019; 16(1): 220–233.

16. D'Oro S, Restuccia F, Melodia T. Toward operator-to-waveform 5G radio access network slicing. *IEEE Communications Magazine* 2020; 58(4): 18–23.

17. Luong NC, Hoang DT, Gong S, et al. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials* 2019; 21(4): 3133–3174.

18. Raza MR, Natalino C, Öhlen P, Wosinska L, Monti P. Reinforcement Learning for Slicing in a 5G Flexible RAN. *Journal of Lightwave Technology* 2019; 37(20): 5161-5169. doi: 10.1109/JLT.2019.2924345

19. 3GPP . Study on channel model for frequencies from 0.5 to 100 GHz. TR 38.901; 2017.

20. 5GAA . C-V2X Use Cases: Methodology, Examples and Service Level Requirements. White Paper; 2019.