A Novel Approach for Dynamic Capacity Sharing in Multi-tenant Scenarios

I.Vilà, J. Pérez-Romero, O.Sallent, A.Umbert
Signal Theory and Comunications Department of Universitat Politècnica de Catalunya (UPC)
Barcelona, Spain
irene.vila.munoz@upc.edu, [jorperez, sallent, annau]@tsc.upc.edu

***Abstract*— Network slicing is included as a key feature of the 5G architecture in order to simultaneously support diverse service types with heterogeneous requirements. The deployment of network slicing in the Radio Access Network (RAN) needs mechanisms that allow the distribution of the available capacity in the system in an efficient manner while satisfying the requirements of the different services. In this paper, a capacity sharing function is proposed, which is approached as a multi-agent reinforcement learning based on the Deep Reinforcement Learning (DRL) algorithm Deep Q-Network (DQN). The proposed algorithm provides the capacity to be assigned to each RAN slice. Performance assessment reveals the promising behaviour of the proposed solution.**

Keywords—RAN Slicing; Capacity sharing; Reinforcement Learning, Deep Q-Network.

I. INTRODUCTION

Network slicing provides 5G networks with the capability to simultaneously support the operation of multiple tenants (e.g. communication providers or mobile virtual network operators (MVNO)), which offer services with heterogeneous requirements [1][2]. This is achieved by provisioning each tenant with an end-to-end logical network, denoted as network slice, which is optimised to the requirements of the tenant. Focusing on the Radio Access Network (RAN) part of a network slice, denoted as RAN slice, efficient mechanisms to manage the common pool of resources available in each of the nodes (e.g. gNB) of the Next Generation (NG)-RAN infrastructure need to be developed in order to support multiple and diverse RAN behaviours at the same time [3]. These mechanisms include capacity sharing strategies to dynamically allocate the appropriate capacity to the existing RAN slices depending on their needs and targeting an efficient use of the available radio resources.

This paper tackles the problem of capacity sharing in RAN slicing scenarios. Some works have approached the capacity sharing problem as an optimisation problem, such as the defined by Karush Kuhn Tucker conditions in [4] or by integer programming formulation in [5]. Moreover, a biconvex optimisation problem has been defined to find a joint sharing solution of radio resources, caching and backhaul in [6]. Other works approach the capacity sharing problem heuristically by making use of an exponential smoothing model in [7], a market-oriented model such as in [8] and [9], a winner bid problem in [10], a fisher market game in [11] or as an iterative algorithm in [12]. In order to deal with the uncertainty inherent in wireless environments and the large-scale of 5G mobile networks [13], some other works have approached the capacity sharing problem in RAN slicing scenarios by using Deep Reinforcement Learning (DRL) algorithms [14]. In this regard, in [15] and [16], the capacity reserved to each slice is provided by means of Deep Q-Network (DQN) and Deterministic Policy Gradients combined with K-Nearest Neighbours, respectively. In turn, [17] and [18] provide the cell capacity share solution by firstly computing the aggregated capacity reserved to each slice at the network level by using DQN and, then, applying an heuristic algorithm to obtain the cell capacity for each tenant. However, none of the previous works has focused on Multi-Agent Reinforcement Learning (MARL) approaches to obtain the capacity sharing solution in RAN slicing scenarios.

978-1-7281-4490-0/20/$31.00 ©2020 IEEE

In this paper, the capacity sharing problem is approached as a collaborative MARL algorithm based on DQN that learns the capacity to be provided to each tenant by associating each agent in the MARL to a different tenant. In this way, tenants can be easily added/removed in the scenario just by adding or removing the corresponding agent. Additionally, the model explores the behaviour of the capacity sharing function when including Service Level Agreement (SLA) definition at the system level and at the cell level instead of considering SLAs at the user level, such as in [15]-[18].

The rest of the paper is organised as follows. Section II formulates the problem of capacity sharing in a multi-cell scenario and Section III describes the proposed MARL DQN approach. Based on this, Section IV includes the results obtained when considering a multi-tenant scenario with diverse traffic load situations. Finally, Section V summarises the conclusions.

II. PROBLEM DEFINITION

Let us consider an Infrastructure Provider (InP) that owns a NG-RAN infrastructure, which is composed of N cells with diverse deployment characteristics (i.e., cell radius, transmission power, frequency of operation). Considering that 5G New Radio (NR) technology is used, each cell n has a total of $N_T(n)$ Physical Resource Blocks (PRBs), which provide a total cell capacity $C_T(n)$ (b/s). The NG-RAN infrastructure is shared among K

tenants and each of them is provided with a RAN Slice Instance (RSI).

In order to satisfy the service requirements, a Service Level Agreement (SLA) is established between the InP and each of the tenants. Specifically, the SLA for the k-th tenant is defined in terms of:

- Scenario Aggregated Guaranteed Bit Rate ($SAGBR_k$): the aggregated bit rate to be provided across all cells to tenant k if requested.
- Maximum Cell Bit Rate ($MCBR_{k,n}$): the maximum bit rate that can be provided to tenant k in cell n. This parameter allows controlling the capacity provided to a tenant among the different cells.

The SLA is established when creating the RSI and its fulfilment is one of the responsibilities of the RAN Slicing Management Function (RSMF). The RSMF produces management services for the correct operation and dynamic configuration of the RSIs and is in charge of the Lifecycle Management (LCM) of the RSIs, including their creation, modification, optimisation and termination. Note that the RSMF corresponds to the Network Slice Subnet Management Function (NSSMF) of the 3GPP management model in [19] for the case of managing RAN slices.

Given that the traffic of the different tenants is not homogeneous among the different cells in the NG-RAN and it can fluctuate over the time, this paper considers that the RSMF includes a capacity sharing function, which distributes the available capacity among the different RSIs across the different cells. This function dynamically tunes the capacity share $\boldsymbol{\sigma_t(k)}$ for each tenant k at each time step t in order to adapt to the spatial and temporal traffic variations among the different cells, minimise SLA breaches (i.e. violations) in the system and optimise the capacity utilisation of the different cells. To this end, the capacity share $\boldsymbol{\sigma_t(k)}$ of tenant k is defined as $\boldsymbol{\sigma_t(k)}=[\sigma_t(k,1),\ldots, \sigma_t(k,n), \ldots, \sigma_t(k,N)]$, where each component $\sigma_t(k,n)$ corresponds to the proportion of the total capacity $C_T(n)$ (i.e. proportion of the total PRBs $N_T(n)$) in cell n provided to that tenant during time step t and ranges $0 \le \sigma_t(k,n) \le MCBR(k,n)/C_T(n)$. Note that the capacity share solution in a cell cannot exceed the total capacity of the cell, so that $\sum_{k=1}^{K} \sigma_t(k,n) \le 1$. The capacity share $\boldsymbol{\sigma_t(k)}$ needs to be dynamically updated in order to adapt to the traffic demands in time steps of duration Δt.

III. Multi-Agent Reinforcement Learning Approach

Considering the complexity and dynamics of the capacity sharing function, the computation of $\boldsymbol{\sigma_t(k)}$ has been addressed by a Multi-Agent Reinforcement Learning (MARL) approach, where each RL agent is associated to a tenant k in the system and centrally learns the policy to tune $\boldsymbol{\sigma_t(k)}$ dynamically by interacting with the environment. As agents need to learn continuously from the network environment and large state and action spaces are expected, a Deep Q-Network (DQN) based algorithm is designed as the RL method.

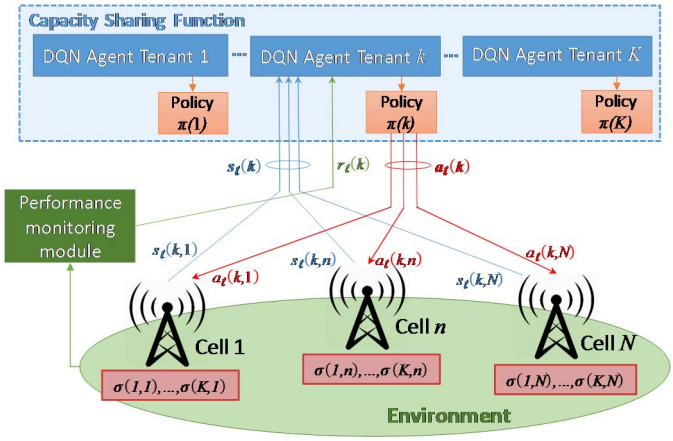

Fig. 1 MARL training scheme

The solution scheme during the learning process of the agents, which is called *training*, is shown in Fig. 1, where the agents of the different tenants are included and the interactions between the agent of tenant k and the different cells in the environment are depicted. At each time step t, each agent obtains the state $\boldsymbol{s_t(k)}$ from the environment and, based on the policy $\pi(k)$, triggers an action $\boldsymbol{a_t(k)}$ to tune $\boldsymbol{\sigma_t(k)}$. Moreover, a reward signal $r_t(k)$ is provided to the k-th agent as a result of the last action $\boldsymbol{a_{t-1}(k)}$. This reward characterizes the obtained performance. In the following, the different components of the designed MARL approach are described in detail.

A. State

At each time step t, the tenant k obtains its state from the environment, denoted as $\boldsymbol{s_t(k)} = [s_t(k,1),\ldots, s_t(k,n), \ldots, s_t(k,N)]$, where each element $s_t(k,n)$ corresponds to the state of the tenant in cell n. $s_t(k,n)$ is defined by the triple $\langle \rho_t(k,n), \sigma_t(k,n), \sigma_{ava,t}(n) \rangle$, where $\rho_t(k,n)$ is the PRB occupation proportion of tenant k in cell n and $\sigma_{ava,t}(n)$ is the available capacity share in cell n not assigned to any tenant, given by $\sigma_{ava,t}(n) = 1 - \sum_{k=1}^{K} \sigma_t(k,n)$.

B. Action

The k-th tenant's agent triggers a joint action for all the cells, defined as $\boldsymbol{a_t(k)} = [a_t(k,1),\ldots, a_t(k,n), \ldots, a_t(k,N)]$, which is composed of the cell-specific action $a_t(k,n)$ for $n=1...N$. This $a_t(k,n)$ is defined as the increment of the capacity share $\sigma_t(k,n)$ of tenant k to be applied in the following time step in cell n. The action $a_t(k,n)$ can take three different values $a_t(k,n) \in \{\Delta, 0, -\Delta\}$, which allows incrementing, maintaining or decrementing $\sigma_t(k,n)$ in steps of Δ gradually without experiencing abrupt changes, by following:

$$\sigma_t(k,n) = \sigma_{t-1}(k,n) + a_t(k,n) \quad (1)$$

Note that the action space for $\boldsymbol{a_t(k)}$ corresponds to all the possible combination vectors of the three possible actions for

each of the cells. Then, the number of possible actions for each tenant is 3^N. Initially, all components of $\boldsymbol{\sigma_{t=0}(k)}$ are initialized to:

$$\sigma_{t=0}(k,n)=SAGBR(k)/(\sum_{k=1}^{K} SAGBR(k)) \quad (2)$$

Then, the actions $\boldsymbol{a_t(k)}$ triggered by each of the tenants are considered and $\boldsymbol{\sigma_t(k)}$ is updated by applying (1) for each cell. Note that, when the chosen action $a_t(k,n)$ by the k-th agent in the n-th cell forces $\sigma_t(k,n)$ out of its bounds (i.e. $\sigma_t(k,n)<0$ or $\sigma_t(k,n)>MCBR(k,n)/C_T(n)$), the last $\sigma_{t-1}(k,n)$ value is maintained.

C. Reward

In order to assess how good was the last action $\boldsymbol{a_{t-1}(k)}$ performed for the previous state $\boldsymbol{s_{t-1}(k)}$, a reward $r_t(k)$ is provided to the k-th tenant's agent. The reward accounts for both aspects, namely the SLA satisfaction and the capacity utilisation, as explained in the following.

The first component of the reward is the SLA satisfaction factor of tenant k, $\gamma_{SLA}(k)$, which is the ratio between the requested and provided capacity, defined as:

$$\gamma_{SLA}(k)= \min\left(\frac{SThr_t(k)}{\min\left(\sum_{n=1}^{N}\min(O_t(k,n),MCBR(k,n)),SAGBR(k)\right)},1\right) \quad (3)$$

where $O_t(k,n)$ is the offered load by the tenant k in the cell n during the last time step t, understood as the required capacity, and $SThr_t(k)$ is the system aggregated throughput of tenant k. Equation (3) is valid for $O_t(k)>0$. For $O_t(k)=0$, $\gamma_{SLA}(k)=1$.

The second component of the reward is the capacity utilisation factor, $\gamma_{ut}(k)$, which aims at minimising the overprovisioning of capacity and is defined as:

$$\gamma_{ut}(k)=\frac{SThr_t(k)}{\sum_{n=1}^{N} C_T(n)\cdot\sigma_t(k,n)} \quad (4)$$

Then, the reward obtained at time step t by tenant k considers the weighted product of its SLA factor, the summation of the SLA factor of the other tenants and the capacity utilisation factor of tenant k, that is:

$$r_t(k)=\gamma_{SLA}(k)^{\varphi_1}\cdot\left(\frac{1}{K\text{-}1}\sum_{\substack{k'=1\\k'\neq k}}^{K}\gamma_{SLA}(k')\right)^{\varphi_2}\cdot\gamma_{ut}(k)^{\varphi_3} \quad (5)$$

where φ_1, φ_2 and φ_3 are the weights of each component.

D. DQN Agent

The proposed approach considers that each tenant is associated to a DQN agent, which centrally learns the policy $\pi(k)$ that dynamically configures $\boldsymbol{\sigma_t(k)}$. For this purpose, the original DQN algorithm in [20] has been particularised, according to state, action and reward definitions previously introduced.

DQN aims at finding the optimal policy $\pi^*(k)$ that maximises the discounted cumulative future reward (i.e., $\sum_{j=0}^{\infty}\gamma^j r_{t+j+1}(k)$, where γ is the discount factor ranging $0\leq\gamma\leq1$). This is achieved by obtaining the optimal action-value function $Q^*(\boldsymbol{s(k)}, \boldsymbol{a(k)})$, which is the maximum expected discounted cumulative reward starting at time step t from $\boldsymbol{s(k)}$ when taking the action $\boldsymbol{a(k)}$ and following the policy $\pi(k)$, given by:

$$Q_k^*(\boldsymbol{s(k)},\boldsymbol{a(k)}) = E\left[r_{t+1}(k)+\gamma \max_{a'(k)} Q_k^*(\boldsymbol{s_{t+1}(k)},\boldsymbol{a'(k)})\middle|\begin{matrix}\boldsymbol{s_t(k)}=\boldsymbol{s(k)}\\\boldsymbol{a_t(k)}=\boldsymbol{a(k)}\end{matrix}\right] \quad (6)$$

In DQN, $Q_k^*(\boldsymbol{s(k)}, \boldsymbol{a(k)})$ is approximated by a deep neural network (NN) with weights θ, denoted as $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$. In order to update $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$, the DQN agent of the k-th tenant is composed of the following elements:

- **Evaluation NN ($Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$)**: corresponds to the main approximation function of the expected reward function $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)})$.
- **Target NN ($Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta^-)$):** a separated NN with weights θ^- is used to obtain the Time Difference (TD)-Target, computed as $r_t(k)+\gamma \max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_t(k)},\boldsymbol{a'(k)},\theta^-)$. This NN is updated every M steps with the weights of the Evaluation NN $\theta^-=\theta$. The architecture of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta^-)$ is the same as the one of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$.
- **Experience Dataset ($D_l(k)$):** for each time step t, the experience tuple $e_t=<\boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)}>$ is stored into a dataset $D_l(k)$ of length l. Then, a mini-batch of experiences $U(D_l(k))$ of length J is randomly selected from $D_l(k)$ to update $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$.

The training of each DQN agent is divided into two parts: the data collection and the update of weights θ. In order to collect data from the environment, *Algorithm* 1 is performed in steps of Δt. For the time step t, a new action $\boldsymbol{a_t(k)}$ is firstly obtained considering the current observed state $\boldsymbol{s_t(k)}$ by an ε-Greedy policy based on $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$. Moreover, the reward $r_t(k)$ is collected and the experience $e_t=<\boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)}>$ is stored in the dataset $D_l(k)$.

The update of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ is performed in parallel to the data collection by iteratively updating $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$, according to *Algorithm* 2. For each sample $e_j=<\boldsymbol{s_{j-1}(k)}, \boldsymbol{a_{j-1}(k)}, r_j(k), \boldsymbol{s_j(k)}>$ extracted from the mini-batch of experiences $U(D_l(k))$, it is performed the upgrade of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ with the objective of reducing the mean squared error (MSE) loss function:

$$L(\theta)= E[(r_j(k)+\gamma \max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_j(k)},\boldsymbol{a'(k)},\theta^-) - Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta))^2] \quad (7)$$

Based on $L(\theta)$, the weights θ are updated according to the gradient descent of $L(\theta)$, $\nabla L(\theta)$, which can be obtained by:

$$L(\theta)=E[(r_j(k)+\gamma \max_{\boldsymbol{a'(k)}} Q_k(\boldsymbol{s_j(k)},\boldsymbol{a'(k)},\theta') - Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta)) \nabla_\theta Q_k(\boldsymbol{s_{j-1}(k)},\boldsymbol{a_{j-1}(k)},\theta)] \quad (8)$$

Then, the weights in the $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ network are upgraded according to:

$$\theta \rightarrow \theta+\tau \nabla L(\theta) \quad (9)$$

where τ is the learning rate.

Algorithm 1. Data collection for training at step *t* for tenant *k*

Generate random ε'.
Collect global state $\boldsymbol{s_t(k)}$ by obtaining $s_t(k,n)$ for $n=1...N$ cells.
Obtain reward $r_t(k)$ as a result of last action $\boldsymbol{a_{t-1}(k)}$.
If $\varepsilon'<\varepsilon$
  Choose a joint random action $\boldsymbol{a_t(k)}$.
Else
  Obtain action through greedy policy $\boldsymbol{a_t(k)}= argmax_{\boldsymbol{a(k)}'} Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)}',\theta)$.
End if
If $D_l(k)$ is full (l samples are stored), remove the oldest one.
Store experience $e_t=<\boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)}>$ in $D_l(k)$.

Algorithm 2. NN update for tenant *k*

Randomly sample a minibatch of experiences $U(D_l(k))$ from $D_l(k)$ of length J.
For j=1..J
  Update evaluation network $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)}',\theta)$ by (9) according to experience $e_j=<\boldsymbol{s_{j-1}(k)}, \boldsymbol{a_{j-1}(k)}, r_j(k), \boldsymbol{s_j(k)}>$.
End for
Every M steps, update $\theta=\theta'$.

E. Multi-Agent operation

The proposed approach considers multiple agents that interact with a common environment. A collaborative scheme is selected, as the reward in (5) takes into consideration the SLA of all tenants.

After initialising the different elements of the DQN agents, the collection of experiences is performed synchronously for all agents in steps of Δt. At each time step t, each agent k observes its state $\boldsymbol{s_t(k)}$, receives $r_t(k)$ as a result of the last action, selects an action $\boldsymbol{a_t(k)}$ and stores experiences $e_t =< \boldsymbol{s_{t-1}(k)}, \boldsymbol{a_{t-1}(k)}, r_t(k), \boldsymbol{s_t(k)} >$ as previously explained according to *Algorithm* 1. Once the actions of all the agents are obtained in a time step, they are communicated to the cells and (1) is applied to obtain the capacity $\boldsymbol{\sigma_t(k)}$ to adopt at the next time step. Note that, since the agents of the different tenants select their actions independently from the others, the joint actions of all the agents in cell n may involve exceeding the cell capacity (i.e. $\sum_{k=1}^{K} \sigma_t(k,n) >1$). In this case, the actions of the tenants that have selected either decreasing or maintaining capacity (i.e. $a_t(k,n) \in\{-\Delta,0\}$) are first applied. After this, if there is no available capacity (i.e., $\sigma_{ava,t}(n)=0$), the actions of tenants that were willing to increase are not applied, so $\sigma_t(k,n)$ is not changed with respect to the previous time step. On the contrary case, i.e. when $\sigma_{ava,t}(n)>0$, $\sigma_{ava,t}(n)$ is distributed among those tenants with $a_t(k,n)= \Delta$ proportionally to their $SAGBR(k)$ value, as long as they are not already provided with more than $SAGBR(k)$.

In parallel, the update of $Q_k(\boldsymbol{s(k)},\boldsymbol{a(k)},\theta)$ for each tenant is carried out by following *Algorithm 2*.

IV. PERFORMANCE EVALUATION

This section evaluates the proposed capacity sharing function by assessing its performance in diverse traffic conditions.

A. Considered scenario

The considered scenario is composed of a single cell that provides service to two different tenants, denoted *Tenant* 1 and *Tenant* 2. The scenario is characterised and configured as indicated in Table I.

The training has been performed by considering a training dataset for the DQN agents to learn how to behave under different traffic load conditions. This dataset is composed of 330 synthetic offered load patterns of the two tenants during a day in the cell of Table I, which has been proved to be sufficient for the training of the model. In each of the patterns, the offered load is computed in periods of 3 minutes, so each pattern consists of 480 steps. The different patterns of each tenant have been combined in order that the DQN agents can visit several states so that the learning can be achieved. The simulation parameters for the training of the DQN agents are detailed in Table II. The results have been obtained by developing the whole model with the library *TF-Agents* [21].

B. Results

The evaluation of the trained model has been performed by considering two scenarios, denoted as *Scenario A* and *Scenario B*, with the different offered load patterns included in Fig. 2 and Fig. 3, respectively, which show the temporal evolution of the aggregated offered load $O_t(k)$ by the two tenants in the cell during a day. *Scenario A* corresponds to an offered load pattern where *Tenant* 1 requires more capacity than *Tenant* 2 during the

TABLE I. SCENARIO CONFIGURATION

Parameter		Cell configuration
PRB Bandwidth (B)		360 kHz
Number of available PRBs ($N_T(n)$)		51 PRBs
Average spectral efficiency		5 b/s/Hz
Total cell capacity $C_T(n)$		91.8 Mb/s
$SAGBR(k)$	Tenant 1	55Mb/s (corresponding to 60% of available capacity)
	Tenant 2	36Mb/s (corresponding to 40% of available capacity)
$MCBR(k)$	Tenant 1	73 Mb/s (corresponding to 80% of available capacity)
	Tenant 2	

TABLE II. SIMULATION PARAMETERS FOR TRAINING

Parameter	Value
Initial collect steps	2000
Number of learning steps	160000
Experience Replay buffer maximum length (l)	100000
Mini-batch size (J)	100
Discount factor(γ)	0.9
Learning rate (τ)	0.001
ε value (ε-Greedy)	0.1
Neural network architecture	Single layer with 100
Step time (Δt)	3 min
$a(k,n)$ increment/decrement step (Δ)	0.3

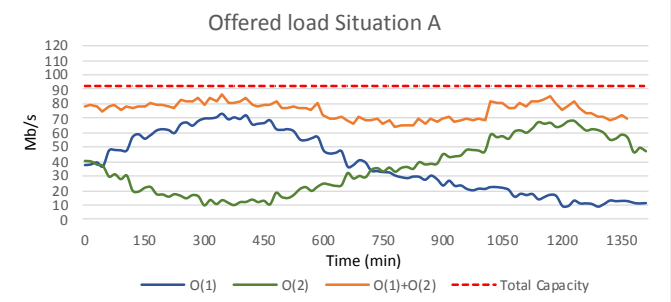

Fig. 2. Offered load of Tenant 1 and Tenant 2 in Scenario A.

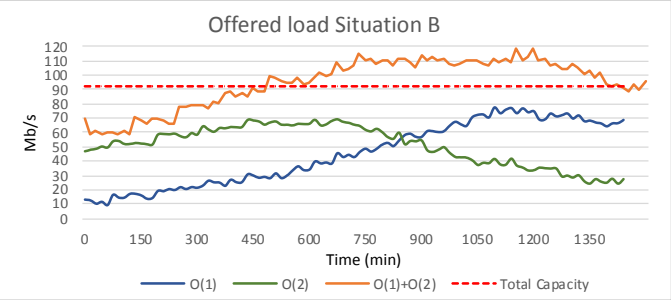

Fig. 3. Offered load of Tenant 1 and Tenant 2 in Scenario B.

morning while the contrary case is given during the afternoon. *Scenario B* presents the contrary case, where the roles of *Tenant* 1 and *Tenant* 2 are reversed. Notice that, in *Scenario A*, the aggregated load by both tenants does not exceed the total capacity (C_T=91.8Mb/s) at any time while in *Scenario B* the total capacity is exceeded for a long period of time.

Fig. 4 and Fig. 5 compare the assigned capacity, obtained by $C_{ass}(k,n) = C_T(n)\cdot\sigma_t(k,n)$, with the offered load and the *SAGBR*(k) for *Tenant* 1 and *Tenant* 2 in *Situation A* and *Situation B*, respectively. In *Situation A*, the offered load of both tenants is generally served as there is enough capacity to fulfil the both of them. The capacity sharing mechanism provides the demanded capacity to both tenants, including those cases when the offered loads of *Tenant* 1 and *Tenant* 2 exceed *SAGBR*(1) and *SAGBR*(2), respectively, making efficient use of the available capacity and exhibiting the capability to exploit the complementarities in the traffic profiles between both tenants. In *Situation B,* similar behaviour is observed but lower assigned capacity than offered is obtained during the periods where more capacity than available is requested. In those periods, the required capacity is given to the tenants whose offered load $O(k)$ is lower than *SAGBR*(k), which shows that the capacity sharing function assures the *SAGBR*(k) established in the SLA.

For benchmarking purposes, the performance obtained in *Situation B* by the MARL based capacity sharing function has been compared to two reference capacity sharing solutions: *Reference* 1, which considers *SAGBR*(k) as the maximum capacity that can be assigned to tenant k in any case, and *Reference* 2, which considers that both tenants share the overall capacity independently on *SAGBR*(k) and in the case that more capacity than available is requested, the capacity is distributed

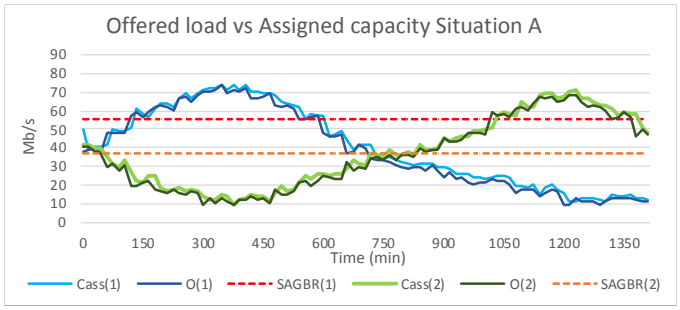

Fig. 4. Offered load $O(k)$ vs assigned capacity in Situation A

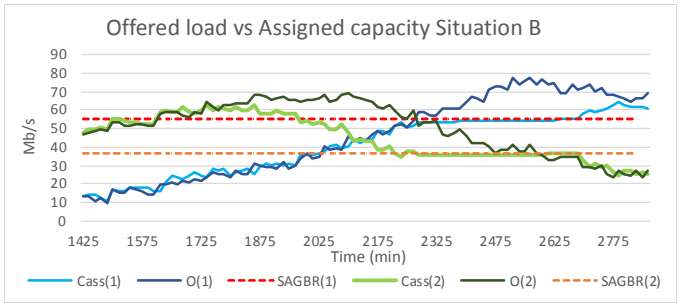

Fig. 5. Offered load $O(k)$ vs assigned capacity in Situation B

among tenants according to their offered load. Fig. 6 and Fig. 7 show the cumulative density function (CDF) of SLA satisfaction ratio $\gamma_{SLA}(k)$ for *Tenant* 1 and *Tenant* 2, respectively. While *Reference* 1 always fulfils the SLA, *Reference* 2 presents much lower SLA satisfaction, given that both tenants are provided with lower throughput than required when the offered load exceeds the total capacity. The proposed MARL solution is able to improve the SLA satisfaction of *Reference* 2 by serving all the offered load $O(k)$ as long as it is lower than *SAGBR*(k). Moreover, Fig. 8 compares the CDF of the percentage of the system utilisation of the three approaches. The system utilisation is defined as the ratio between the aggregate throughput of the two tenants and the total cell capacity. *Reference* 1 presents the lowest system utilisation, as no more than *SAGBR*(k) is provided even though there is enough capacity unused in the system to satisfy the offered load $O(k)$. However, the proposed MARL approach substantially improves the utilisation, with a performance very close to *Reference* 2. These results show how the presented approach allows satisfying the SLA and making efficient use of the resources, achieving a good trade-off between the two benchmarking schemes.

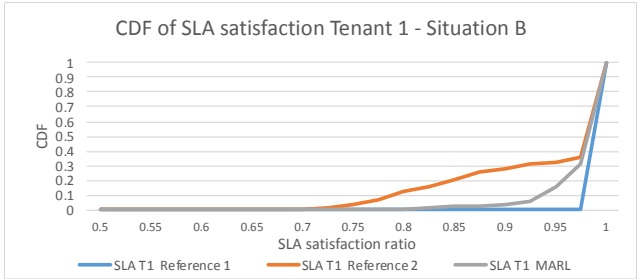

Fig. 6. Cumulative Density Function (CDF) of SLA satisfaction of Tenant 1 in Situation B

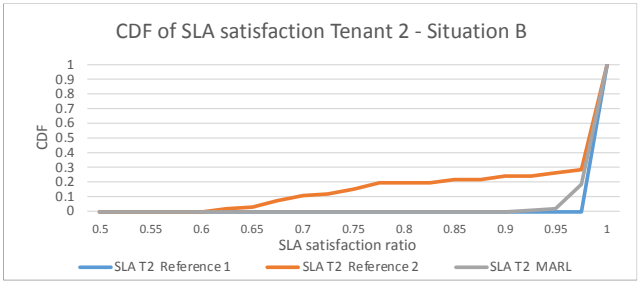

Fig. 7. CDF of SLA satisfaction of Tenant 2 in Situation B

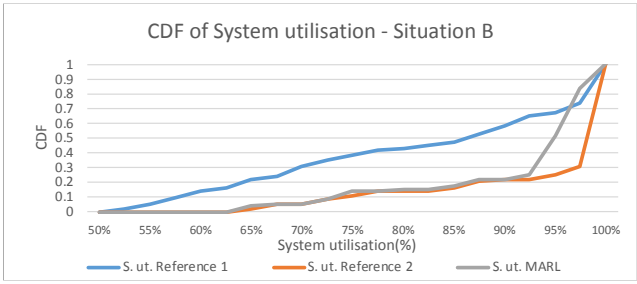

Fig. 8. CDF of overall system utilisation of in Situation B.

V. CONCLUSIONS

This paper has presented a novel capacity sharing function for multi-tenant scenarios, which has been approached by means of a collaborative multi-agent reinforcement learning based on Deep Q-Network. Each Deep Q-Network agent has been associated to a different tenant, which interacts with a common environment, aiming at satisfying the traffic requirements while fulfilling the service level agreement. The multi-agent approach solution has been trained and evaluated by analysing the capacity assigned to each of the tenants, the service level agreement satisfaction and the system utilisation. Results have shown that agents satisfactorily adapt the assigned capacity to each tenant to their traffic requirements while achieving high service level agreement satisfaction and efficiently using the available capacity in the system.

ACKNOWLEDGEMENT

This work has been supported by the Spanish Research Council and FEDER funds under SONAR 5G grant (ref. TEC2017-82651-R), by the European Commission's Horizon 2020 research and innovation program under grant agreement #871428, 5G-CLARITY project, and by the Secretariat for Universities and Research of the Ministry of Business and Knowledge of the Government of Catalonia under grant 2019FI_B1 00102.

REFERENCES

[1] P. Rost, et al. "Mobile network architecture evolution toward 5G," in *IEEE Communication Magazine*, May, 2016.

[2] K. Samdanis, X. Costa-Perez and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," in IEEE Communications Magazine, vol. 54, no. 7, pp. 32-39, July 2016.

[3] R. Ferrús, O. Sallent, J. Pérez-Romero, R. Agustí, "On 5G Radio Access Network Slicing: Radio Interface Protocol Features and Configuration", *IEEE Communications Magazine*, May, 2018,pp.184-192.

[4] D. Marabissi and R. Fantacci, "Highly Flexible RAN Slicing Approach to Manage Isolation, Priority, Efficiency," in *IEEE Access*, vol. 7, pp. 97130-97142, 2019.

[5] J. Gang and V. Friderikos, "Optimal resource sharing in multi-tenant 5G networks," 2018 *IEEE Wireless Communications and Networking Conference (WCNC)*, Barcelona, 2018, pp. 1-6.

[6] P. L. Vo, M. N. H. Nguyen, T. A. Le and N. H. Tran, "Slicing the Edge: Resource Allocation for RAN Network Slicing," in *IEEE Wireless Communications Letters*, vol. 7, no. 6, pp. 970-973, Dec. 2018.

[7] A. S. D. Alfoudi, S. H. S. Newaz, A. Otebolaku, G. M. Lee and R. Pereira, "An Efficient Resource Management Mechanism for Network Slicing in a LTE Network," in *IEEE Access*, vol. 7, pp. 89441-89457, 2019.

[8] J. Pérez-Romero, O. Sallent, R. Ferrús and R. Agustí, "Profit-Based Radio Access Network Slicing for Multi-tenant 5G Networks," 2019 *European Conference on Networks and Communications (EuCNC)*, Valencia, Spain, 2019, pp. 603-608.

[9] Ö. U. Akgül, I. Malanchini and A. Capone, "Dynamic Resource Trading in Sliced Mobile Networks," in *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 220-233, March 2019.

[10] J. Shi, H. Tian, S. Fan, P. Zhao and K. Zhao, "Hierarchical Auction and Dynamic Programming Based Resource Allocation (HA&DP-RA) Algorithm for 5G RAN Slicing," 2018 *24th Asia-Pacific Conference on Communications (APCC)*, Ningbo, China, 2018, pp. 207-212.

[11] P. Caballero, A. Banchs, G. De Veciana and X. Costa-Pérez, "Network Slicing Games: Enabling Customization in Multi-Tenant Mobile Networks," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 662-675, April 2019.

[12] J. Pérez-Romero, O. Sallent, R. Ferrús and R. Agustí, "Self-optimized admission control for multitenant radio access networks," 2017 IEEE 28th *Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Montreal, QC, 2017, pp. 1-5.

[13] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang and L. Wang, "Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," in *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 44-52, June 2019.

[14] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38, Nov. 2017.

[15] R. Li et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," in *IEEE Access*, vol. 6, pp. 74429-74441, 2018.

[16] C. Qi, Y. Hua, R. Li, Z. Zhao and H. Zhang, "Deep Reinforcement Learning With Discrete Normalized Advantage Functions for Resource Management in Network Slicing," in *IEEE Communications Letters*, vol. 23, no. 8, pp. 1337-1341, Aug. 2019.

[17] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah and W. Jiang, "Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks," in *IEEE Access*, vol. 7, pp. 45758-45772, 2019.

[18] G. Sun, K. Xiong, G. O. Boateng, D. Ayepah-Mensah, G. Liu and W. Jiang, "Autonomous Resource Provisioning and Resource Customization for Mixed Traffics in Virtualized Radio Access Network," in *IEEE Systems Journal*, vol. 13, no. 3, pp. 2454-2465, Sept. 2019.

[19] 3GPP TS 28.531 v16.2.0, "Management and orchestration; Provisioning; (Release 16)," Dec. 2019.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[21] S. Guadarrama, et. al (2018).TF-Agents: A library for Reinforcement learning in TensorFlow. Available at: https://github.com/tensorflow/agent