Impact Analysis of Training in Deep Reinforcement Learning-based Radio Access Network Slicing

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

***Abstract*—Deep Reinforcement Learning (DRL) has recently emerged as a promising technique to deal with different problems in the 5G and beyond Radio Access Network (RAN). The practical implementation of DRL solutions in the real network embraces a training process that is fundamental to materialize the expected benefits associated to these techniques. However, little effort has been devoted in the literature to analyze this training process when applying DRL in the RAN. In an effort to contribute to fill this gap, this paper presents an impact analysis of the training process on the obtained performance by a DRL solution for RAN slicing. To this end, a methodology to specify the training dataset is introduced together with the definition of relevant metrics. Then, the paper presents different simulation results to determine the features of the training dataset that allow a satisfactory training and a high performance of the obtained policy when applied during the inference stage.**

Keywords—RAN slicing, Capacity Sharing, Deep Reinforcement Learning, Training.

I. INTRODUCTION

Deep Reinforcement Learning (DRL) tools, which combine reinforcement learning with deep learning, have emerged as potential solutions to effectively address various problems in 5G and beyond networks (see e.g., the survey [1]) due to its capability to deal with decision-making problems with high-dimensional inputs. One area of application of DRL is network slicing, which is a key feature of the 5G architecture that allows sharing a common network infrastructure among multiple tenants (i.e., mobile network operators, vertical industries, etc.) with heterogeneous needs [2]. This is achieved by providing each tenant with an end-to-end logical network, referred to as network slice, that is customized and optimized to the tenant needs and that runs on top of the same physical network infrastructure than the slices of other tenants. In this area, DRL is identified as a potential solution to address the capacity sharing problem when slicing the Radio Access Network (RAN). This problem consists in dynamically allocating the radio resources of the RAN nodes among the tenants while satisfying the fluctuating traffic demands and achieving an efficient resource utilization. The use of DRL for this problem has been profusely studied in recent years [3]-[9].

While a solid background has been generated on methods and algorithmic solutions for decision making, the detailed analysis of the above state-of-the-art enables the identification of a key research gap, namely, how the training stage of DRL is conducted and it is distinguished from the inference stage (i.e., executing a trained policy on the real network). Indeed, previous DRL-based solutions for capacity sharing in [3]-[7] have not explicitly distinguished the training and inference stages. Only the work in [8] makes this distinction and discusses on the generalization capability of the learnt policies when conducting the training with a large number of situations. However, none of the above works has provided insights into the features and requirements of the data used for training and their impact on the achieved performance.

To the best of the authors' knowledge, this paper is the first to conduct a quantitative impact analysis of training on the performance of the learnt policies by DRL-based capacity sharing solutions. The main contributions and novelties are summarized as follows: (1) The paper identifies the features of training datasets used for DRL-based capacity sharing and quantifies their impact on the training and inference stages. (2) The paper introduces a methodology for building training datasets with given features to properly configure the training environment. (3) Several metrics to quantify the impact of the built training datasets are defined. (4) Finally, results assessing the methodology when applied to the Deep Q-Network - Multi-Agent Reinforcement Learning (DQN-MARL) capacity sharing solution of [9] are provided. The rest of the paper is organized as follows: Section II describes the proposed methodology while Section III includes the results. Finally, Section IV summarizes the conclusions and the future work.

II. METHODOLOGY FOR TRAINING IMPACT ANALYSIS

The intrinsic online learning behaviour of DRL becomes a major challenge for its application in the capacity sharing problem. Due to the high impact of capacity sharing decisions on the network performance and efficiency, erroneous actions selected by DRL agents during the trial-and-error learning process may lead to unacceptable performance degradations in the real network. Also, since capacity sharing solutions operate at time scales in the order of seconds to minutes, long training durations would be required for the DRL agents to experience a sufficient variety of situations in the real network. To tackle this problem, it is possible to train the DRL agent in a simulated training environment that emulates the behaviour of the real network. The training environment is configured by means of a training dataset that determines the situations that the DRL agent will experience during the training stage. In this way, the agent can accumulate sufficient experience in a wide variety of situations in order to learn the decision-making policy that will be later on applied on the real network during the so-called inference stage.

An adequate training needs to ensure that a sufficiently representative number of situations are encountered by the DRL agent during the training stage, so that it can properly generalize from them in order to make the best decisions during the inference stage. In the case of capacity sharing, since DRL agents decide on the capacity assigned to each tenant according to their traffic needs over time, the configuration of the training environment should be based on the time evolution of the traffic demand (i.e., offered load) of each tenant. In this paper, we model this evolution as a temporal pattern of one-day duration. Then, a training dataset is defined as a set of temporal patterns that will be used as inputs to the training environment and experienced by the

DRL agents during the training stage. The way to generate this dataset and the metrics for analysing the impact of training is described in the following.

A. *Training stage*

Let us consider the availability of J temporal patterns, where each pattern j represents the time evolution of the offered load of a tenant during a day, $O_j(t)$, in a cell. The offered load is the required capacity in bits/s of a tenant normalized to the total cell capacity, so that it falls in the range [0,1]. It is obtained for t=0…T in time steps of duration Δt. Offered load temporal patterns can be based on historic real data extracted from the real network or can be generated synthetically.

Every temporal pattern is characterized in terms of two features: (1) Dynamic Range (D), which is the difference between the maximum and the minimum values of $O_j(t)$; (2) Level (L), which corresponds to the average value of $O_j(t)$. For both features, the temporal patterns are classified into *High* and *Low*, leading to four categories, each one given by a combination of classes of D and L, e.g., a category is {D=*High*, L=*High*}. The number of patterns in each category should be sufficiently represented. In the case that the available temporal patterns are not enough, which may occur e.g., when scarce real data measurements are used, the number of patterns can be augmented by adding a random component (e.g., uniformly distributed) to each of the available temporal patterns, emulating the traffic fluctuations.

The available J temporal patterns are used to build the training dataset. Since capacity assignment decisions for a tenant depend on both the offered load of that tenant and the offered loads of the rest of the tenants, the training of a DRL agent considers the offered load of a target tenant, denoted as tenant 1, and the aggregate offered load of the rest of the tenants, which for simplicity can be referred here to as tenant 2. Hence, the training dataset is composed of a total of Q pairs of one temporal pattern for tenant 1 and another one for tenant 2, which are selected from the set of J patterns. The Q pairs are randomly ordered to generate the time evolution of the offered load that is injected to the simulated environment during the training process.

The features of the selected patterns for each tenant determine the actual offered load values of tenant 1, o_1, and tenant 2, o_2, that are included in the training dataset. Then, to assess the range of offered load values in the training dataset, the *coverage* of a dataset, *COV*, is defined. It is the ratio between the number of offered loads (o_1,o_2) included in the training dataset over the space of possible load values, which considers the load in the range [0,1] and discretized in steps of Δo. A training dataset with *COV* value close to 100% will allow that the DRL agent experiences almost all the possible offered load pairs of both tenants during the training stage.

During the training stage the DRL agent will be making decisions in accordance with the experienced offered load values included in the training dataset and progressively updating the decision-making policy based on the outcome of these decisions. This process is executed until the DRL agent achieves convergence in the learnt policy. To define the convergence condition, the well-known loss metric (see definition in [1]) is considered. The loss tends to decrease and stabilize (i.e., loss fluctuations are small) when the DRL agent successfully learns a good approximation of the optimum policy, and thus, convergence is achieved. To quantify convergence, the loss is averaged in periods of M training time steps. Then, the loss variation $v(m,W)$ at period m over the window of the last W periods is defined as the difference between the maximum and the minimum average loss during the last W periods divided by the total average loss over all the W periods. Based on this, the criterion considered here is that convergence is achieved at period m when $v(m,W)$, is lower than a given threshold v_{th}. The number of time steps until reaching this condition determines the convergence time, t_c. Moreover, to assess the training stability, the standard deviation of the average loss in the window $[t,t+W_S)$, denoted as $\sigma(t,W_S)$, is also obtained.

B. *Inference stage*

After the training is completed, the policy can be applied in the real network environment to assign the capacity to the tenants according to their actual experienced offered loads. The achieved performance of the policy during inference depends on the generalization capability of the trained policy to adapt to offered load situations not included in the training dataset but also on the similarity between the offered loads included in the training dataset and those experienced during inference. To measure this similarity, the *percentage of coincidence*, *PC*, is defined as the percentage of the observed offered load values (o_1, o_2) during inference that had been included in the training dataset more than d_{min} times. For illustrative purposes, Fig. 1 shows the offered loads experienced during an inference stage, represented as bullets, on top of a training dataset 2D-histogram. The training dataset contains patterns that belong to the category {D=*Low*, L=*Low*} for both tenants and presents a *COV*=22.6%. The offered load values experienced during inference present a high similarity to the ones included in the training dataset, achieving a *PC*=95% when d_{min}=100.

During the inference stage, the performance of the trained policy is also assessed by obtaining the aggregated reward of both tenants at each time step t, $R(t)$, which rates how good were the capacity allocation decisions for tenant 1 and tenant 2 for the actual experienced loads.

III. Performance Evaluation

This section assesses the training performance of different training datasets generated according to the methodology described in Section II when applied to the training of a specific capacity sharing solution.

A. *DQN-MARL capacity sharing solution*

The DQN-MARL capacity sharing solution considered for studying the impact of training is described in [9]. The

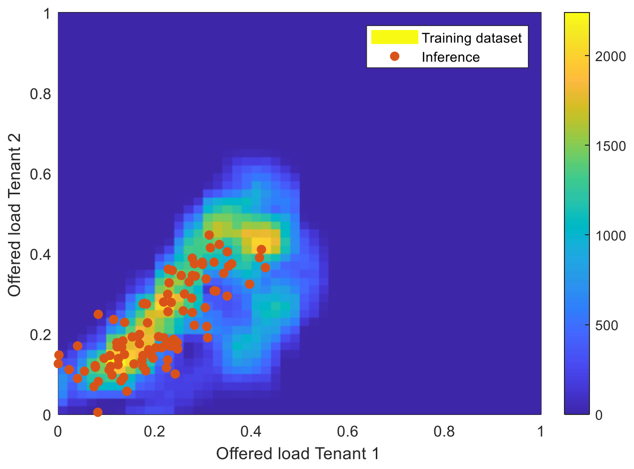

Fig. 1. Training dataset vs inference stage experienced load.

solution has been designed to dynamically distribute the available capacity in a RAN infrastructure composed of N cells among K tenants, each of them provided with a RAN slice. The solution aims at efficiently using the capacity in the cells while satisfying the traffic demands of the tenants and the Service Level Agreement (SLA) established for each of them. The SLA of the k-th tenant is defined in terms of the Scenario Aggregated Guaranteed Bit Rate, $SAGBR_k$, to be provided across all cells to the tenant if requested and the Maximum Cell Bit rate, $MCBR_k$, that can be provided to the tenant in each cell. To this end, in the considered multi-agent solution each DQN agent is associated to a different tenant and learns the policy that dynamically tunes the capacity share (i.e., the fraction of the cell capacity that is provided to the tenant in a cell) in the different cells. The tuning of the capacity shares is performed in time steps of duration Δt depending on the actions selected by the DQN agent for each cell, which can be to increase/decrease the capacity share of the cell in an action step of Δ or to keep it equal.

B. *Considered scenario and datasets*

The scenario considered for evaluation is comprised of a single cell, which serves the users of K=2 tenants, denoted as *Tenant* 1 and *Tenant* 2. The scenario considers that the cell has a capacity c_n=117 Mb/s and the established SLAs are $SAGBR_1$=70.2 Mb/s and $SAGBR_2$=46.8 Mb/s, corresponding to the 60% and the 40% of the cell capacity, respectively, and $MCBR_1$= $MCBR_2$=93.6 Mb/s, corresponding to the 80% of c_n. The DQN parameters considered for training are a time step duration Δt = 3 min and an action step Δ=0.03. The rest of DQN parameters used for training are those of [9].

To build the training datasets, a total of J=8400 offered load temporal patterns of one-day duration have been considered and classified according to the dynamic range D and offered load level L. Fig. 2 depicts a representative set of the available temporal patterns, the central mark on each box indicating the median of temporal pattern, the bottom and top edges of the box indicating the 25th and 75th percentiles, respectively, and the whiskers extending to the most extreme data points. The representation of Fig. 2 shows that temporal patterns with diverse dynamic ranges and offered load levels are included in the dataset. For instance, the pattern j=18 belongs to the category {D=*High*, L=*Low*} and the pattern j=9 to {D=*Low*, L=*Low*}. The rest of patterns have been obtained from those of Fig. 2 by adding a random component uniformly distributed in the range [-0.05,0.05] until having a total of 2100 patterns for each of the possible 4 categories.

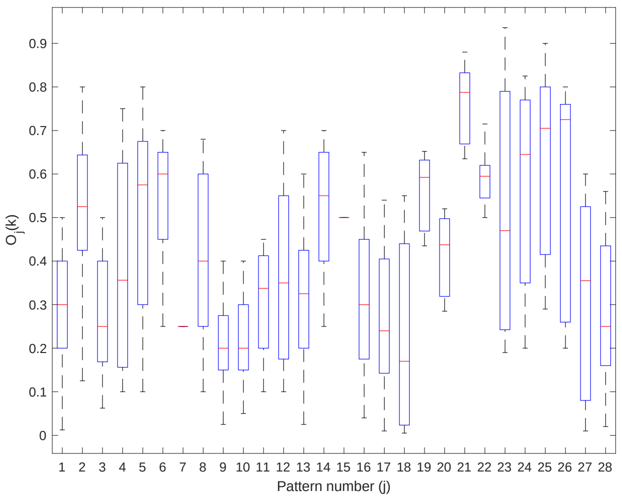

Fig. 2. Temporal patterns analysis

C. *Impact of the training dataset on the training stage*

This section studies the impact of the features of training datasets on the training process. To this end, the training of the DQN-MARL capacity sharing model has been performed for two datasets, denoted as *Dataset* 1 and *Dataset* 2, both of them with Q=4200 pairs of temporal patterns of each tenant. *Dataset* 1 is a specific dataset that includes temporal patterns that only belong to the category {D=*Low*, L=*Low*} for both tenants. Instead, *Dataset* 2 consists in a generic dataset that includes temporal patterns belonging to all the categories. The coverages of *Datasets* 1 and 2 are COV=22.57% and COV=92%, respectively. The training of both datasets has been performed 50 times, each time with a different order of the pairs of patterns included in the datasets. This allows mitigating potential effects that the random order in which the offered loads are experienced by the DQN agent and the randomness of the ε-Greedy policy, which selects a random action with probability ε, may have on the training evolution.

Table I includes the 95th percentile of the convergence time t_c and the average and 95th percentile of the standard deviation of loss $\sigma(t,W_S)$ over the different executions of the training for datasets 1 and 2. The computations assume a loss average period M=5000 time steps, a variation window W=$10\cdot10^4$ time steps, a convergence time threshold v_{th}=0.06 and a standard deviation window W_S=$50\cdot10^4$ time steps. Results indicate that convergence is faster with *Dataset* 1 than with *Dataset* 2, being the 95th percentile of the convergence time with *Dataset* 1 approximately 25% less than with *Dataset* 2. The reason is that *Dataset* 1, which only includes patterns from a specific category, covers a lower number of states than *Dataset* 2, so a shorter number of iterations are required to go through all of them. It is also observed that the standard deviation of the loss is also smaller when using *Dataset* 1 than when using *Dataset* 2. Both the average and the 95th percentile obtained with *Dataset* 2 are approximately twice than those with *Dataset* 1, which reflects that a much more stable learning is achieved for *Dataset* 1 due to its smaller coverage. Therefore, it is concluded that the coverage of a dataset, and thus, its level of specificity or generality, has a direct impact on the process of training, achieving faster convergence and higher stability those datasets with low COV.

D. *Impact of the training dataset on the inference stage*

In this section, an impact analysis of the features of the training dataset on the learnt policy is conducted. To perform this analysis considering a variety of coverage values, 16 training datasets have been defined, exhibiting levels of COV ranging from as low as 0.04% to as high as 92%. Then, the policy learnt with each training dataset has been analysed during the inference stage. For the inference stage, a simulator of the real network has been considered that takes as input 80 different pairs of temporal patterns for tenant 1 and tenant 2 that belong to a wide range of dynamic ranges and offered load levels.

In order to assess the performance of a policy during the inference stage, the assigned capacity share by the policy in each time step is compared to the optimum value, obtained by means of an exhaustive search algorithm. Specifically, in each

Table I. Performance of the training stage

	Dataset 1	**Dataset 2**
95th percentile of t_c (time steps)	$75\cdot10^4$	$105\cdot10^4$
Average $\sigma(t,W_S)$	$9.25\cdot10^{-4}$	$21\cdot10^{-4}$
95th percentile of $\sigma(t,W_S)$	$20\cdot10^{-4}$	$56\cdot10^{-4}$

time step the aggregate reward of both tenants is computed for all the possible values of capacity shares of Tenant 1 and Tenant 2, discretized in steps of Δ. Then, the one that achieves the maximum aggregate reward of both tenants is selected as the optimum. Based on this, the optimality ratio, defined as the ratio between the aggregate instant reward $R(t)$ obtained by the learnt policy and the optimum aggregate instant reward, is computed in each time step. The optimality ratio is recorded for each pair of offered loads of tenant 1 and 2 (discretized in steps of Δo=0.02) observed during the inference stage. Then, we define the metric denoted as $P_{0.9}$ as the percentage of offered load pairs with optimality ratio higher than 0.9.

Fig. 3 shows the value of $P_{0.9}$ for each learnt policy as a function of the coverage *COV* of the training dataset used to obtain the policy. The trend line of Fig. 3 reveals that, for *COV* smaller than approximately 25%, $P_{0.9}$ grows when increasing *COV*, while for *COV* higher than 25%, it flattens at around 80%. This reflects the strong generalization capability of the considered DQN approach since with a training dataset of relatively low *COV* value such as 25%, it achieves a performance close to the optimum for most of the offered load levels in the inference stage. In other words, for *COV* above 25%, during the execution of the DQN-based algorithm in a practical scenario, appropriate decisions will be taken even though the actual offered loads levels have not been observed during the training stage.

Fig. 3 also shows that the performance in terms of $P_{0.9}$ for datasets with low *COV* is much worse, and suffers strong variations depending on the actual offered loads that were included in the training dataset. In this case, the percentage of coincidence *PC* metric between the offered loads in the inference stage and the training datasets becomes especially relevant for the analysis. To show this, the learnt policy for a training dataset with *COV*=8% has been assessed for the case of experiencing a *PC*=100% and *PC*=32% during the inference stage, considering d_{min}=100 in the computation of *PC*. It is obtained that for the case of *PC*=100%, $P_{0.9}$=94% is achieved whereas for the case of *PC*=32%, only $P_{0.9}$=48% is obtained. This proves that the achieved performance for datasets with low *COV* has high dependency on the *PC*.

IV. Conclusions and Future Work

This paper has performed an impact analysis of the training in Deep Reinforcement Learning (DRL)-based capacity sharing solutions for RAN slicing. A methodology is proposed that specifies how to create training datasets with given features to train DRL models for simulation-based training environments and defines some performance metrics to assess the impact of the training datasets on the training and on the inference stages.

The methodology has been used to assess the performance in a multi-agent reinforcement learning solution based on Deep Q-Network. Results have shown that: (a) The coverage of states (*COV*) of the training dataset affects the convergence time and the convergence stability: training datasets with low coverages achieve lower convergence times and more stability than datasets with large coverages of states. Specifically, it has been obtained that a dataset with *COV* of about 22% achieves convergence in 75% of the time than a dataset with *COV* of 92%. (b) Training datasets with coverage values of around 25% are sufficient to achieve a satisfactory training process that provides good performing policies. In this respect, it has been able to achieve optimality ratios higher than 0.9 in around 80% of the observed offered load levels during the inference. (c) In general, training datasets with coverage values below 25% lead to poor performing policies. In this case, the performance is highly dependent on the percentage of coincidence between the offered loads of the training dataset and those experienced during the inference stage.

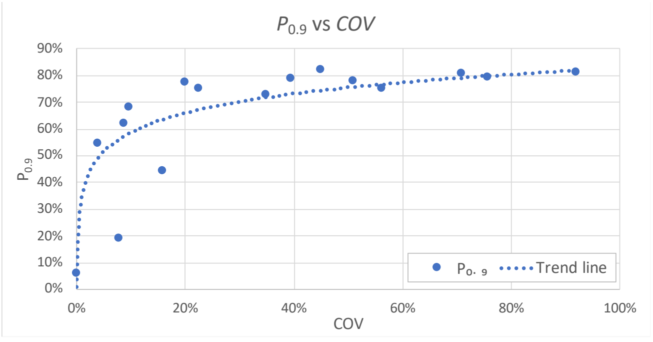

Fig. 3. $P_{0.9}$ with respect to COV.

While the analysis performed in this paper has allowed analyzing the impact of the training dataset features on the achieved performance for the case of DRL-based capacity sharing solutions, future work envisages the study of the impact and requirements of training for a wider range of problems in the RAN (e.g., load balancing, coverage optimisation, etc.). This would allow generalizing the requirements and the impact of training in DRL-based solutions for RAN management, which is fundamental for establishing a framework for the application of DRL solutions in practical 5G and beyond systems.

Acknowledgment

This work has been supported by the Spanish Research Council and FEDER funds under SONAR 5G grant (ref. TEC2017-82651-R).

References

[1] N. C. Luong et al., "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Comm. Surveys & Tutorials*, vol. 21, no. 4, pp. 3133-3174, Fourthquarter 2019.

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

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

[4] Y. Hua, R. Li, Z. Zhao, X. Chen and H. Zhang, "GAN-Powered Deep Distributional Reinforcement Learning for Resource Management in Network Slicing," *IEEE Journal on Sel. Areas in Comms*., Feb. 2020.

[5] 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.

[6] T. Li, X. Zhu and X. Liu, "An End-to-End Network Slicing Algorithm Based on Deep Q-Learning for 5G Network," in *IEEE Access*, 2020.

[7] G. Sun, G. O. Boateng, D. Ayepah-Mensah, G. Liu and J. Wei, "Autonomous Resource Slicing for Virtualized Vehicular Networks With D2D Communications Based on Deep Reinforcement Learning," in *IEEE Systems Journal*, vol. 14, no. 4, pp. 4694-4705, Dec. 2020.

[8] Y. Abiko, et.al, "Flexible Resource Block Allocation to Multiple Slices for Radio Access Network Slicing Using Deep Reinforcement Learning,",*IEEE Access*, vol. 8, 2020.

[9] I. Vilà, J. Pérez-Romero, O. Sallent, A. Umbert, "A Multi-Agent Reinforcement Learning Approach for Capacity Sharing in Multi-tenant Scenarios," in *IEEE Trans. on Veh. Tech.*, July 2021.