2021 IEEE 46th Conference on Local Computer Networks (LCN) | 978-1-6654-1886-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/LCN52139.2021.9524965

Transfer Learning-Based Accelerated Deep Reinforcement Learning for 5G RAN Slicing

Ahmad M. Nagib*, Hatem Abou-zeid†, Hossam S. Hassanein*
**School of Computing*, *Queen's University*, Kingston, Canada, {ahmad, hossam}@cs.queensu.ca
†*Ericsson*, Ottawa, Canada, {hatem.abou-zeid}@ericsson.com

Abstract—**Deep Reinforcement Learning (DRL) algorithms have been recently proposed to solve *dynamic* Radio Resource Management (RRM) problems in 5G networks. However, the slow convergence experienced by traditional DRL agents puts many doubts on their practical adoption in cellular networks. In this paper, we first discuss the need to have accelerated DRL algorithms. We then analyze the exploration behavior of various state-of-the-art DRL algorithms for slice resource allocation, and compare it with the traditional 5G Radio Access Network (RAN) slicing baselines. Finally, we propose a transfer learning-accelerated DRL-based solution for slice resource allocation. In particular, we tackle the challenge of slow convergence by transferring the policy learned by a DRL agent at an expert base station (BS) to newly deployed agents at target learner BSs. Our approach shows a remarkable reduction in convergence time and a significant performance improvement compared with its non-accelerated counterparts when tested against multiple traffic load variations.**

Index Terms—**cellular networks, RAN slicing, resource allocation, deep reinforcement learning, transfer learning, accelerated reinforcement learning, 5G.**

I. Introduction

5G networks and beyond are being designed and built to support a diverse set of network scenarios. This can be realized thanks to the advancements in radio access technologies (RATs), communication paradigms, and cell and user equipment (UE) types. However, these heterogeneous technologies and devices add several layers of complexities to cellular networks. Accordingly, a mobile network operator (MNO) is required to optimize more parameters to be able to provide a wide set of services. The process of optimally setting such parameters is not straight forward as it depends on time-varying factors. A significant percentage of these parameters provide MNOs with control over the available radio resources, and hence such optimization process is referred to as radio resource management (RRM).

The growing complexities of next-generation wireless networks require intelligent and automated solutions. As a result, machine learning (ML) techniques have been proposed to solve many RRM-related problems in the wireless networks literature. More recently, many researchers are paying more attention to Deep Reinforcement Learning (DRL) algorithms due to their evaluative feedback capabilities. Such capabilities allow them to adapt to the multifaceted complexities of the dynamic Radio Access Network (RAN) environments. DRL agents interact with such environments and build an approximate model based on the sampled experience progressing toward, at least, a near optimal solution. As a result, many DRL-based solutions have been proposed for RRM-related use cases such as RAN Slicing, power control, handover control, link adaptation, and packet scheduling [1].

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: STPGP 521432-2018.

DRL algorithms, however, are known to face multiple challenges, especially during the initial exploration phase. One major issue is the slow convergence experienced by a standard DRL agent. This issue forces DRL agents to stay in the exploration phase for a significantly long time. In such period, a DRL agent is learning by taking actions that were not explored previously. This situation increases the chances of running into extreme drops in system performance. By taking random unexplored actions, the DRL agent will most probably end up choosing a non-optimal action given a certain system state. This is naturally unavoidable but can only be tolerated in cases that do not require real-time feedback.

In RAN systems, however, end users' quality of experience (QoE) can be significantly affected by the potential drops in performance. In 5G RAN slicing, service-level agreements (SLAs) define the minimum QoE requirements that can be tolerated by each provided service. For instance, a DRL-based RAN slicing controller that keeps exploring for a long time will experience several non-optimal actions. This will result in potentially violating the defined SLAs, and this consequently translates into monetary penalties. Hence, resource allocation in 5G RAN Slicing requires a fast policy-learning scheme that minimizes the exploration phase duration given any QoE measure. Such scheme will allow the RAN slicing controller to adapt, in real-time, to the time-varying wireless channel conditions and user activities of the provided services.

The aforementioned issue faced by DRL agents, among others, puts many doubts on their practical adoption to solve dynamic RRM problems such as slice resource allocation in commercial networks. Such challenges are rarely tackled in the wireless networks literature or research. On a related note, transfer learning (TL) has recently achieved some noticeable results in the wireless communications domain [2]. TL is a paradigm that generally focuses on reusing knowledge gained while solving a learning task. TL applies the acquired knowledge to a different but related task. This can provide a fast and efficient way to train an artificial neural network (ANN) model exploiting the gained experience rather than learning from scratch. We propose to use TL to tackle the challenge of slow convergence in DRL-based RAN slicing resource allocation. This includes the reuse of the policy learned by a DRL agent at a source expert base station (BS) to accelerate the training of a newly deployed DRL agent at a target learner BS.

The main contributions of this paper can be summarized as follows:

• We highlight the need of accelerated DRL-based RRM solutions and discuss the techniques that address the challenge of slow convergence of DRL algorithms in the context of RRM. We argue that it is essential that researchers address this challenge in order for DRL algorithms to find their way to RRM commercial solutions.

• Moreover, we develop a DRL-based solution to 5G RAN slicing resource allocation problem to analyze the exploration and reward convergence behavior of various state-of-the-art DRL algorithms. The simulation infrastructure includes an OpenAI GYM-compatible Reinforcement Learning (RL) environment that is shared publicly to enable fellow researchers to further address the slow convergence challenge among other challenges.

• We finally propose a transfer learning-based approach to tackle the challenge of slow convergence in DRL-based slicing resource allocation. We propose to accelerate the DRL-based solution by transferring the policy learned by a DRL agent at an expert base station to newly deployed agents at learner base stations instead of learning from scratch. We analyze the exploration behavior with and without the proposed acceleration approach in learner BSs of different traffic loads reusing the same policy learned at an expert BS.

To the best of our knowledge, this is the first study to employ transfer learning to tackle the challenge of slow convergence in DRL-based RAN Slicing. The rest of the paper is organized as follows. Section II discusses the related work. The system model and the proposed approach is described in Section III. Section IV provides the reader with the experimental evaluation setup. In Section V we discuss the results. Lastly, our work is concluded and some future directions are presented in Section VI.

II. Related Work

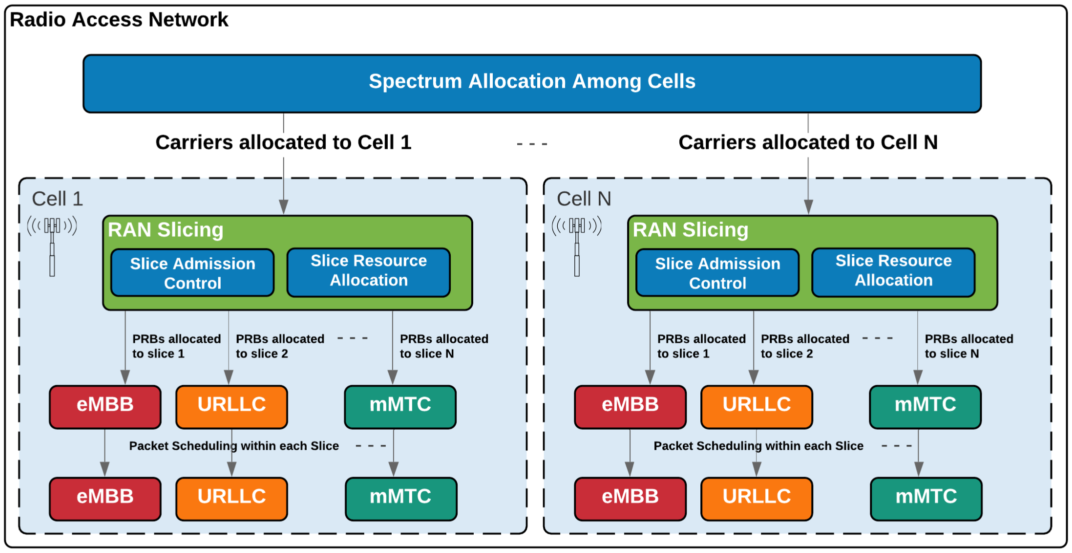
It may take a DRL agent thousands of learning iterations to converge to an optimal or a near optimal configuration for a given RRM functionality. This is mainly due to the exploratory behavior of the DRL agents and the non-stationary nature of RAN systems. This may not be a problem in case of simulation-based RL training. Nevertheless, real networks deployments of RRM solutions can not tolerate such long exploration phase. Recently, several researchers started to partially address this issue as part of their RL-based solutions. For instance, a heuristic mechanism is used in [3] to guide the RL exploration phase by exploiting the existing knowledge. The heuristic approach is proposed as part of a user scheduling and resource allocation solution and it acts as an indicator function on the RL agent's action space. Therefore, the agent takes actions based on both the value function and the traditional scheduling rules using a weighted sum method.

The authors of [4] propose an approach to guide their RL solution to the cellular networks resource scheduling problem. They make use of expert knowledge gained from an existing traditional solution, namely the proportional fair (PF) scheduling algorithm. The authors suggested that the DRL agent's performance during the exploration phase can be improved in terms of convergence speed. They propose to deploy both the PF algorithm and their RL agent as competitors. The authors suggest calculating the RL reward by comparing the system performance resulting from the two competitors' actions.

Moreover, the work in [5] proposes an ML framework to accelerate DRL agent convergence in the context of downlink resource allocation for ultra reliable low latency communication (URLLC). The authors use generative adversarial neural networks (GANs) to pretrain the DRL framework using a mix of real and synthetic data. This approach allows the DRL agent to gain an offline experience through being exposed to a wide range of network conditions before being deployed in a real network. It is also suggested that such approach can help the DRL agent to recover in real-time whenever it is exposed to a drastic condition in the real network.

A meta-learning approach is proposed in [6] as a way to tune an RL solution. The authors suggest that a meta-tuned RL agent experience a faster convergence in unseen environments. They propose to solve an optimal coverage problem using an RL agent that controls drone base stations (DBSs). DBSs are required in that case to provide uplink connectivity to ground users given their stochastic access requests. Furthermore, a collaborative learning framework for 5G RAN slicing resource scheduling is proposed in [7]. The framework combines DRL and deep supervised learning to solve an online resource scheduling problem and a large time-scale resource allocation problem respectively. With respect to improving DRL convergence speed, the authors proposed to use Asynchronous Advantage Actor-Critic (A3C) method suggesting that it improves the convergence speed compared against the Actor Critic (AC) method.

The authors of [8] propose an interference management decentralized RL solution to share the spectrum among heterogeneous cells. As part of their solution, they propose an enhanced initialization procedure to overcome the slow convergence of tabular Q-learning. Given a newly experienced state, the procedure typically starts with updating the Q-value of the action taken. Additionally, the authors propose to estimate the costs of the other actions given that same state and update their values in the Q-table accordingly. More significantly, a DRL-based energy consumption optimization strategy is proposed in [9]. The authors combine relational DRL with transfer learning (TL) to address the insufficient generalization ability and the slow recovery when exposed to new conditions. The authors suggest that the scheme combining DRL and TL speeds up learning when compared with training from scratch in new scenarios.



Radio Access Network
Spectrum Allocation Among Cells
Carriers allocated to Cell 1
Carriers allocated to Cell N
Cell 1
Cell N
RAN Slicing
Slice Admission Control
Slice Resource Allocation
PRBs allocated to slice 1
PRBs allocated to slice 2
PRBs allocated to slice N
eMBB
URLLC
mMTC
Packet Scheduling within each Slice

Fig. 1: Overview of RAN Slicing.

III. Transfer Learning-Accelerated Deep Reinforcement Learning-Based RAN Slicing

A. Radio Access Network Slicing

Both RAN and core network are considered part of the end-to-end network slicing, each with a slightly different optimization goal. In this paper, we mainly focus on the RAN part of network slicing. RAN slicing objective is to share the physical infrastructure among several services. RAN slicing is mainly concerned with two RRM functionalities, slice admission control and resource allocation. Slice admission control allows an infrastructure provider to accept or deny a service provider's slice request. While resource allocation in RAN slicing is concerned with assigning the available PRBs to the admitted slices approved by the admission control function. An overview of RAN slicing and its main functionalities are depicted in Fig. 1.

The available resources at a given time are significantly affected by the stochastic channel quality. Moreover, they are affected by the time-varying user demands for the provided services. The traffic demand of each type of service is dynamic and can not be easily predicted, particularly in the short term. At the beginning of a slicing window, the available limited resources are assigned among the admitted slices. These allocated resources are expected to enable the services provided by the admitted slices to comply with their different QoS requirements given the dynamic network conditions. The exact requirements are defined by the SLAs and should not be violated by the infrastructure provider, otherwise monetary penalties can be enforced. The aforementioned points pose many challenges for DRL-based RAN slicing solutions and prevent RAN slicing from tolerating long DRL exploration phase.

B. System Model

As mentioned in Section III-A, resource management for network slicing can be considered from several perspectives. In this paper, we focus on the downlink case of the radio access part, and more specifically the RAN slicing resource allocation problem. The main goal is to allocate the limited PRBs to the available slices, maintaining an acceptable spectral efficiency (SE) while keeping an acceptable delay, and generally, quality of experience (QoE) satisfaction. The slice resource allocation problem can be mathematically formulated as follows:

There exists a set of s parameters, described by the vector $x \in \mathbb{R}^{\mathrm{m}}$, that needs to be optimized. In our case, s reflects the number of slices sharing the available bandwidth W, and hence, these parameters control the number of PRBs allocated to each slice. At a given instance, a RAN slicing controller decides to choose a specific slicing PRB allocation configuration, i.e. $x(a)$, out of the x possible configurations where $a = 1, 2, 3...X$. Based on such decision, the system performance is affected.

For the purpose of this paper, the system performance is represented in terms of throughput and latency of the admitted slices and can be represented by a single value as follows:

$$f(x(a), \theta(t)) = \alpha T + \beta L \in \mathbb{R}^{\mathrm{m}} \qquad (1)$$

where T and L are the throughput and latency of the available slices, while α and β represent the importance of the throughput and latency for each slice respectively. Moreover, $\theta(t)$ is the system state at time t. This function is unknown to the controller, therefore it can not explicitly relate an input to an output and can only observe the function's outcome. The system state can be represented by the traffic load, the channel quality or other external factors that might affect the RAN system performance. The majority of these variables evolve in a way that is hard to infer theoretically especially in time scales of seconds or shorter.

The RAN slicing controller explores different slice allocation configurations and observes the corresponding system performance in search of the optimal configuration that maximizes the performance, i.e.

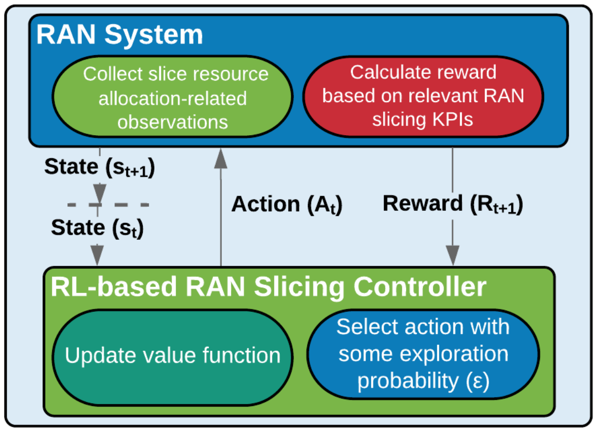

RAN System
Collect slice resource allocation-related observations
Calculate reward based on relevant RAN slicing KPIs
State (s_{t+1})
State (s_t)
Action (A_t)
Reward (R_{t+1})
RL-based RAN Slicing Controller
Update value function
Select action with some exploration probability (ε)

Fig. 2: RL-based slicing controller–environment interaction.

$$\hat{x} = \underset{x}{\text{argmax}}\ f(x) \tag{2}$$

C. Reinforcement Learning-based Slicing Resource Allocation

The most important feature that distinguishes reinforcement learning (RL) from the other types of machine learning is that it evaluates the actions taken rather than specifying correct actions [10]. RL does not require complete knowledge of the RAN system or prior knowledge of the network. Both requirements are inefficient and infeasible for a stochastic environments such as 5G RANs. Thus, the DRL is an attractive approach to solve the previously defined slicing resource allocation problem [11].

A DRL-based RAN Slicing controller, that is the RL agent, typically interacts with the RAN environment bidirectionally as seen in Fig. 2. At any given slicing time step, the DRL agent observes the RAN system state and chooses an action to take, i.e. resource allocation for each slice. The action taken changes the RAN environment in a way and the RL agent receives feedback in terms of a reward value that represents the system performance.

The DRL agent aims at maximizing the reward feedback that it gets from its interaction with the 5G RAN system. The reward function is designed by network experts to guide the DRL agent's search for the optimal policy π. It is often represented in terms of a weighted sum of relevant network's key performance indicators (KPIs). This way, the DRL-based RAN slicing controller indicates how good the action taken was. This is estimated based on agent's sampled experience from interacting with the RAN environment in a real time and dynamic-open control fashion.

D. Mapping to Reinforcement Learning

Based on the model defined in Section III-B, A DRL agent would take an action at the beginning of each slicing window to decide the PRB allocation for each slice; $x(a) = (w_1, ..., w_S)$, subject to $w_1 + ... + w_S = W$. In this paper, such an action is taken based on the number of packets within a specific time window for each slice, $p = (p_1, ..., p_S)$, i.e. the observed system state. We define the reward function as the weighted sum of the throughput and the latency. The goal is to maximize the long-term reward expectation, that is,

$$E\{f(x(a), \theta(t))\} \tag{3}$$

where the notation $E(.)$ is the expectation of the argument,

$$\begin{aligned} &\underset{x}{\text{argmax}}\ E\{f(x(a), \theta(t))\} \\ &= \underset{x}{\text{argmax}}\ E\{\alpha T(x(a), \theta(t)) + \beta L(x(a), \theta(t))\} \\ &= \underset{x}{\text{argmax}}\ E\{\alpha T(w, d) + \beta L(w, d)\} \end{aligned} \tag{4}$$

where $\theta(t)$, the system state at time t, represents the demand for each provided service, that is, $d = (d_1, ..., d_S)$ where d_i is a given traffic model for service i. This allows us to learn a policy π that takes a state $s \in$S as input and outputs an action, where $a = \pi(s) \in A$. The key challenge to solve (4) lies in the time-varying demand in terms of traffic models and number of users for each service type. The optimal solution for the problem can be precisely calculated by carrying out an exhaustive search. In such case, all the possible allocations should be considered at the beginning of every slicing window and the corresponding system performance should be noted. This approach, however, is computationally very expensive and practically infeasible. Hence, DRL is a good alternative to solve the problem. The exact RAN slicing RL Design parameters are highlighted in Table I-(b).

E. Transfer Learning-Accelerated RAN Slicing

Slow convergence of DRL algorithms is a challenge that relates to the number of learning time steps it takes the RL-based RAN slicing controller to find a good set of slice allocation configurations given a certain system state. The DRL agent needs to observe a representative variety of the RAN system's possible states several times. The learning happens by iteratively updating a value function until convergence. This process is referred to as exploration phase. The value function gives an estimate of the expected return if the agent starts in a given state or state-action pair, and then acts according to a particular policy.

Transfer Learning (TL) is widely used in image object classification, where pre-trained top-performing models are used as the basis for image recognition and related computer vision tasks. This includes, but not limited to, initializing an ANN with the architecture and weights from such pre-trained models to solve an object classification problem using a local dataset that might include a different set of objects.

We employ the same concept but in the context of DRL. We propose a TL-accelerated RAN slicing, where the source task is performed by a DRL agent at an expert BS, while the target task is performed by a DRL agent at a learner BS. The DRL agent at an expert BS learns a policy from scratch until converging to a good policy, while the DRL agent at a learner BS reuses the

TABLE I: Simulation Parameters and RL Agent Design Details

(a) RAN Slicing Simulation Parameter Settings			
	Video	**VoLTE**	**URLLC**
Scheduling Algorithm	Round Robin per 0.5 ms slot		
Bandwidth Allocation Window Size	2000 scheduling time slots (1 second)		
Number of Users (Expert BS)	Poisson (Max = 20, Mean = 12)	Poisson (Max = 20, Mean = 12)	Poisson (Max = 7, Mean = 2)
Number of Users (Learner BS)	Poisson (Max = 13, Mean = 8)	Poisson (Max = 36, Mean = 23)	Poisson (Max = 4, Mean = 1)
Packet Interarrival Time Distribution (Expert and Learner BSs)	Truncated Pareto (Mean = 6 ms, Max = 12.5 ms)	Uniform (Min = 0, Max = 160 ms)	Exponential (Mean = 180 ms)
Packet Size Distribution (Expert and Learner BSs)	Truncated Pareto (Mean = 100 Byte, Max = 250 Byte)	Constant (40 Byte)	Truncated Lognormal (Mean = 2 MB, Standard Deviation = 0.722 MB, Max = 5 MB)
Packet Interarrival Time Distribution (Learner BSs)	Truncated Pareto (Mean = 4 ms, Max = 6 ms)	Uniform (Min = 0, Max = 100 ms)	Exponential (Mean = 100 ms)
Packet Size Distribution (Learner BSs)	Truncated Pareto (Mean = 60 Byte, Max = 150 Byte)	Constant (80 Byte)	Truncated Lognormal (Mean = 1 MB, Standard Deviation = 0.5 MB, Max = 2 MB)
(b) RAN Slicing RL Design			
State		The number of packets within a specific time window for each slice $(p_{Video}, p_{VoLTE}, p_{URLLC})$	
Action		Bandwidth allocated to each slice (22 allocation configurations) $(w_{Video}, w_{VoLTE}, w_{URLLC})$, s.t. $w_{Video} + w_{VoLTE} + w_{URLLC} = W$	
Reward		A weighted sum of throughput and latency experienced in a slicing window	
RL Parameters	**RL Algorithms**	DQN, DDQN, Dueling DQN, PPO, AC, A2C + Traditional Slicing Baselines	
	Total Number of Time Steps	Expert BS: 200,000, Learner BS: 60,000	
	Exploration	Expert BS: 0.9, Learner BS: 0.1	
	Exploration Decay	Expert BS: 0.1, Learner BS: 0.01	
	Batch Size	20	

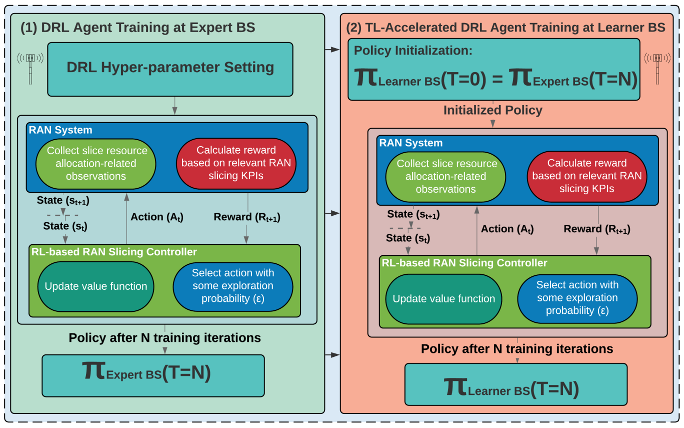

(1) DRL Agent Training at Expert BS
DRL Hyper-parameter Setting
RAN System
Collect slice resource allocation-related observations
Calculate reward based on relevant RAN slicing KPIs
State (s_{t+1})
State (s_t)
Action (A_t)
Reward (R_{t+1})
RL-based RAN Slicing Controller
Update value function
Select action with some exploration probability (ε)
Policy after N training iterations
$\pi_{Expert\ BS}(T=N)$
(2) TL-Accelerated DRL Agent Training at Learner BS
Policy Initialization:
$\pi_{Learner\ BS}(T=0) = \pi_{Expert\ BS}(T=N)$
Initialized Policy
RAN System
Collect slice resource allocation-related observations
Calculate reward based on relevant RAN slicing KPIs
State (s_{t+1})
State (s_t)
Action (A_t)
Reward (R_{t+1})
RL-based RAN Slicing Controller
Update value function
Select action with some exploration probability (ε)
Policy after N training iterations
$\pi_{Learner\ BS}(T=N)$

Fig. 3: TL-accelerated DRL-based RAN Slicing

expert BS's learned policy π to tackle the practical challenge of DRL slow convergence as shown in Fig. 3.

$$\Pi_{LearnerBS}(t=0) = \Pi_{ExpertBS}(t=N) \tag{5}$$

where N is the number of learning iterations carried out by the DRL agent at the expert BS until convergence.

This includes reusing the architecture and weights from the expert agent's trained model. The interaction between a DRL agent and the RAN system is time-consuming and computationally expensive. Reusing a learned policy possibly reduces the dependence on a large number of training samples in the target domain, thus, accelerating the DRL action exploration phase at the learner BSs.

As equation (5) suggests, we propose to transfer $\pi_{ExpertBS}$ to a target learner BS to initialize $\pi_{LearnerBS}$ of its newly deployed DRL agent. The traffic load at a BS can be defined in terms of the number of users, inter-arrival times and packet sizes. We propose to generate different traffic loads at the learner BSs, i.e. $d_{LearnerBS} \neq d_{ExpertBS}$. Therefore, the temporal traffic distribution of a source and a target BS are different, and hence, the source and target tasks are slightly different.

This approach is also valid when the training is done in simulation, while the DRL agent is deployed in real networks. In other words, the policy is learned in simulation rather than real networks. Adopting a simulation-based DRL training often leads to sub-optimal solutions. However, if the policy learned at the virtual expert BS is transferred to a real network learner BS, the state and action spaces can be reduced, and hence, the exploration phase in the target task can be shortened.

IV. Experiments and Evaluation

A. Simulation Environment Settings

Reproducing an existing DRL-based RAN slicing solution is not straightforward due to the absence of DRL-based RRM benchmark environments that can be easily integrated and reused out of the box. Hence, the algorithms and environment implementations will vary. We implemented an OpenAI GYM-compatible RL environment in order to study the exploration behavior of the developed DRL-based RAN slicing controller using various DRL agents and configurations. The created environment will be available on GitHub[1] to allow further

[1]http://www.github.com/ahmadnagib/SARL-RRM

investigation and evaluation of accelerated RL approaches tackling the slow convergence challenge in RAN slicing.

We first evaluate the performance of the DRL solution we adopted to solve equation (4) by simulating a scenario at the expert BS with three types of services; VoLTE, video and URLLC. The prevalent 4G networks mainly classify services into voice and best effort, hence it is hard to have access to real network traces of the services addressed in this paper. We simulate several other scenarios at the learner BSs in order to test the generality of our approach as described in the next sub-section.

The number of users for each service at a given slicing window follows a Poisson distribution as shown in Table I-(a). Additionally, user requests are generated based on the distributions shown in the table. The ones used at the expert BS are similar to those in [12]. In such case, URLLC users generate the largest but the least frequent packets compared with users of the other services. VoLTE users generate the smallest packets, while video packets are the most frequent ones.

Users belonging to the same slice share bandwidth equally. More specifically, a round-robin scheduler is used within each slice at the granularity of 0.5 ms. Moreover, the bandwidth allocation window size is one second. In other words, the DRL agent takes an action to adjust the PRB allocation to each slice every second. We summarize the parameters used to create the environment and train various DRL agents in Table I-(b). As shown in the table, we have evaluated various state-of-the-art DRL algorithms implemented in the Tensorforce Python package[2]. We mainly tested three classes of DRL algorithms, namely Deep Q-Network (DQN), Actor Critic (AC), and Proximal Policy Optimization (PPO). Two other variants of DQN were used, namely Dueling DQN, and Double DQN (DDQN). Additionally, Advanced Actor Critic (A2C), a second variant of AC was investigated.

While analyzing the exploration and reward convergence behavior of the trained DRL agents, they were compared against the following traditional slicing baseline methods :

• Number of Users Forecasting-based Allocation: The number of active users in the upcoming slicing window for each service type is forecasted assuming a perfect predictor. Then the available bandwidth is sliced weighted by the predicted number.

• Number of Packets Forecasting-based Allocation: The number of packet requests in the upcoming slicing window for each service type is forecasted assuming a perfect predictor. Then the available bandwidth is sliced weighted by the predicted number.

• Hard Slicing: The bandwidth is equally distributed among the available slices, i.e. one third of the available PRBs are allocated to each slice.

The three aforementioned methods decide the percentage of PRBs to be allocated to each slice. Afterwards, round-robin scheduling is followed within each slice as in the case of the DRL-based approach.

[2]Tensorforce: a TensorFlow library for applied reinforcement learning - available at http://www.github.com/tensorforce/tensorforce

B. Transfer Reinforcement Learning Settings

The objectives of both the expert and the learner agents are the same. However, as seen in Table I-(a), we generate different traffic loads at the learner base station to explore the approach's capacity for generalization. It is expected that the traffic load at the learner BSs will vary from the expert BS in both the simulation-to-real network and real network-to-real network scenarios. We use the RL mapping defined in Section III-D at both the expert and learner BSs. According to the scenarios defined, the DRL agent action at both expert and learner BSs can be represented as $(w_{Video}, w_{VoLTE}, w_{URLLC})$, s.t. $w_{Video}+w_{VoLTE}+w_{URLLC} = W$. Moreover, the state of both categories of BSs can be represented as $(p_{Video}, p_{VoLTE}, p_{URLLC})$. Finally, based on the system performance function defined in Section III-B, the reward function can be defined as:

$$\begin{aligned} R = \alpha\ (T_{VoLTE} + T_{Video} + T_{URLLC}) \\ + \beta\ (L_{VoLTE} + L_{Video} + L_{URLLC}) \end{aligned} \tag{6}$$

In this study, we give more weight to latency in deciding the system's performance, and hence β is much larger than α. We also assume that the policy learned at expert BS becomes available to the learner BS's DRL agent before the latter starts the exploration process. It is good to note that the issue of slow convergence of the DRL agent is not directly related to the latency defined in the reward function. However, having a shorter exploration phase will lead to faster convergence to better latency performance, given the reward function formulation.

1) Expert Base Station Settings:

a) Reinforcement Learning Agents: We decided to employ the policy learned at the expert BS using the AC agent to be later transferred to initialize all the learner BSs' policies. This decision was based on the results from the first part of this study in Section V-A.

b) Traffic Load Model: We generated one traffic model for the expert BS as seen in the Table I-(a). It is represented in terms of the number of users, inter-arrival times, and packet sizes.

2) Learner Base Stations Settings:

a) Reinforcement Learning Agents: We decided to explore TL's potential to guide the PPO and A2C agents at the learner BSs. The PPO's performance was very close to a random agent and needs to be guided if deployed at a BS to reduce the convergence time and stabilize the exploration phase. Additionally, A2C was also chosen to see TL's ability to further enhance the performance of an already well performing agent.

b) Traffic Load Model: Multiple levels of variations were used at the target learner BSs to reflect the expected variation from the expert BS as seen in Table I-(a). We examined the DRL-agent behavior against three main levels of variation which can be categorized as follows:

- Different Seed: Traffic and number of users are generated based on the same models used at the expert BS but with a different random seed.

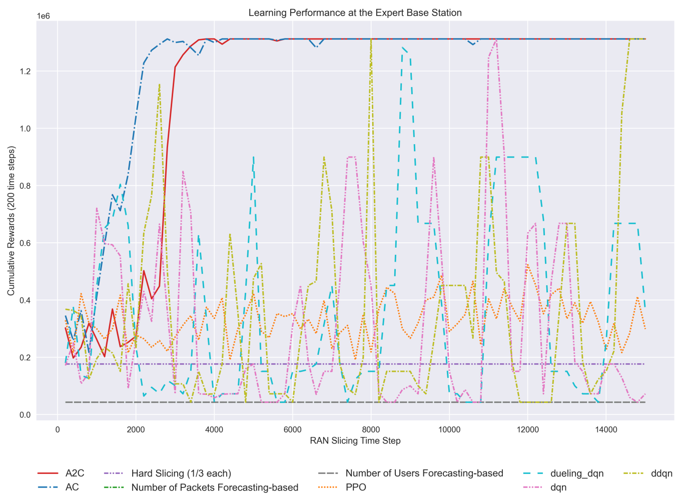


Fig. 4: Comparison of DRL Agents Learning Performance.

- Different Number of Users: Traffic is generated based on a number of users model different from the expert BS and with a different random seed.
- Different Traffic Model: The traffic generated follows models different from the one used at the expert BS. Additionally, the number of users models and random seeds are also different.

V. Results and Discussion

A. Expert Base Station

It is obvious that the three classes of algorithms behave differently with the problem in hand given the settings used. Fig. 4 shows the cumulative reward every 200 slicing resource allocation decisions, i.e. 200 DRL-agent learning time steps. The performance of the PPO algorithm goes randomly up and down. The DQN variants have a different behavior. The change in their performance is less frequent but the jumps are much larger. Finally, the AC variants experience the best performance, both in terms of stable exploration of action space and convergence speed. We can observe the following:

• A2C agent inherits the best behavior in terms of convergence time. However, compared with AC, it takes a longer time to get closer to the best performance. However, it still took the AC variants thousands of learning iterations (around 5800) to converge.

• DQN and PPO agents experience a very poor performance in terms of convergence, and hence receive non-optimal reward value for a long time. Thus, the RAN system will suffer from poor throughput and latency during such long duration, potentially leading to several SLA violations. As previously mentioned, MNOs can not tolerate such convergence delay in practical settings as this translates into monetary penalties.

• AC variants and PPO agents perform much better with respect to performance variations, i.e. they experience more stable exploration.

• All the examined RAN slicing baselines perform poorly as, unlike DRL, they only consider load without paying attention to the latency and actual temporal distribution of such load. Hence, they will potentially perform very badly if other KPIs, such as latency, were of more importance as in the case of the reward function used in this paper.

We meant to use the same initial DRL configurations for all the trained algorithm to highlight the importance of the DRL agent's hyper-parameter tuning. The results indicate the sensitivity and importance of hyper-parameter optimization in accelerating and stabilizing the agents' learning process. The process of hyper-parameter setting is very exhausting and time consuming. Even the automation of such process is not straight forward and is computationally expensive.

B. Learner Base Stations

We simulated the three traffic variations defined in Section IV-B2. The results for different seed, different number of users, different traffic model variations are shown in Fig. 5. It is clear that the proposed TL-accelerated DRL shows performance improvement in all the defined target traffic scenarios at the learner BSs. However, more investigation is needed to study the effect of the TL-based approach on the various RAN slicing scenarios including the nature of the training data. We can observe the following:

• Although A2C has a relatively good performance, the TL-accelerated approach still contributed to the reduction in the convergence time in all the A2C cases. The reduction exceeded 17,000 learning iterations in some cases as seen in Fig. 5b.

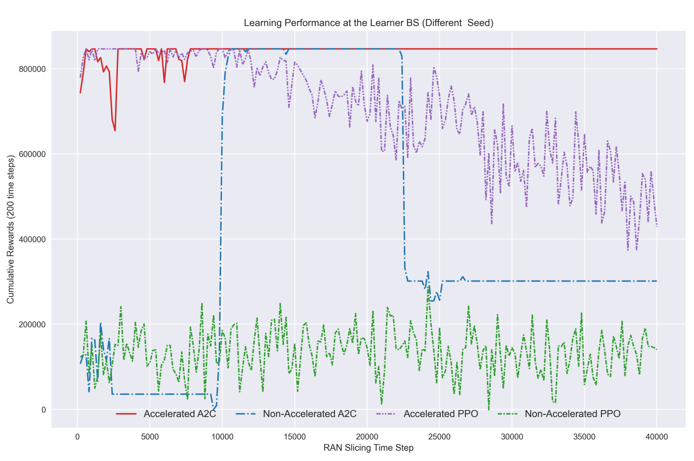
• As seen in Fig. 5a and Fig. 5b, the TL-accelerated approach not only helped the A2C agent in reducing the convergence time, but also in avoiding being stuck with a sub-optimal solution for a long time, and hence converging to a better solution. This is a remarkable feature as it allows TL to guide the poorly-tuned DRL agents.

• Even when there is a big variation in the traffic model such as the case in Fig. 5c and Fig. 5d, the TL-accelerated approach still contributes to the improvement of both A2C and PPO performance. This is promising for MNOs as variations are expected to happen when moving from a simulation-based expert BS to a real network learner BS for instance. However, the PPO agent seems to be not stable and may still need extra investigation and hyper-parameter tuning.

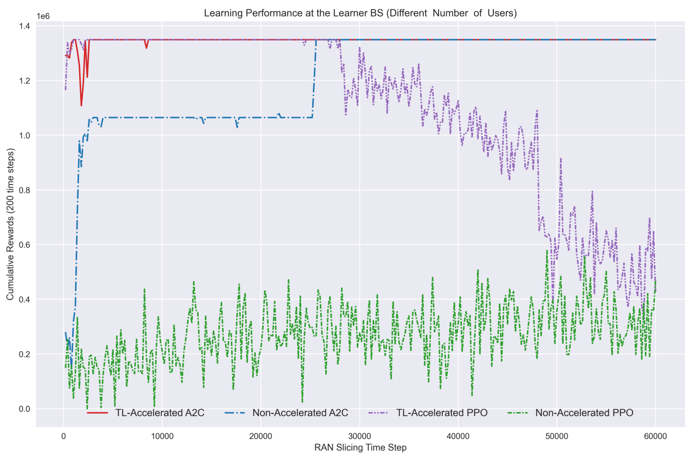
VI. Conclusion and Future Work

In this paper, we developed a DRL-based solution to the 5G RAN slicing resource allocation problem to analyze the exploration and reward convergence behavior of various state-of-the-art DRL algorithms. We then proposed a transfer learning-based approach to accelerate our DRL-based resource allocation solution. To the best of our knowledge, this is the first study to employ transfer learning to accelerate DRL-based RAN Slicing.

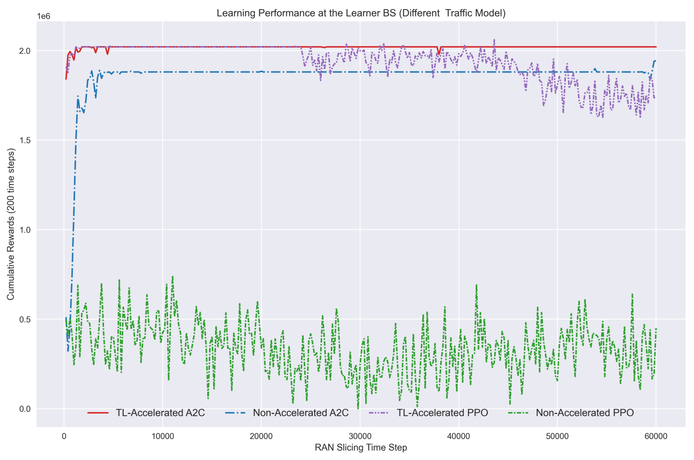
Our approach demonstrated the potential to save thousands of learning iterations at learner base stations. It showed a general pattern of reducing the convergence time and improving the system performance compared with its non-accelerated counterparts tested against multiple traffic load variations. Our approach was furtherly able to enhance the performance of



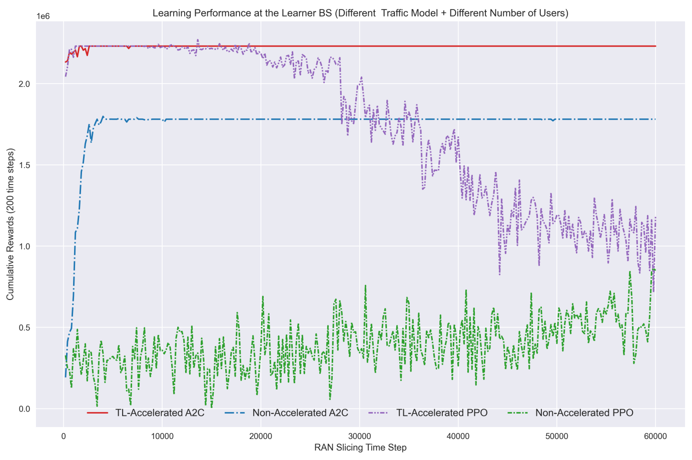
(a) Different Seed



(b) Different Number of Users



(c) Different Traffic Model



(d) Different Traffic Model and Number of Users

Fig. 5: Comparison of Accelerated and Non-accelerated DRL

already well performing agents. It also enhanced the DRL agent's ability to avoid sub-optimal solutions in multiple cases. This is a very promising alternative to the expensive DRL hyper-parameter tuning, especially when considering the deep learning interpretability issues.

We argue that it is essential that researchers address the challenge of slow convergence of DRL agents. In addition, more effort should be directed toward investigating the effect of using real network traces and those based on mathematical models or simulations with respect to acceleration. The challenge of unstable DRL agents' exploration phase is another interesting research problem that should be addressed.

References

[1] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for ai-enabled wireless networks: A tutorial," *arXiv preprint arXiv:2011.03615*, 2020.

[2] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6g wireless communications: Recent advances and future challenges," *IEEE Transactions on Reliability*, 2021.

[3] L. Wang, C. Yang, X. Wang, J. Li, Y. Wang, and Y. Wang, "Integrated resource scheduling for user experience enhancement: A heuristically accelerated drl," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.

[4] J. Wang, C. Xu, Y. Huangfu, R. Li, Y. Ge, and J. Wang, "Deep reinforcement learning for scheduling in cellular networks," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.

[5] A. T. Z. Kasgari, W. Saad, M. Mozaffari, and H. V. Poor, "Experienced deep reinforcement learning with generative adversarial networks (gans) for model-free ultra reliable low latency communication," *IEEE Transactions on Communications*, 2020.

[6] Y. Hu, M. Chen, W. Saad, H. V. Poor, and S. Cui, "Meta-reinforcement learning for trajectory design in wireless uav networks," *arXiv preprint arXiv:2005.12394*, 2020.

[7] M. Yan, G. Feng, J. Zhou, Y. Sun, and Y.-C. Liang, "Intelligent resource scheduling for 5g radio access network slicing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7691–7703, 2019.

[8] M. Simsek, A. Czylwik, A. Galindo-Serrano, and L. Giupponi, "Improved decentralized q-learning algorithm for interference reduction in lte-femtocells," in *2011 Wireless Advanced*. IEEE, 2011, pp. 138–143.

[9] G. Sun, D. Ayepah-Mensah, R. Xu, V. K. Agbesi, G. Liu, and W. Jiang, "Transfer learning for autonomous cell activation based on relational reinforcement learning with adaptive reward," *IEEE Systems Journal*, 2021.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] Y. Liu, J. Ding, and X. Liu, "A constrained reinforcement learning based approach for network slicing," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–6.

[12] R. Li, Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.