On using Deep Reinforcement Learning to dynamically derive 5G New Radio TDD pattern

Miloud Bagaa[§], Karim Boutiba[*], Adlen Ksentini[*]
[*] EURECOM, Sophia-Antipolis, France
[§] CSC-IT Center for Science Ltd., Espoo, Finland
Email: miloud.bagaa@csc.fi, {karim.boutiba, adlen.ksentini}@eurecom.fr

***Abstract*—The deployment of 5G and 6G is highly motivated by the emerging network services that demand more bandwidth and very low latency. Besides, these services are shifting from dominant Downlink (DL) Traffic to a more equilibrate DL/UpLink (UL) and dominant UL traffic for specific emerging services. One option to accommodate this new behavior is to use Time Duplex Division (TDD), where the radio frame is shared between UL and DL time slots, namely UL/DL pattern. While 4G TDD has a fixed number of configurations that cannot be updated on runtime, 5GNR allows complete flexibility to define the UL/DL pattern. Therefore, 5G base stations can dynamically change the pattern to adapt to the type of traffic (i.e., UL or DL). However, the 5G standard does not specify algorithms or solutions to derive the UL/DL pattern. To fill this gap, we propose a Deep Reinforcement Learning (DRL) that adds intelligence to the base station to self-adapt to the traffic pattern of the cell type. The proposed DRL algorithm monitors UL and DL buffers at the 5G base station to derive the optimal UL/DL pattern in respect to the current traffic configuration. The proposed solution delivers the optimal configuration in a timely and efficient manner. Simulation results demonstrated the efficiency of the proposed algorithm to avoid buffer overflow and ensure the generality by reacting to traffic pattern changes.**

I. Introduction

Emerging 5G and 6G network services are shifting from Downlink (DL)-dominant traffic to more equilibrate DL and Uplink (UL) traffic, even to more UL-dominant traffic [1]. Indeed, emerging network services such as Augmented and Virtual Reality applications generate high UL traffic corresponding to offloaded intensive computation to be run at a remote application sitting at the Edge. Another example is the high-quality video streaming captured by drones for building surveillance that requires more UL traffic than DL. One solution to accommodate this new trend in terms of traffic model is Time Division Duplex (TDD). TDD allows using the entire bandwidth by dividing it into time slots where some are assigned to UL and some to DL.

Long Term Evaluation (LTE) or 4G proposes 7 different configurations of the TDD frame, which allows configuring the eNB according to the traffic patterns. But, one concern with this solution is that the configuration is fixed in time and cannot be adapted to the traffic dynamic, i.e., if in a certain moment of time more DL or UL need to be accommodated, then there is no possibility to increase the number of UL or DL slots, without rebooting the eNB. To overcome this issue, 5G NR introduces a more flexible solution, where the number of UL and DL slots in the TDD frame can be changed dynamically. This flexibility will allow the gNB to adapt to the frame configuration according to the traffic pattern by selecting the number of slots dedicated to UL and DL. However, the 5G NR specifications only cover the mechanism allowing the gNB to inform the UE about the UL/DL slots pattern in a TDD frame, leaving the algorithm deriving the pattern UL/DL opens.

In this paper, we fill this gap by proposing a novel algorithm, namely Deep Reinforcement Learning (DRL)-based 5G RAN TDD Pattern (DRP), which allows deriving the UL/DL pattern of TDD frames dynamically and accommodating cell traffic whatever it is DL or UL dominant. DRP monitors UL and DL traffic periodically and derives the optimal pattern. DRP uses the Buffer Status Report (BSR) sent by UEs for the UL traffic and the state of the radio bearer channel queues at gNB, which avoids having an exact pattern of the traffic. DRP is run by gNB before the MAC scheduling process. DRP is particularly efficient for 5G private network deployment, allowing to deploy gNB in a plug-and-play mode.

The remaining of the paper is organized as follows. Section II gives the necessary background and related work. Section III introduces the main idea and the problem formulation targeted by this paper. The DRP solution is described in section IV, while section V shows the simulation results and its discussion. Finally, section VI concludes the article.

II. Background

A. 5G NR TDD

5G New Radio (NR) introduces several new features to improve the performance of mobile networks. First, 5G NR uses larger bandwidth (up to 100 MHz in < 6 GHz frequency band, and up to 400 MHz in > 6 GHz frequency band) to accommodate data-rate demanding applications [2]. Second, 5G NR introduces different physical layer numerologies to reshape radio units in time and frequency. Unlike LTE that uses a (Time Transmission Interval) of $1ms$, 5G NR reduces TTI to 2, 4, 8, and 16 times smaller. For the sake of paper readability, the used notations are summarized in Table I. Numerology in 5G, noted $\mu \in 0, 1, 3, 4$, is defined by a Sub-Carrier Spacing (SCS) and a Cyclic Prefix (CP). 5G NR specifies five numerologies, which result in different SCS and slot durations. Indeed, the SCS and slot duration are given as follows: $15 * 2^{\mu}$ and $1/2^{\mu}$, respectively. While LTE uses a fixed time slot duration (i.e., $0.5ms$), 5G NR reduces the slot duration up to 16 times (when $\mu = 4$), which allows decreasing the RAN latency considerably.

It should be noted that each slot in 5G NR is constituted by 14 OFDM symbols, while the subframe and the radio frame length are similar to LTE, i.e., 1ms and 10ms, respectively. Table II summarizes the number of slots in a subframe/frame for each numerology for normal CP.

Like LTE, 5G NR supports both Frequency Division Duplex (FDD) and Time Division Duplex (TDD) operations. However, unlike LTE that specifies seven predefined patterns for

TABLE I: Summary of Notations & Variables.

Notation / Variable	Description
$\mathcal{B}$	the gNB.
Γ	the set of UEs in the network.
γ	A UE $\gamma \in \Gamma$.
δ	TDD period.
μ	A numerology used by gNB.
$\mathcal{T}_\delta$	The number of slots during a period δ. $\mathcal{T}_\delta = \{1, 2, 3 \cdots 16\}$.
$\lambda_\gamma^{\mathcal{U}}$	The uplink traffic generated by UE γ.
$\lambda_\gamma^{\mathcal{D}}$	The downlink traffic of UE γ.
$\lambda_\Gamma^{\mathcal{U}}$	The uplink traffic generated by all UEs Γ. Formally, $\lambda_\Gamma^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \lambda_\gamma^{\mathcal{U}}$
$\lambda_\Gamma^{\mathcal{D}}$	The downlink traffic of all UEs Γ. Formally, $\lambda_\Gamma^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \lambda_\gamma^{\mathcal{D}}$
$\psi_\gamma^{\mathcal{U}}$	The uplink buffer of UE γ.
$\psi_\gamma^{\mathcal{D}}$	The downlink buffer of UE γ.
$\Psi_\Gamma^{\mathcal{U}}$	The uplink buffer of all UEs Γ. Formally, $\Psi_\Gamma^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{U}}$.
$\Psi_\Gamma^{\mathcal{D}}$	The downlink buffer of all UEs Γ. Formally, $\Psi_\Gamma^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \psi_\gamma^{\mathcal{D}}$.
$\mathcal{S}$	OFDM symbol.
$\mu_{\mathcal{S}}$	The amount of traffic in bytes transmitted by $\mathcal{S}$.
α	A constant that specifies the priority between the uplink and downlink traffics.
$\Phi_\gamma^{\mathcal{U}}$	The initial amount of stored data in bytes in the uplink buffer $\psi_\gamma^{\mathcal{U}}$.
$\Phi_\gamma^{\mathcal{D}}$	The initial amount of stored data in bytes in the downlink buffer $\psi_\gamma^{\mathcal{D}}$.
$\mathcal{X}_\gamma$	A real variable that denotes the percentage of uplink slots reserved for the UE $\gamma \in \Gamma$.
$\mathcal{Y}_\gamma$	A real variable that denotes the percentage of downlink slots reserved for the UE $\gamma \in \Gamma$.

UL and DL allocation in a radio frame, 5G NR allows defining UL/DL patterns more flexibly. Indeed, it is possible that a slot may not be configured to be fully used for DL or for UL. OFDM symbols in a slot can be classified as "downlink", "flexible", or "uplink". Flexible symbols can be configured either for UL or for DL transmissions. Finally, like LTE, a guard period is necessary for the transceiver to switch from DL to UL and allow timing advance in UL.

The slot configuration, or DL/UL pattern, is indicated to UE either via Broadcast or RRC configuration message. We distinguish between a common configuration that concerns all the slots marked as DL or UL, and a dedicated configuration that covers all slots and symbols noted as Flexible. The DL/UL pattern is repeated periodically according to $dl-UL-TransmissionPeriodicity$, noted δ. The value of δ depends on the NR numerology (μ). For instance, $\delta = 0.625ms$ can be used only with $\mu \in 3, 4$, while $\delta = 2.5ms$ can be used for all numerology expecting $\mu = 0$. Hence, depending on the numerology and δ, the number of slots (noted $\mathcal{T}_\delta$) varies completely. It is derived as follows: $\mathcal{T}_\delta = \delta \times 2^\mu$. For instance, if $\delta = 2.5ms$ and $\mu = 2$, the number of available slots is equal to 10. If $\delta = 5ms$ and $\mu = 4$, the number of available slots equal to 80.

In addition to δ, the common configuration includes the number of slots for downlink (d_{slots}) located at the beginning of the transmission period and for uplink (u_{slots}) located at the end of the transmission period. d_{sym} symbols within the slot immediately following the last full downlink slot and last u_{sym} symbols in the slot preceding the first full uplink slot are also indicated in the common configuration. The remaining symbols are considered flexible symbols. These flexible symbols can further be allocated to either downlink or uplink by making use of dedicated configuration. Figure 1 illustrates a UL/DL pattern. For more details on TDD pattern management in 5G NR, readers may refer to [3].

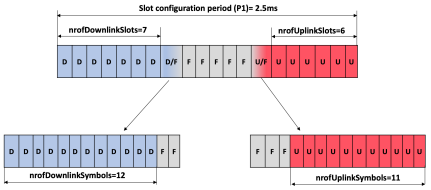


Fig. 1: 5G NR TDD example pattern

B. Related work

Many dynamic TDD resource allocation algorithms for optimizing both power efficiency and transmission performance in LTE were investigated [4]–[6]. T. Ding et al. analyze the performance of employing dynamic TDD for dense small cell networks towards 5G [7]. In [8], authors considered a cluster-based dynamic UL/DL re-configuration considering a centralized-RAN scenario in dense deployments. The central control unit uses a low complexity trellis exploration algorithm to reconfigure the UL/DL ratio aiming at maximizing the overall RAN throughput. In [9], the authors presented a game-theoretic approach where each eNB re-configures its TDD frame aiming at minimizing the UL/DL delay while considering cross-slot interference. As mentioned earlier, all these works consider the LTE TDD configuration, ignoring the new flexibility of 5G NR.

In [10], the authors explored employing deep reinforcement learning to adaptively allocate TDD UL/DL resources in the high mobility 5G HetNet. As high mobility of users leads to the high dynamic network traffic and unpredicted link state change, a new method to predict the dynamic traffic and channel condition and schedule the TDD configuration in real-time is proposed. However, this work does not consider the 5G NR specificities and requires additional information that is not available at the base station. In [11], the authors proposed a service-oriented soft spectrum slicing for 5G TDD. The objective is to use the flexibility of TDD to adjust the UL/DL dynamically using forecasted traffic and user mobility. The problem has been modeled using weighted optimisation. Although the paper tackles 5G, it uses the TDD LTE configuration (i.e., fixed patterns). In addition, the optimization problem needs to be solved periodically to define the new pattern, which is not realistic in a real deployment.

III. Network Model and Problem Formulation

A. Network model

In the system, we focus only on one gNB $\mathcal{B}$, whereby a set of UEs are connected. Let Γ denote the set of UEs in the system. UEs $\gamma \in \Gamma$ are attached to the same gNB. We assume that gNB is using TDD operations. Each TDD period δ is fixed by gNB.Each UE $\gamma \in \Gamma$ has uplink and downlink traffics that can vary from an industrial vertical to another. While uplink traffic is generated and transmitted from the UE γ to the gNB $\mathcal{B}$, the downlink traffic is sent from the gNB $\mathcal{B}$ to the UE γ. Let $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ denote the amount of uplink and downlink traffic in bytes of UE γ, respectively.

In contrast to LTE, where the downlink traffic $\lambda_\gamma^{\mathcal{D}}$ is more critical than the uplink one $\lambda_\gamma^{\mathcal{U}}$, in the $5G$ system and beyond the amount of traffic in uplink and downlink

TABLE II: number of slots in a subframe/frame for each numerology for normal CP

μ	SCS	No. of slots per subframe = 2^{μ}	No. of slots per radio frame = 10*2^{μ}	Slot duration
0	15khz	1	10	1
1	30khz	2	20	0.5
2	60khz	4	40	0.25
3	120khz	8	80	0.125
4	240khz	16	160	0.0625

is application dependent. For instance, in a network slice that manages autonomous driving or unmanned aerial vehicle (UAV), high uplink traffic $\lambda_{\gamma}^{\mathcal{U}}$ is expected. On another side, in some verticals, such as video on demand (VOD) and live streaming, the downlink traffic $\lambda_{\gamma}^{\mathcal{D}}$ is more important. Other industrial verticals and network slices, such as immersive applications (virtual reality, augmented reality, and holograph communication), require high data rates in both directions. The players in these applications are characterized by colossal collaborative interactions, tremendous precision, and high data synchronization. Furthermore, the same UE γ can be subscribed in multiple network slice, which makes hard to predict the uplink $\lambda_{\gamma}^{\mathcal{U}}$ and downlink $\lambda_{\gamma}^{\mathcal{D}}$ traffic of a UE γ.

For the sake of simplicity and without loss of generality, we assume that each UE γ has limited uplink $\Psi_{\gamma}^{\mathcal{U}}$ and downlink $\Psi_{\gamma}^{\mathcal{D}}$ buffers, respectively. Let $|\Psi_{\gamma}^{\mathcal{U}}|$ and $|\Psi_{\gamma}^{\mathcal{D}}|$ denote the size of uplink and downlink buffers in bytes, respectively. While $\Psi_{\gamma}^{\mathcal{U}}$ is located at the UE γ, $\Psi_{\gamma}^{\mathcal{D}}$ is located at the gNB $\mathcal{B}$. The UE γ periodically keeps informing the gNB $\mathcal{B}$ about the state of $\Psi_{\gamma}^{\mathcal{U}}$. In 5G NR, this operation corresponds to the Buffer Size Report (BSR) sent by UE when requesting uplink resources. Meanwhile, the downlink buffers are monitored by gNB, as it corresponds to the radio bearer data channels maintained by gNB for each UE. Let $\Psi_{\Gamma}^{\mathcal{U}}$ and $\Psi_{\Gamma}^{\mathcal{D}}$ denote the uplink and downlink buffer of all the UEs. Formally, $\Psi_{\Gamma}^{\mathcal{U}} = \sum_{\gamma \in \Gamma} \psi_{\gamma}^{\mathcal{U}}$, and $\Psi_{\Gamma}^{\mathcal{D}} = \sum_{\gamma \in \Gamma} \psi_{\gamma}^{\mathcal{D}}$.

B. Problem formulation

In this work, we assume that the number of slots $\mathcal{T}_{\Delta}$ is fixed in each frame Δ. However, their distribution on uplink and downlink traffic is unknown, and it is the target of this paper. The main research question targeted by this paper is how to distribute the slots among the uplink and downlink traffic, such that the service level agreement (SLA) is preserved and the uplink $\Psi_{\Gamma}^{\mathcal{U}}$ and downlink $\Psi_{\Gamma}^{\mathcal{D}}$ buffers do not overrun their boundary. The main challenge faces the paper is both uplink $\lambda_{\gamma}^{\mathcal{U}}$ and downlink $\lambda_{\gamma}^{\mathcal{D}}$ traffics are unknown and hard to predict.

As aforementioned, each slot has 14 OFDM symbol $\mathcal{S}$, each of which can transmit $\mu_{\mathcal{S}}$ bytes. Thus, each slot $t \in \mathcal{T}_{\delta}$ transmits $14 \times \mu_{\mathcal{S}}$. Let $\mathcal{X}_{\gamma}$ a real variable that denotes the percentage of uplink slots reserved for the UE $\gamma \in \Gamma$. Similarly, let $\mathcal{Y}_{\gamma}$ a real variable that denotes the percentage of reserved slots for downlink traffic. Formally, the following statements should hold (2), (3) and (4).

If we denote by $\mathcal{X}_{\Gamma}$ a real variable that denotes the percentage of uplink slots reserved for all the UEs Γ, then $\mathcal{X}_{\Gamma} = \frac{1}{|\Gamma|} \sum_{\gamma} \mathcal{X}_{\gamma}$. Similarly, if $\mathcal{Y}_{\Gamma}$ is a real variable that denotes the percentage of downlink slots reserved for all the UEs Γ, then $\mathcal{Y}_{\Gamma} = \frac{1}{|\Gamma|} \sum_{\gamma} \mathcal{Y}_{\gamma}$. Also, the following statement holds:

$$\mathcal{X}_{\Gamma} + \mathcal{Y}_{\Gamma} = 1$$

Let α a given constant ($0 \leq \alpha \leq 1$) that defines the priority between the uplink and downlink traffics. This parameter can be defined by the customer to specify the priorities between the two traffics. If $\alpha = 1$, then we are interested only to optimize the uplink traffic, whereas, if $\alpha = 0$, then we are interested only to optimize the downlink traffic. The proposed solution should be periodically applied to specify $\mathcal{X}_{\gamma}$ and $\mathcal{Y}_{\gamma}$ in order to prevent the overflow of uplink buffer $\Psi_{\gamma}^{\mathcal{U}}$ and downlink buffer $\Psi_{\gamma}^{\mathcal{D}}$. Let $\Phi_{\gamma}^{\mathcal{U}}$ and $\Phi_{\gamma}^{\mathcal{D}}$ denote the initial stored data of the uplink and downlink buffers. Both $\Phi_{\gamma}^{\mathcal{U}}$ and $\Phi_{\gamma}^{\mathcal{D}}$ are initialized by zero. At each iteration, we aim to optimize the following linear integer programming:

$$\begin{aligned} \textbf{min} \; & \frac{\alpha}{|\Psi_{\Gamma}^{\mathcal{U}}|} \times \sum_{\gamma \in \Gamma} \left(\Phi_{\gamma}^{\mathcal{U}} + \lambda_{\gamma}^{\mathcal{U}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{X}_{\gamma} \times \mathcal{T}_{\delta} \right) \\ & \frac{1-\alpha}{|\Psi_{\Gamma}^{\mathcal{D}}|} \times \sum_{\gamma \in \Gamma} \left(\Phi_{\gamma}^{\mathcal{D}} + \lambda_{\gamma}^{\mathcal{D}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{Y}_{\gamma} \times \mathcal{T}_{\delta} \right) \end{aligned} \tag{1}$$

S.t,

$$0 \leq \mathcal{X}_{\gamma} \leq 1 \tag{2}$$

$$0 \leq \mathcal{Y}_{\gamma} \leq 1 \tag{3}$$

$$\mathcal{X}_{\gamma} + \mathcal{Y}_{\gamma} = 1 \tag{4}$$

$$\forall \gamma \in \Gamma : \Phi_{\gamma}^{\mathcal{U}} + \lambda_{\gamma}^{\mathcal{U}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{X}_{\gamma} \times \mathcal{T}_{\delta} \leq |\Psi_{\gamma}^{\mathcal{U}}| \tag{5}$$

$$\forall \gamma \in \Gamma : \Phi_{\gamma}^{\mathcal{D}} + \lambda_{\gamma}^{\mathcal{D}} - 14 \times \mu_{\mathcal{S}} \times \mathcal{Y}_{\gamma} \times \mathcal{T}_{\delta} \leq |\Psi_{\Gamma}^{\mathcal{D}}| \tag{6}$$

$$\forall \gamma \in \Gamma : \mathcal{X}_{\gamma} \times \mathcal{T}_{\delta} = \mathcal{A}_{\gamma} \tag{7}$$

$$\forall \gamma \in \Gamma : \mathcal{Y}_{\gamma} \times \mathcal{T}_{\delta} = \mathcal{B}_{\gamma} \tag{8}$$

$$\forall \gamma \in \Gamma : (\mathcal{A}_{\gamma}, \mathcal{B}_{\gamma}) \in \mathcal{N}^2 \tag{9}$$

The objective function (1) aims to minimize the amount of stored data in the uplink and downlink buffers to prevent their overflow. While $\lfloor \sum_{\gamma \in \Gamma} \mathcal{X}_{\gamma} \times \mathcal{T}_{\delta} \rfloor$ denotes the number of slots reserved for the uplink traffic, $\lceil \sum_{\gamma \in \Gamma} \mathcal{X}_{\gamma} \times \mathcal{T}_{\delta} \rceil$ is the number of slots reserved for the downlink traffic. We have used weighted normalized sum method to prevent an objective (i.e., buffer) dominant the other. Meanwhile, constraints, (2), (3) and (4), ensure that the variables $\mathcal{X}_{\gamma}$ and $\mathcal{Y}_{\gamma}$ are rates of slots distribution for uplink and downlink. Meanwhile, constraints 5 and 6 ensure that the uplink and downlink buffer of $\gamma \in \Gamma$ is not overflow, respectively. Meanwhile, constraints, (7), (8) and (9), ensure that the number of uplink and downlink solts distributed on each UE $\gamma \in \Gamma$ is an integer. Note that $\mathcal{A}_{\gamma}$ and $\mathcal{B}_{\gamma}$ are two integer variables should be fixed by the system. While $\mathcal{A}_{\gamma}$ denotes the number of uplink slots of γ, $\mathcal{A}_{\gamma}$ denotes the downlink slots.

Unfortunately, we cannot use the optimization problem mentioned above for distributing the slots mainly due to two reasons: i) Solving the optimization problem is time-consuming while we should take decisions within few milliseconds as the solution is online and acting at the RAN level; ii) The amount of uplink $\lambda_{\gamma}^{\mathcal{U}}$ and downlink $\lambda_{\gamma}^{\mathcal{D}}$ traffics are unknown and it is hard to predict them a priory.

IV. DRP: DRL-BASED 5G RAN TDD PATTERN

As aforementioned, it is hard to distribute OFDM slots using optimization efficiently and without prior knowledge of the traffic generation patterns. For this reason, we have proposed the DRP system that leverages DRL, more precisely the Deep Deterministic Policy Gradient (DDPG) Algorithm, to define the 5G NR TDD pattern dynamically. The DRL hides the complexity and stochastic of the environment and helps the DRP framework to make efficient and quick decisions that adapt according to the traffic patterns. Moreover, the DRP framework gains the ability to learn with time and adapts to different and unseen situations. In the balance of this section, we will present the DRP system overview and DRL background, more precisely the DDPG Algorithm, and a detailed description of the DRP system.

A. DRP System Overview and DRL Background

Deep Reinforcement Learning (DRL) will play a crucial role in communication, and networking [12] with the ability to provide a self-configured and self-optimized network that easily adapts to the network changes. Moreover, DRL is a lightweight framework that enables for providing quick decisions, and hence takes real-time actions in the network that is characterized by its dynamicity and needs fast decisions. As depicted in Fig. 2, DRL techniques are based on the interaction of the DRP Agent with its environment by applying different actions and receiving rewards according to the taken actions. The DRP agent interacts with the environment at discrete time steps. At each step t (i.e., interaction), the environment in a state s_t, the agent takes the action a_t and receives the reward r_{t+1}, and hence the environment moves to the next state s_{t+1}. DRL enhances legacy Reinforcement Learning (RL) by considering continuous states. In contrast to Markov Decision Processes (MDPs), DRL is a model-free approach and does not require transition probabilities between states.

The ability of DRL-based solutions to deal with unknown and unseen environments makes them the best fit for addressing the 5G TDD pattern configuration problem. The DRL Algorithm mainly consists of two steps: i) The learning (exploration) step; ii) The exploitation step. In order to overcome the need of model (i.e, transition probability), during the learning step, the agent interacts with the environment by following stochastic policy (i.e., random actions) to explore and build the knowledge about the environment. While, in the exploitation step, the agent exploits the acquired knowledge by following the optimal policy π_* that provides the optimal action a_t, to take in each state s_t, in a way that maximizes future cumulative discounted reward G_t defined as follows:

$$G_t \doteq \sum_{k=0}^{T} \gamma^k r_{t+k+1} = r_{t+1} + \gamma G_{t+1} \tag{10}$$

With $\gamma \in [0, 1]$ defined as the discount rate that penalises the future rewards, and T equal to the time horizon which is finite for episodic problems (i.e., problems that ends when the environment is a final state) and infinite for continuing problems.

In general, DRL methods are classified into three categories : i) *value-based* methods, such as Monte Carlo, SARSA, Q-Learning, and DQN; ii) *policy-based* methods, such as REINFORCE (i.e., Monte-Carlo Policy Gradient) and REINFORCE with baseline; iii) *actor-critic* methods that combine the two previous methods, such as A2C, A3C, DDPG, and PPO [13]. In actor-critic approach, we have mainly two families, the stochastic policy approach (e.g., A2C and A3C) and the deterministic approach (e.g., DDPG). In the stochastic policy approach, the actions are selected from the Actor with different probability using the softmax activation function. The agent should pick the action that has a high probability. Unfortunately, the main limitation of the stochastic policy approach is the number of actions should be limited. In contrast, in the deterministic approach, the actions are generated directly from the actor-network, enabling continuous actions. In this paper, we are interested in specifying the percentage of uplink and downlink slots as depicted in Fig. 2. For this reason, we have adapted DDPG Algorithm. We will explain further the DDPG Algorithm when explaining our DRP approach.

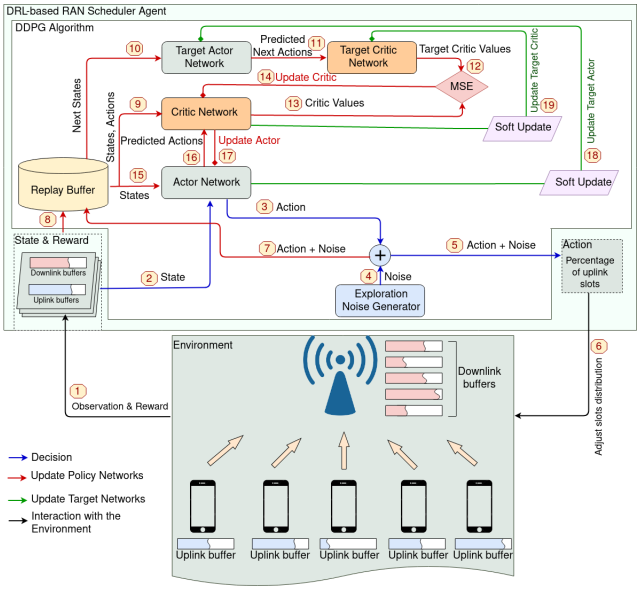


Fig. 2: Envisioned DRL-based 5G RAN TDD Pattern (DRP) System

B. DRP Detailed Description

We have designed the DRP agent to be lightweight to ensure fast interaction with the environment. Also, we have designed the DRP agent to ensure generality and then work in an unseen environment. The DRP agent has been designed to work independently from the number of slots and the size of the buffers. Moreover, it considers the variation and correlation in the buffer states to predict the traffic patterns. In what follows, we define the elements of the DRP agent, including the state, the reward, and the action.

i) State: Let $\xi_t^{\mathcal{U}}$ and $\xi_t^{\mathcal{D}}$ denote the amount of traffic in the uplink $\Psi_\Gamma^{\mathcal{U}}$ and the downlink $\Psi_\Gamma^{\mathcal{D}}$ at the step t, respectively. To ensure the generalization, we define the observation $\mathcal{O}_t^{\mathcal{U}}$ and $\mathcal{O}_t^{\mathcal{D}}$ of the uplink and downlink buffers as normalized values (Fig. 2: 1). Formally, $\mathcal{O}_t^{\mathcal{U}} = \frac{\xi_t^{\mathcal{U}}}{\Psi_\Gamma^{\mathcal{U}}}$ and $\mathcal{O}_t^{\mathcal{D}} = \frac{\xi_t^{\mathcal{D}}}{\Psi_\Gamma^{\mathcal{D}}}$, respectively. The benefits of the normalization are twofold: i) It ensures the generality by enabling the DRP agent to be agnostic to the scenario scale. It works similarly in different buffers with different sizes. The most important is to catch the buffer fullness ratio of $\Psi_\Gamma^{\mathcal{U}}$ and $\Psi_\Gamma^{\mathcal{D}}$; ii) It is well known that the activation functions in the neural network work well for small values, which positively impacts DRP's convergence. Moreover, to capture the traffic patterns, we define the state s_t as follow:

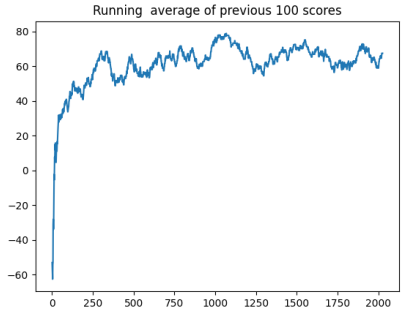


Fig. 3: Convergence evaluation of DRP agent during the training mode

$$s_t = \Big(\bigcup_{i=0}^{\mathcal{K}} \mathcal{O}_{t-i}^{\mathcal{U}}, \bigcup_{i=0}^{\mathcal{K}} \mathcal{O}_{t-i}^{\mathcal{D}} \Big) \tag{11}$$

In the state s_t, the DRP agent, besides the current observation, considers $\mathcal{K}$ previous observations before taking any action. This enables to capture the behavior of both buffers and traffics before making any action.

ii) Action: We have only one continuous action a_t that presents the percentage of slots should be reserved for the uplink traffic at the step t. The DRP agent enforces the taken decisions as depicted in Fig. 2:6. Accordingly, the $\lfloor a_t \times \mathcal{T}_\delta \rfloor$ slots are reserved for the uplink traffic, and $\lceil (1-a_t) \times \mathcal{T}_\delta \rceil$ slots are reserved for the downlink traffic. It is worth noting that the action is independent on the number of slots $\mathcal{T}_\delta$, which ensures the generality and enables the DRP agent to be agnostic to the scenario scale.

iii) Reward: We have adapted an episodic approach, whereby each episode runs for max T steps before ends. The reward r_t has been defined as follow:

$$r_t = \begin{cases} \alpha \times (1 - \mathcal{O}_t^{\mathcal{U}}) + (1-\alpha) \times (1 - \mathcal{O}_t^{\mathcal{D}}) & \text{If} |\mathcal{O}_t^{\mathcal{U}}| < 1 \wedge |\mathcal{O}_t^{\mathcal{D}}| < 1 \\ -\mathcal{M} & \text{Otherwise} \end{cases} \tag{12}$$

Where, α is the priority between the uplink and downlink traffics. The DRP agent receives a positive reward for each step succeeds to keep the buffers $\Psi_\Gamma^{\mathcal{U}}$ and $\Psi_\Gamma^{\mathcal{D}}$ do not exceed their threshold. Moreover, the emptiest the buffers are, the highest reward becomes. When one of the buffers exceeds its capacity, then the agent receives a penalty $-\mathcal{M}$, such that $\mathcal{M}$ is a significant number. This strategy will enforce the DRP agent to keep both buffers empty as much as possible and prevent their overflow, which has a positive impact on the QoS.

As depicted in Fig. 2, DRP system leverages DDPG algorithm and executed on three different steps: i) Decision making (Fig. 2: $1-6$) presented with blue color; ii) Updating policy networks (Fig. 2: $7-17$) presented with red color; iii) Updating target networks (Fig. 2: $18-19$) presented by green color. In DDPG, we have two networks: a) Policy networks that consist of the Actor and the Critic neural networks. These networks are used to predict the deterministic action a_t. While the actor network has as input the state s_t and it is used to predict the action a_t, the critic network has as inputs s_t and a_t and returns the Q value that is used for criticizing the taken action; b) The target network that consists of target actor and target critics. These two networks are frozen and used to help the convergence of policy networks and stabilize their learning. In deep learning, the optimizer (i.e., ADAM) should update the neural network parameters of the policy networks closer to the labels, which are the fixed target neural network values. Moreover, to stabilize the learning, a replay buffer is used. The training is performed using a random sample from the replay buffer, which reduces the correlation between the agent's experiences.

Decision making: At the reception of the observation $(\mathcal{O}_t^{\mathcal{U}}, \mathcal{O}_t^{\mathcal{D}})$, the DRP agent generates the state s_t using the equation (11) (Fig. 2: 2). The received state is used by the actor network to predict the deterministic action a_t (11) (Fig. 2: 3). In order to enable the DRP agent to explore the environment, a noise $\mathcal{N}$ is added to the action (11) (Fig. 2: 4). Accordingly, the uplink and downlink slots are reserved (Fig. 2: $5-6$).

Updating policy network: In order to take optimal actions, the policy network should be updated (Fig. 2: $7-17$). The taken action with noise should be stored in the replay buffer (Fig. 2: 7), as well as their corresponding state and reward (Fig. 2: 8). First the critic network is updated by leveraging a random batch sample $(s_t, a_t + \mathcal{N}, r_t)$ from the replay buffer (Fig. 2: $9-14$). Using mean square error (MSE) and ADAM optimizer, the parameters of the critic network are optimized by considering the critic values and target critic values. Then, the actor network is also optimized by leveraging the gradient generated against the critic network.

Updating policy network: The target networks (actor and critic) should be updated slowly and periodically towards the policy networks using soft update (Fig. 2: $18-19$). This strategy helps the Algorithm for providing optimal deterministic action.

V. Performance Evaluation

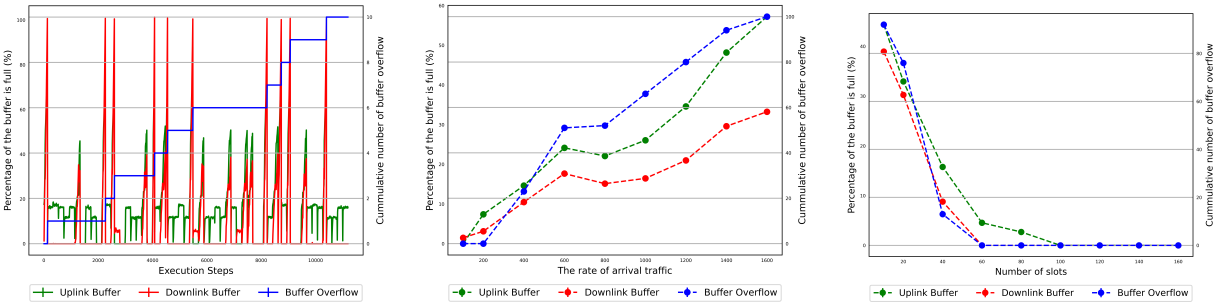
We have implemented our simulation environment using Python, and Pytorch . We have used a physical machine with 8 Core i7-8665U 1.90GHz and 16 GB of memory using Centos 7 as an operating system. We have employed two fully connected hidden layers of 400 and 300 nodes for both policy and target networks. We have also used layer normalization between the hidden layers to enable smoother gradients, faster training, and better generalization accuracy. While Rectified Linear Unit (ReLU) activation function has been used in the two hidden layers, Hyperbolic Tangent (*tanh*) activation function has been used in the output layer. We have employed a discount factor γ of 0.99, batch size of 64, and the learning rates of the actor and critic networks are set to 10^{-5} and 10^{-3}, respectively. We have used the soft update with coefficient τ 0.001. Also, ADAM optimizer has been leveraged in both actor and critic networks.

A. DRP Training mode

As depicted in Fig. 3, we have trained our DRP agent using 2000 independent episodes. We have fixed the maximum number of steps at each episode T by 100. We have set the penalty reward $\mathcal{M}$ to -100, and the number of slots $\mathcal{T}_\delta$ to 40. We have considered three successive observations (i.e., $\mathcal{K} = 3$). From the figure, we observe that the DRP agent converges at 500 episodes.

B. DRP Inference mode

We have evaluated the DRP agent in terms of: i) The percentage of the fullness of the uplink and downlink buffers; ii) The number of buffer overflows. In the figure 4, the left Y-axis represents the percentage of the fullness of the uplink



(a) Impact of time (b) Impact of arrival traffic rate λ (c) Impact of number of slots $\mathcal{T}_\delta$

Fig. 4: Performance evaluation of DRP system during the inference mode

and downlink buffers, whereas the right Y-axis represents the number of overflows. In all the experiments, we have fixed the maximum number of steps T by 200. We have conducted three sets of experiments: i) We have varied the number of episodes while fixing the number of slots $\mathcal{T}_\delta$ by 40 and the amount of uplink $\lambda_\gamma^{\mathcal{U}}$ and downlink $\lambda_\gamma^{\mathcal{D}}$ traffic by 400; ii) We have varied $\mathcal{T}_\delta$ while fixing $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ by 400 and number of episodes by 100; iii) We have varied $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ while fixing $\mathcal{T}_\delta$ by 40 and number of episodes by 100.

Fig. 4(a) depicts the impact of the execution steps on the DRP agent. The first observation that we can draw from this figure is the average amount of traffics in the uplink, and downlink buffers are 20%. Also, we observe that the fullness of the buffer did not exceed 45% in the 12000 execution steps. We also witness that the number of times a buffer overflow happens did not exceed 10 times, which is less than 0.083%. This figure also confirms the ability of the DRP agent for generalizing. While it is trained with $T = 100$, it has been tested with $T = 200$ without feeling any difference.

Fig. 4(b) depicts the impact of $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ on the DRP agent. Each plotted point presents the average of 100 episodes, each of which with 200 steps. The first observation that we can draw from this figure is the amount of traffic hurts the DRP agent. The more data traffic generation rate is, the higher likelihood to get buffer overflow becomes. We observe that whatever the amount of traffic, the average fullness of both buffers does not exceed 60%. The number of buffer overflows did not exceed 100 cases, which is 0.5% in the worst-case scenarios. The obtained results demonstrate the DRP agent's ability to make the generality and perform well in unseen environments.

Fig. 4(c) shows the impact of the number of slots on the DRP agent. We observe that the number of slots has a positive impact on the fullness of the buffers and the number of buffer overflows. The more number of slots is, the higher capacity for transmitting the packets becomes. We observe that DRP agent succeeded to get empty buffers when more 60 slots are used. These results demonstrate the efficiency of the proposed solution for making the generality and achieving its main design goals. We also observe that the percentage of buffer fullness does not exceed 45% in the worst-case scenarios. Also, the number of buffer overflows did not exceed 90, which is less than 0.45%.

VI. Conclusion

In this paper, we introduced DRP, a Deep Learning Reinforcement (DRL)-based solution that allows to derive and adjust the TDD pattern in 5G NR. DRP is used by 5G base station before the UE scheduling process to compute the number of slots dedicated to UL and DL in a TDD frame. Without knowing the UEs traffic model, DRP can derive the optimal TDD pattern aiming at reducing both UL and DL buffers. DRP uses available information at the gNB, i.e., BSR, and DL buffer size, to deduce the optimal TDD pattern. Simulation results clearly showed that DRP could avoid buffer overflow and dynamically adapt to the cell traffic. Our future focus on implementing DRP on top of OpenAirInterface (OAI) 5G to demonstrate self-adapted and plug-and-play deployment of 5G base station.

References

[1] A. Amokrane, A. Ksentini, Y. H. Aoul, and T. Taleb, "Congestion control for machine type communications," in *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*. IEEE, 2012, pp. 778–782.

[2] S. Si-Mohammed, A. Ksentini, M. Bouaziz, Y. Challal, and A. Balla, "UAV mission optimization in 5g: On reducing MEC service relocation," in *IEEE Global Communications Conference, GLOBECOM 2020, Virtual Event, Taiwan, December 7-11, 2020*. IEEE, 2020, pp. 1–6.

[3] *5G; NR; Radio Resource Control (RRC); Protocol specification (3GPP TS 38.331 version 15.3.0 Release 15)*, Std., October 2018.

[4] M. Ding, D. López-Pérez, A. V. Vasilakos, and W. Chen, "Dynamic TDD transmissions in homogeneous small cell networks," in *IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014, Workshops Proceedings*. IEEE, 2014, pp. 616–621.

[5] Z. Shen, A. Khoryaev, E. Eriksson, and X. Pan, "Dynamic uplink-downlink configuration and interference management in TD-LTE," *IEEE Commun. Mag.*, vol. 50, no. 11, pp. 51–59, 2012. [Online]. Available: https://doi.org/10.1109/MCOM.2012.6353682

[6] A. Khoryaev, A. Chervyakov, M. Shilov, S. Panteleev, and A. Lomayev, "Performance analysis of dynamic adjustment of TDD uplink-downlink configurations in outdoor picocell LTE networks," in *4th International Congress on Ultra Modern Telecommunications and Control Systems, ICUMT 2012, St. Petersburg, Russia, October 3-5, 2012*. IEEE, 2012, pp. 914–921.

[7] T. Ding, M. Ding, G. Mao, Z. Lin, A. Y. Zomaya, and D. López-Pérez, "Performance analysis of dense small cell networks with dynamic TDD," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9816–9830, 2018.

[8] D. Zhu and M. Lei, "Cluster-based dynamic DL/UL reconfiguration method in centralized RAN TDD with trellis exploration algorithm," in *2013 IEEE Wireless Communications and Networking Conference (WCNC), Shanghai, Shanghai, China, April 7-10, 2013*. IEEE, 2013, pp. 3758–3763.

[9] M. S. ElBamby, M. Bennis, W. Saad, and M. Latva-aho, "Dynamic uplink-downlink optimization in tdd-based small cell networks," in *11th International Symposium on Wireless Communications Systems, ISWCS 2014, Barcelona, Spain, August 26-29, 2014*. IEEE, 2014, pp. 939–944.

[10] F. Tang, Y. Zhou, and N. Kato, "Deep reinforcement learning for dynamic uplink/downlink resource allocation in high mobility 5g hetnet," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 12, pp. 2773–2782, 2020.

[11] R. Shrivastava, K. Samdanis, and V. Sciancalepore, "Towards service-oriented soft spectrum slicing for 5g TDD networks," *J. Netw. Comput. Appl.*, vol. 137, pp. 78–90, 2019.

[12] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.