Multi-Agent Deep Reinforcement Learning to enable dynamic TDD in a multi-cell environment

Karim Boutiba*, Miloud Bagaa§, and Adlen Ksentini*
* EURECOM, France. firstname.lastname@eurecom.fr
§Department of Electrical and Computer Engineering, Université du Québec à Trois-Rivières, Trois-Rivières, QC, Canada. Email: miloud.bagaa@uqtr.ca

***Abstract*—Dynamic Time Division Duplex (D-TDD) is a promising solution to address newly emerging 5G and 6G services characterized by asymmetric and dynamic uplink (UL) and downlink (DL) traffic demands. However, there are two major issues: (i) determining the TDD scheme (i.e., the number of slots devoted to UL and DL) to meet the dynamic traffic demands of the Users Equipment (UE); (ii) cross-link interference between cells that use different TDD schemes. The 3GPP standard neither specifies algorithms or solutions to derive the TDD configuration nor solves the cross-link interference. To fill this gap, we model the dynamic TDD problem in 5G NR as a linear programming problem. Then, we design Multi-Agent Deep Reinforcement Learning based 5G RAN TDD Pattern (MADRP), a fully decentralized solution based on the Multi-Agent Deep Reinforcement Learning (MADRL) approach. Based on the simulation results, the algorithm effectively prevents buffer overflows, avoids cross-link interference, and adapts to changes in the traffic pattern, ensuring its versatility. We compared our solution with the optimal solution and different static TDD configurations. We found that MADRP outperforms the static TDD configurations. We finally discuss the algorithm's limitations in terms of the number of cells, traffic variance, and cross-link interference probability.**

***Index Terms*—Dynamic TDD, Multi-agent Deep Reinforcement Learning, 5G NR.**

I. Introduction

5G networks and beyond are designed to support an extensive range of applications [1][2], including those requiring high-speed data transfer and low-latency communication, such as immersive holographic communication, Internet of Skills, and 4D Interactive mapping [3]. Compared to previous generations of mobile networks, these emerging applications generate high UpLink (UL) traffic, corresponding to offloaded intensive computations that must be executed by a remote application located at the network's edge. As a result, emerging services in 5G and 6G networks may be DownLink (DL) dominant, UL dominant, or balanced between UL and DL. As a result, dynamic TDD has become a key enabler for 5G and beyond, as it allows resources to be allocated to these applications as needed, ensuring optimal performance and quality of service (QoS). Dynamic TDD allows the base station (gNB) to change the TDD scheme dynamically without interrupting user connectivity, i.e., by changing the number of dedicated UL and DL slots based on the users' traffic patterns. Thus, Dynamic TDD is able to: (i) Improve resource utilization efficiency, e.g., when traffic is dominant on the UL or DL, allocating more slots on the UL or DL will avoid wasting resources in the other non-dominant direction; (ii) Reduce the latency since dynamic TDD reduces the queue buffer size faster than static TDD [4]; (iii) Increase application throughput as more slots can be allocated to UL or DL according to various traffic patterns. However, the 5G NR specifications only cover the mechanism allowing the gNB to inform the User Equipment (UE) about the UL/DL slots pattern in a TDD frame, leaving the algorithm deriving the pattern UL/DL open. In [5], we have filled 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 according to the existing cell traffic whatever it is DL or UL dominant. Besides, we proposed an implementation on the top of OpenAirInterance[6] in [4]. However, we considered dynamic TDD for private 5G deployment where a single cell is considered; in a multi-cell environment, dynamic TDD is more challenging. Indeed, whereas in a single-cell environment, the only challenge was to find the UL/DL ratio without knowing the traffic pattern, in a multi-cell environment, the challenge is twofold: (i) Finding the UL/DL ratio without knowing the traffic pattern; (ii) Mitigating cross-link interference. The latter is defined as interference that occurs when one gNB transmits while another receives in the same frequency band (i.e., two neighboring gNBs that use a different TDD pattern). This usually occurs within the same operator using the same frequency band for its gNBs.

In this paper, we extend DRP to solve the dynamic TDD problem in a multi-cell environment and introduce the Multi-Agent Deep Reinforcement Learning (DRL)-based 5G RAN TDD Pattern (MADRP) framework. The MADRP approach is fully decentralized. Each MADRP agent is located close to the gNB, serving a particular cell. Compared to a centralized approach, MADRP is executed close to the gNB, which reduces the control latency between the gNB and MADRP. Moreover, MADRP reduces the signaling overhead normally generated when gNBs send data to a central entity. Each MADRP agent monitors the gNB's UL and DL buffers and the number of edge users with neighboring cells. Note that edge users of a cell correspond to UEs attached to that cell and are physically located at the cell's boundaries where the signal strength from neighboring cells is significant. Then, each agent uses this local observation along with messages from neighboring cells to derive the optimal TDD pattern to accommodate connected users while avoiding cross-link interference with neighboring cells.

The main contributions of the paper are summarized as follows:

- Model the dynamic TDD problem in 5G NR considering a multi-cell environment. The proposed model takes into account dynamic traffic patterns and cross-link interference.
- Design a solution to solving the problem without knowing the traffic pattern. The suggested approach is designed as a fully distributed solution, enabling it to operate near the gNB and minimize control latency consequently.
- The proposed solution is designed to guarantee generality in terms of (i) number of cells (ii) radio configuration in terms of the numerology, the TDD period, and the buffer capacity.
- The proposed solution takes into account the propagation effect of cross-link interference between cells, i.e., the indirect cross-link interference that occurs between two cells without direct cross-link interference due to a third cell interfering with the two cells.
- Implementation of the proposed solution using the MADDPG algorithm [7].
- Simulation results with different numbers of cells under different traffic patterns and cross-link interference probability. We compared MADRP with the optimal solution and different static TDD configurations with different traffic UL/DL proportions. The results show that MADRP outperforms the static TDD configurations even in high cross-link interference scenario and the gap between the optimal solution and the MADRP solution is small in low-interference scenario.

The remainder 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 this paper targets. The MADRP 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 [8]. Second, 5G NR introduces different physical layer numerologies to reshape radio units in time and frequency. Unlike LTE, which 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 NR, noted $\nu \in 0, 1, 3, 4$, 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. The latter corresponds to the time duration of 14 OFDM symbols. An OFDM symbol duration reduces with increased SCS, hence reducing the time duration of a slot. Indeed, the SCS and slot duration are given as follows: $15 * 2^{\nu}$ and $1/2^{\nu}$, respectively. While LTE uses a fixed time slot duration (i.e., $0.5ms$), 5G NR reduces the slot duration up to 16 times (when $\nu = 4$), which allows decreasing the RAN latency considerably. Since the frame duration is fixed (i.e., 10ms), the number of slots in a frame depends on the numerology (i.e., $10 \times 2^{\nu}$).

Table I: Summary of Notations & Variables.

Notation / Variable	Description
$\mathcal{C}$	set of radio cells
Γ_c	the set of UEs connected to cell c.
γ	A UE $\gamma \in \Gamma_c$.
δ	TDD period.
ν_c	A numerology used by a cell c.
$\mathcal{T}_{\delta}^{c}$	The number of slots in cell c during a period δ . $\mathcal{T}_{\delta}^{c} = \{2, \cdots, 16\}$.
$\lambda_{\gamma}^{\mathcal{U}}$	The UL traffic generated by $\gamma \in \Gamma_c$.
$\lambda_{\gamma}^{\mathcal{D}}$	The DL traffic from a cell towards the UE $\gamma \in \Gamma_c$.
$\psi_{\gamma}^{\mathcal{U}}$	The UL buffer of the UE $\gamma \in \Gamma_c$.
$\psi_{\gamma}^{\mathcal{D}}$	The DL buffer of the UE $\gamma \in \Gamma_c$.
$\lambda_{c}^{\mathcal{U}}$	The UL traffic generated by Γ_c.
$\lambda_{c}^{\mathcal{D}}$	The DL traffic from cell c towards the UEs $\gamma \in \Gamma_c$.
$\Psi_{c}^{\mathcal{U}}$	The UL buffer of cell c.
$\Psi_{c}^{\mathcal{D}}$	The DL buffer of cell c.
$\mu_{c}^{\mathbf{D}}$	The amount of traffic in bytes transmitted by cell c per slot.
$\mu_{c}^{\mathbf{U}}$	The amount of traffic in bytes received by cell c per slot.
α	A constant that specifies the priority between the UL and DL traffics.
$\Phi_{c}^{\mathcal{U}}$	The initial amount of stored data in bytes in the UL buffer $\psi_{c}^{\mathcal{U}}$.
$\Phi_{c}^{\mathcal{D}}$	The initial amount of stored data in bytes in the DL buffer $\psi_{c}^{\mathcal{D}}$.
$\mathcal{F}_{c_1,c_2}$	A Boolean constant that denotes if there is interference between cells c_1 and c_2.
$\mathcal{X}_c$	A real variable that denotes the percentage of allocated UL slots in cell c.
$\mathcal{Y}_c$	A real variable that denotes the percentage of allocated DL slots in cell c.

Like LTE, 5G NR supports Frequency Division Duplex (FDD) and Time Division Duplex (TDD) operations. However, unlike LTE, which specifies seven predefined UL and DL allocation patterns 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 to allow timing advance in UL.

The slot configuration, or DL/UL pattern, is indicated to UE either via Broadcast or Radio Resource Control (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 (ν). In addition to δ, the common configuration includes the number of slots for DL (d_{slots}) located at the beginning of the TDD period and for UL (u_{slots}) located at the end of the TDD period. d_{sym} symbols within the slot immediately following the last full DL slot and the last u_{sym} symbols in the slot preceding the first

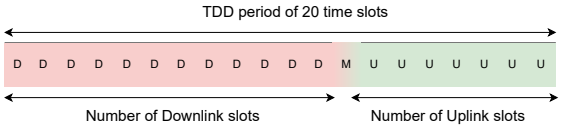


Figure 1: 5G NR TDD example pattern

full UL slot are also indicated in the common configuration. The remaining symbols are not used to give time to the device to switch from DL to UL. These flexible symbols can further be allocated to either DL or UL by using a dedicated configuration. Figure 1 illustrates a UL/DL pattern. For more details on TDD pattern management in 5G NR, readers may refer to [9]. Compared to LTE, a more flexible frame structure can provide a greater traffic adaptation gain but also lead to more dynamic cross-link interference. Cross-link interference occurs when one gNB transmits while another receives in the same frequency band (i.e., two neighboring gNBs using a different TDD pattern). As shown in Figure 2, two types of cross-link interference are introduced: gNB-to-gNB interference and UE-to-UE interference, which can significantly degrade system performance and decrease user throughput.

The UE-to-UE interference impacts edge users, while gNBs are affected by the gNB-to-gNB interference when they are neighbors with high transmission power gNBs. Generally, gNBs use high transmission power to accommodate edge users. Thus, edge users are considered a main factor for cross-link interference. As a result, we assume that neighboring cells may not experience cross-link interference if there are no edge users.

B. Related Works

The problem studied in this paper can be decomposed into two sub-problems: (i) The distribution of time slots between the UL and the DL (ii) the mitigation of the cross-link interference between neighboring cells. In the literature, most solutions focus on solving only one of the two sub-problems.

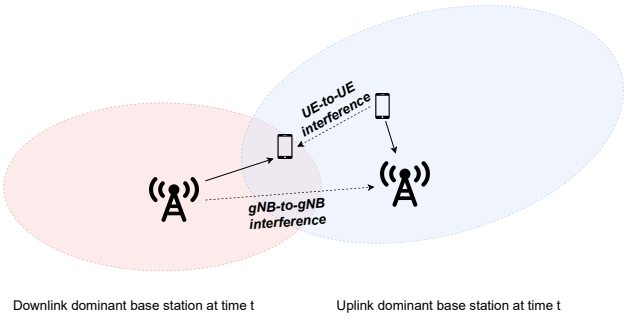


Figure 2: Illustration of cross-link interference in dynamic TDD system

1) Time slots distribution: In [10], the authors explored employing deep reinforcement learning to adaptively allocate TDD UL/DL resources in 5G networks considering high mobility UEs. However, this work requires additional information that is not available at the gNB, such as the ideal channel capacity and real-time channel capacity. 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 a weighted optimization whose objective is to maximize the allocated slice bandwidth for each corresponding load. The weights describe the normalized slice size suitable for each service. Although the paper addresses 5G, it uses the LTE TDD configuration (i.e., fixed TDD patterns) and the proposed solution requires predicting the traffic demands for each service, which can lead to over-allocation or under-allocation of resources when the predicted traffic is not accurate.

2) Cross-link interference mitigation: The authors of [12] investigated interference management schemes for 5G dynamic TDD and proposed new interference suppression schemes using advanced receivers. They derived the analytic expressions of the receivers for dynamic TDD interference suppression by theoretical analysis. In [13], the authors designed a distributed algorithm to be used by all transmitters to compute their power allocations in real time and thus avoid gNB-to-gNB interference. However, the cost of these solutions is high because they require updating the transmitters and receivers at all the gNBs and UEs. In [14], the authors overview the academic research and standardization efforts undertaken to solve this cross-link interference problem and make the D-TDD system a reality. However, they did not provide any solution.

3) Joint time slots distribution and cross-link interference mitigation: In [15], the authors proposed a Q-learning approach to reconfigure the TDD pattern of gNBs in order to maximize users' Quality of Experience (QoE) while mitigating cross-link interference. However, they relied on a centralized architecture that does not guarantee generalization, i.e., if they add/remove another gNB, they have to re-train the model from scratch. In addition, they only considered fixed configuration patterns, ignoring the flexibility of 5G NR. In [16], authors proposed a dual reinforcement machine learning approach for online pattern optimization in 5G new radio TDD deployments. However, they considered a centralized approach with the limitations mentioned above. Also, they require knowledge of the UL latency, which is impossible to obtain in real 5G deployments. In [17], the authors proposed a dynamic resource allocation scheme in TDD. They have designed a clustering algorithm to group the radio units into different sets. Then, they adopted coordinated multipoint technology to eliminate interference in each set. However, they used a centralized approach and considered fixed configuration patterns without taking advantage of the flexibility of 5G NR. In [18], authors modeled the D-TDD configuration problem as a dynamic programming problem. Then, they designed a fully decentralized solution with distributed MARL technology. Each agent in MARL makes decisions only based on local observations. However, their models lack generality because they assume a fixed number of TDD patterns with a fixed number of slots, so a change in numerology involves retraining all MADRL agents. In addition, they only considered 4 types of traffic, i.e., high UL, high DL, low UL, and low DL, which implies the need to know the traffic type and pattern in advance.

Overall, current solutions rely on a centralized approach that increases control latency and signaling overhead. Furthermore, they do not consider the flexibility of 5G NR in terms of numerology and dynamic number of slots per TDD period which results in a lack of generality and the necessity to retrain the models for each numerology. Furthermore, they ignore the propagation effect of cross-link interference, i.e., indirect cross-link interference that occurs between two cells with no direct interference due to the transitivity relation between cells. Figure 3 depicts three cells with a direct cross-link interference between cell 1 and cell 3, cell 2 and cell 1. By transitivity, cell 2 and cell 3 are experiencing indirect cross-link interference. As a result, the three cells must align their TDD period even without direct cross-link interference between cell 2 and cell 3.

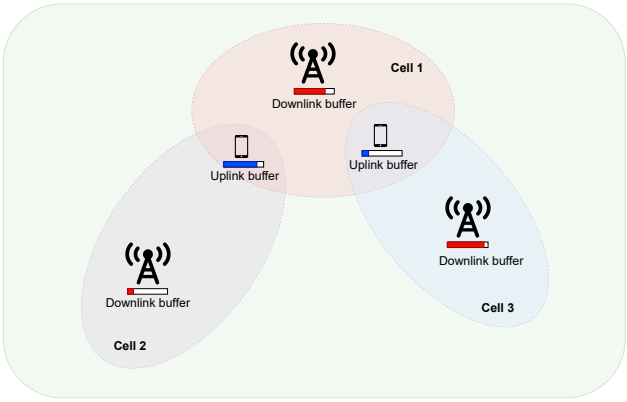


Figure 3: Multi-cell scenario with 3 Radio Units (RU) and 2 User Equipment(UE) with different traffic characteristics experiencing indirect interference

III. Network model and problem formulation

A. Network model

In the system, we consider a set of Radio Access Network (RAN) cells, denoted $\mathcal{C}$, whereby a set of UEs, denoted Γ_c, are connected to each cell $c \in \mathcal{C}$. We assume that cells are covered by radio units operating under TDD and using a fixed TDD period δ and the same numerology. Each UE $\gamma \in \Gamma_c$ has UL and DL traffic that can vary from one application to another. The UL traffic is generated and transmitted from the UE γ to the cell c radio unit, while the DL traffic is sent from the radio unit of cell c to the UE γ. Let $\lambda_\gamma^{\mathcal{U}}$ and $\lambda_\gamma^{\mathcal{D}}$ denote the amount of UL and DL traffic in bytes of UE γ, respectively. In contrast to LTE, where the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more critical than the UL traffic $\lambda_\gamma^{\mathcal{U}}$, in $5G$ networks and beyond the amount of traffic in UL and DL is application dependent. For instance, in AR applications that offload the AR processing to the edge, high UL traffic $\lambda_\gamma^{\mathcal{U}}$ is expected. On another side, in some applications, such as video streaming, the DL traffic $\lambda_\gamma^{\mathcal{D}}$ is more important. Other applications, such as immersive applications (e.g., the Metaverse), require high data rates in both directions. The users in new generation applications are characterized by colossal collaborative interactions, tremendous precision, and high data synchronization. Furthermore, the same UE γ can use multiple applications, which makes it hard to predict the UL $\lambda_\gamma^{\mathcal{U}}$ and DL $\lambda_\gamma^{\mathcal{D}}$ traffic of a UE γ. Let $\lambda_c^{\mathcal{U}}$ and $\lambda_c^{\mathcal{D}}$ denote the traffic of the cell c in UL and DL, respectively. Formally, $\lambda_c^{\mathcal{U}} = \sum_{\gamma \in \Gamma_c} \lambda_\gamma^{\mathcal{U}}$, and $\lambda_c^{\mathcal{D}} = \sum_{\gamma \in \Gamma_c} \lambda_\gamma^{\mathcal{D}}$.

Let $\psi_\gamma^{\mathcal{U}}$ and $\psi_\gamma^{\mathcal{D}}$ denote the UL and DL buffer size of UE γ, respectively. While $\psi_\gamma^{\mathcal{U}}$ is located at the UE γ, $\psi_\gamma^{\mathcal{D}}$ is located at the gNB connected to the radio unit serving cell $c \in \mathcal{C}$. The UE γ periodically keeps informing the gNB 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 UL resources. Meanwhile, the DL buffers are monitored by gNB, corresponding to the radio bearer data channels maintained by gNB for each UE. Let $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$ denote the UL and DL buffer of all the UEs connected to cell $c \in \mathcal{C}$. Formally, $\Psi_c^{\mathcal{U}} = \sum_{\gamma \in \Gamma_c} \psi_\gamma^{\mathcal{U}}$, and $\Psi_c^{\mathcal{D}} = \sum_{\gamma \in \Gamma_c} \psi_\gamma^{\mathcal{D}}$. For the sake of simplicity and without loss of generality, we assume that each cell c has a maximum UL $\Psi_c^{\mathcal{U}}$ and DL $\Psi_c^{\mathcal{D}}$ buffers size, respectively. Let $|\Psi_c^{\mathcal{U}}|$ and $|\Psi_c^{\mathcal{D}}|$ denote the maximum size of UL and DL buffers in bytes, respectively.

The UEs at the edge of the cells experience cross-link interference when two or more neighbor cells are using the same frequency and different slot directions (i.e., UL and DL) at a given time. Figure 4 illustrates a scenario of 3 neighboring cells and 7 UEs with different traffic patterns. Each UE is connected to a cell (the connection is illustrated with a pointed line). The oval shape around each radio unit represents the cell coverage. In this example, the intersection between cells' coverage represents the cross-link interference region wherein UEs can experience cross-link interference.

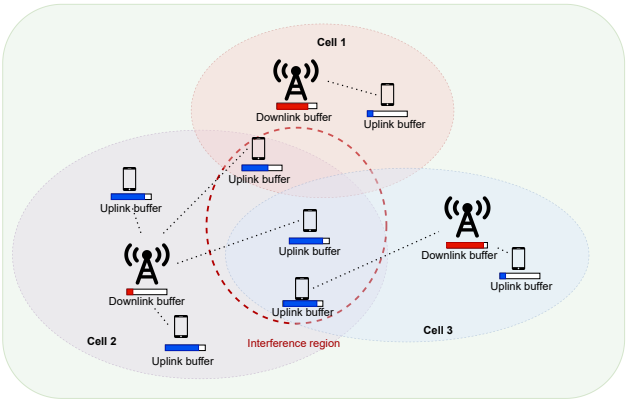


Figure 4: Multi-cell scenario with 3 Radio Units (RU) and 7 User Equipment(UE) with different traffic characteristics showing the interference region

B. Problem formulation

In this work, we assume that all the cells in $\mathcal{C}$ are using the same TDD period δ and are synchronized in time (i.e., slots boundaries of all the cells are aligned). However, the distribution of the UL and DL traffic is unknown. The main research question targeted by this paper is how to distribute the TDD slots among the UL and DL traffic to guarantee that: (i) UL and DL buffers are not overflowed; (ii) edge users do not experience cross-link interference. The main challenge addressed in this paper is that both UL $\lambda_c^{\mathcal{U}}$ and DL $\lambda_c^{\mathcal{D}}$

traffics are unknown and hard to predict. Let $\mu_c^{\mathcal{D}}$ and $\mu_c^{\mathcal{U}}$ denote the transmission capacity of cell c per slot in DL and UL, respectively. We assume that the cell capacity is static during the period δ. Let $\mathcal{X}_c$ be a real variable that denotes the percentage of UL slots available at cell $c \in \mathcal{C}$. Similarly, let $\mathcal{Y}_c$ be a real variable that denotes the percentage of reserved slots for the DL. Formally, the following statements should hold (2), (3) and (4). All the UEs $\gamma \in \Gamma_c$ will share the UL and DL slots. The number of UL and DL slots in which a UE γ is scheduled is proportional to the UE's UL and DL buffers, respectively. Further, the percentage of UL and DL slots in which a UE $\gamma \in \Gamma_c$ is scheduled is defined as follows $\S_\gamma = \frac{\psi_\gamma^{\mathcal{U}}}{\Psi_c^{\mathcal{U}}} \times \mathcal{X}_c$ and $\dagger_\gamma = \frac{\psi_\gamma^{\mathcal{D}}}{\Psi_c^{\mathcal{D}}} \times \mathcal{Y}_c$, respectively.

Let α be a given constant ($0 \leq \alpha \leq 1$) that defines the priority between the UL and DL traffics. This parameter can be defined by the solution designer to specify the priorities between the two traffics. If $\alpha = 1$, then we are interested only in optimizing the UL traffic. Otherwise, if $\alpha = 0$, we are only interested in optimizing the DL traffic.

The proposed solution should be periodically applied to specify $\mathcal{X}_c$ and $\mathcal{Y}_c$ aiming at preventing the overflow of UL buffers $\Psi_c^{\mathcal{U}}$ and DL buffers $\Psi_c^{\mathcal{D}}$. Let $\mathcal{F}_{c_1,c_2}$ be a Boolean constant (i.e., fixed by the system for one iteration) that denotes if there is an interference between the two neighboring cells c_1 and c_2. $\mathcal{F}_{c_1,c_2}$ equals to 0 when there are no edge users between c_1 and c_2. $\mathcal{F}_{c_1,c_2}$ equals to 1 otherwise. If $\mathcal{F}_{c_1,c_2}$ equals to 1, cells c_1 and c_2 must use the same TDD pattern (i.e., the same percentage of UL and DL slots in the TDD period δ). This helps to avoid the cross-link interference at the edge UEs. Otherwise, cells c_1 and c_2 can use different TDD patterns since there is no edge users that will be impacted by the cross-link interference. Let $\Phi_c^{\mathcal{U}}$ and $\Phi_c^{\mathcal{D}}$ denote the initially stored data of the UL and DL buffers. Both $\Phi_c^{\mathcal{U}}$ and $\Phi_c^{\mathcal{D}}$ are initialized by zero. At each iteration, we aim to optimize the following linear integer programming:

$$\min \sum_{c \in \mathcal{C}} \Bigg(\frac{\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{U}}|} \times \Big(\Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - \mu_c^{\mathcal{U}} \times \mathcal{X}_c \times \mathcal{T}_\delta^c \Big) + \frac{1-\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{D}}|} \times \Big(\Phi_c^{\mathcal{D}} + \lambda_c^{\mathcal{D}} - \mu_c^{\mathcal{D}} \times \mathcal{Y}_c \times \mathcal{T}_\delta^c \Big) \Bigg) \tag{1}$$

S.t,

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{X}_c \leq 1 \tag{2}$$

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{Y}_c \leq 1 \tag{3}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c + \mathcal{Y}_c = 1 \tag{4}$$

$$\forall (c_1, c_2) \in \mathcal{C}^2,\ c_1 \neq c_2 : \mathcal{F}_{c_1,c_2} \times \ (\mathcal{X}_{c_1} - \mathcal{X}_{c_2}) = 0 \tag{5}$$

$$\forall c \in \mathcal{C} : \Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - \mu_c^{\mathcal{U}} \times \mathcal{X}_c \times \mathcal{T}_\delta^c \leq |\Psi_{\mathcal{C}}^{\mathcal{U}}| \tag{6}$$

$$\forall c \in \mathcal{C} : \Phi_c^{\mathcal{D}} + \lambda_c^{\mathcal{D}} - \mu_c^{\mathcal{D}} \times \mathcal{Y}_c \times \mathcal{T}_\delta^c \leq |\Psi_{\mathcal{C}}^{\mathcal{D}}| \tag{7}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c \times \mathcal{T}_\delta^c = \mathcal{A}_c \tag{8}$$

$$\forall c \in \mathcal{C} : \mathcal{Y}_c \times \mathcal{T}_\delta^c = \mathcal{B}_c \tag{9}$$

$$\forall c \in \mathcal{C} : (\mathcal{A}_c, \mathcal{B}_c) \in \mathcal{N}^2 \tag{10}$$

The objective function (1) aims to minimize the amount of stored data in the UL and DL buffers in all the cells to prevent their buffers overflow. While $\lfloor \mathcal{X}_c \times \mathcal{T}_\delta^c \rfloor$ denotes the number of slots reserved for the UL traffic, $\lceil \mathcal{Y}_c \times \mathcal{T}_\delta^c \rceil$ corresponds to the number of slots reserved for the DL traffic in each cell $c \in \mathcal{C}$ during the period δ. We have used the weighted normalized sum method to prevent an objective (i.e., buffer) from dominating the other. Meanwhile, constraints (2), (3) and (4) ensure that the variables $\mathcal{X}_c$ and $\mathcal{Y}_c$ are rates of slots distribution for UL and DL. Meanwhile, constraint (5) ensures that two cells experiencing cross-link interference can not use different TDD patterns (i.e., different percentages of UL and DL slots in the TDD period δ), aiming at avoiding the impact of the cross-link interference on the edge UEs. Meanwhile, constraints (6) and (7) ensure that the UL and DL buffers of $c \in \mathcal{C}$ do not overflow, respectively. Note that $\mathcal{A}_c$ and $\mathcal{B}_c$ are two integer variables that denote the number of UL slots and DL slots of cell c, respectively. Constraints (8) and (9) transform the percentages of UL and DL slots to integers given the number of slots available in the period δ. Constraint (10) ensures that the number of UL and DL slots allocated in each cell $c \in \mathcal{C}$ is an integer.

The proposed model can be reformulated by replacing $\mathcal{Y}_c$ by $1 - \mathcal{X}_c$ in equations (1), (3), (5), (7) and (9) and removing the constants from the Objective function (1). Therefore, we obtain a boxed-constrained linear minimization problem as follows:

$$\min \sum_{c \in \mathcal{C}} \Bigg(\Big(\frac{1-\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{D}}|} \times \mu_c^{\mathcal{D}} \times T_\delta^c - \frac{\alpha}{|\Psi_{\mathcal{C}}^{\mathcal{U}}|} \times \mu_c^{\mathcal{U}} \times T_\delta^c \Big) \times \mathcal{X}_c \Bigg) \tag{11}$$

S.t,

$$\forall c \in \mathcal{C} : 0 \leq \mathcal{X}_c \leq 1 \tag{12}$$

$$\forall (c_1, c_2) \in \mathcal{C}^2,\ c_1 \neq c_2 : \mathcal{F}_{c_1,c_2} \times \ (\mathcal{X}_{c_1} - \mathcal{X}_{c_2}) = 0 \tag{13}$$

$$\forall c \in \mathcal{C} : \frac{\Phi_c^{\mathcal{U}} + \lambda_c^{\mathcal{U}} - |\Psi_{\mathcal{C}}^{\mathcal{U}}|}{\mu_c^{\mathcal{U}} \times T_\delta^c} \leq \mathcal{X}_c \tag{14}$$

$$\forall c \in \mathcal{C} : \mathcal{X}_c \leq \frac{-\Phi_c^{\mathcal{D}} - \lambda_c^{\mathcal{D}} + |\Psi_{\mathcal{C}}^{\mathcal{D}}| + \mu_c^{\mathcal{D}} \times T_\delta^c}{\mu_c^{\mathcal{D}} \times T_\delta^c} \tag{15}$$

If we assume a static environment where we know the traffic distribution, it requires an exponential time to solve the problem optimally in the worst case using the simplex method as shown in [19]. However, we should recall that to solve the optimization problem, there is a need to know in advance the exact traffic model, which is not possible in reality.

Unfortunately, we cannot use the optimization problem mentioned above for distributing the slots in a dynamic environment. This is mainly due to the fact that the amount of UL $\lambda_c^{\mathcal{U}}$ and DL $\lambda_c^{\mathcal{D}}$ traffics are unknown and hard to be predicted a priory.

IV. MADRP: Multi-Agent Deep Reinforcement Learning based 5G RAN TDD Pattern

As stated earlier, it is hard to efficiently distribute TDD slots by solving the optimization model without prior knowledge of traffic generation patterns. Besides, traffic patterns between cells may differ, and cross-link interference between neighboring cells may exist. For these reasons, using a multi-agent framework where each agent controls a cell configuration is essential. Each agent will adapt the TDD pattern to accommodate its cell traffic while agents of neighboring cells collaborate with each other to align their TDD pattern to avoid cross-link interference when there are edge UEs between neighboring cells. The multi-agent framework has several motivations: (i) Decreases the control latency since the agents are executed near the gNB; (ii) Reduces the control overhead compared to the centralized solution since the agents observe local information to make a decision instead of sending the observation to a central entity to take the decision; (iii) Decreases the action space since each agent will take one action compared to a single agent solution where the agent will take a tuple of actions. Decreasing the action space makes the training faster and more stable. In this context, we propose the MADRP system that leverages MADRL, more precisely, the MADDPG algorithm, to dynamically define the 5G NR TDD pattern. The MADRL hides the complexity and stochastically of the environment and helps the MADRP framework to make efficient and quick decisions that adapt according to traffic patterns. Besides, MADRL allows different cells to make distributed decisions using local information, enabling communication between cells to collaborate and avoid cross-link interference. Moreover, the MADRP framework gains the ability to learn with time and adapt to different and unseen situations. In the balance of this section, we will present the DRL and MADRL background, the MADRP system overview, more precisely, the MADDPG Algorithm, and a detailed description of the MADRP system.

A. DRL Background

Deep Reinforcement Learning (DRL) will play a crucial role in communication and networking [20] with the ability to provide a self-configured and self-optimized network that easily adapts to network changes. Moreover, DRL is a lightweight framework that enables quick decisions and hence takes real-time actions in the network characterized by its dynamicity and needs fast decisions. DRL techniques are based on the interaction of the DRL Agent with its environment by applying different actions and receiving rewards according to the actions taken. Let $\mathcal{S}$ denotes a set of possible states and $\mathcal{A}$ denotes a set of actions. The state $s \in \mathcal{S}$ is a tuple of the environment's features relevant to the problem at hand. Also, it describes the agent's relation with its environment. Assuming discrete time steps, the agent observes the state of its environment, $s^t \in \mathcal{S}$ at time step t. It then takes action $a^t \in \mathcal{A}$ according to a certain policy π. Once the agent takes action a^t, its environment moves from the current state s^t to the next state s^{t+1}. As a result of this transition, the agent gets a reward r^{t+1} that characterizes its benefit from taking action a^t at state s^t. This scheme forms an experience at time $t+1$, hereby defined as $e^{t+1} = (s^t, a^t, r^{t+1}, s^{t+1})$, which describes an interaction with the environment. The set of interactions $e^t \in \mathcal{E}$ is called Replay Buffer, which is used to train the DRL agent in order to derive the optimal policy π^*. The latter provides the optimal action a^t, to take in each state s^t, in a way to maximize future cumulative discounted reward G^t defined as follows:

$$G^t \doteq \sum_{k=0}^{T} \gamma^k r^{t+k+1} = r^{t+1} + \gamma G^{t+1} \tag{16}$$

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

B. MADRL Background

Unfortunately, traditional DRL approaches, such as Q-Learning or policy gradient, are poorly suited to multi-agent environments. One issue is that each agent's policy changes as training progresses, and the environment becomes non-stationary from the perspective of any individual agent (in a way that is not explainable by changes in the agent's own policy). This presents learning stability challenges and prevents the straightforward use of past experience replay. Policy gradient methods, on the other hand, usually exhibit very high variance when coordination of multiple agents is required [7]. Besides, the state-action space will grow exponentially when a learning agent keeps track of all agent actions. Hence, MADRL approach was introduced to address the above issues.

MADRL system can be represented by the tuple of ($\{\mathcal{S}_j\}_1^{\mathcal{N}}$, $\{\mathcal{A}_j\}_1^{\mathcal{N}}$, $\{\pi_j\}_1^{\mathcal{N}}$, $\{r_j\}_1^{\mathcal{N}}$). Let $\mathcal{G}$ denotes a set of agents. Each agent $j \in \mathcal{G}$ observes a state $s_j^t \in \mathcal{S}_j$ from the environment and executes an independent action a_j^t from its own set of actions $\mathcal{A}_j$ on the basis of its local policy $\pi_j : S_j \rightarrow S_j$. Agents perform joint action $a^t = a_1^t, a_2^t, \cdots, a_N^t \in \mathcal{A}$, where $A = (\mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N)$, which leads the environment to move from state $s_j^t \in \mathcal{S}_j$ to a new state $s_j^{t+1} \in \mathcal{S}_j$, then the agent j receives a reward r_j^{t+1}. In a centralized reward setting (i.e., the agents are cooperating), the agents receive a common reward r^{t+1}. The goal of each agent is to learn a local optimal policy π_j^* that forms a central optimal policy $\pi^* = \pi_1^*, \pi_2^*, \cdots, \pi_N^*$.

In general, MADRL leverages DRL methods for each agent. DRL methods are classified into three categories: $i)$ *value-based* methods, such as DQN; $ii)$ *policy-based* methods, such as REINFORCE (i.e., Monte-Carlo Policy Gradient); $iii)$ *actor-critic* methods that combine the two previous methods, such as A3C and DDPG [21]. In the 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 probabilities 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 that 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 UL and DL slots. For this reason, we have adopted the MADDPG algorithm, which is an improved version of DDPG applied in multi-agent environment. We will explain further the MADDPG Algorithm when explaining our MADRP approach.

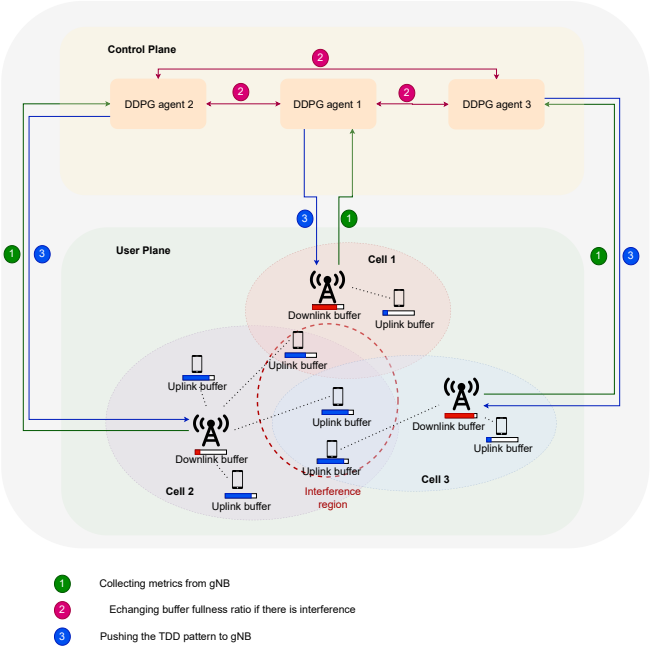


Figure 5: MADRP Architecture

C. MADRP System Overview

As depicted in Figure 5, we propose a MADRL learning scheme with one agent for each cell. The set of agents is part of the RAN control plane, while the RAN functions executing dynamic TDD at the gNB level are part of the RAN user plane. Each agent observes the state of the cell's buffers and exchanges information with neighboring agents (i.e., agents controlling the neighboring cells), thus observing the state of the buffers associated with neighboring cells in case of cross-link interference. Communication between agents can be based on the Xn Application Protocol (XnAP) specification to comply with 3GPP [22]. Specifically, an agent will send an XnAP message containing the Information Element (IE) "Intended TDD DL-UL Configuration NR" to inform the destination gNB of the source gNB's current TDD configuration in order to mitigate cross-link interference.

We have adopted a centralized training and decentralized execution approach. Thus, we allow the policies to use extra information to ease the training step, so long as this information is not used at test time. It is unnatural to do this with Q-learning, as the Q function generally cannot contain different information at training and test time. Thus, MADDPG extends the actor-critic policy gradient methods, whereby the critic is augmented with extra information about other agents' policies.

More concretely, let consider $\mathcal{N}$ agents with policies parameterized by $\theta = (\theta_1, ..., \theta_{\mathcal{N}})$, and let $\pi = (\pi_1, ..., \pi_{\mathcal{N}})$ be the set of all agent policies. Let $[R_i]$ be the cumulative discounted reward of agent i. As we have seen in the background section, each agent aims to maximize R_i. And, since MADDPG is a policy gradient method, it samples the actions directly from the policy π. Let $\mu = (\mu_1,..., \mu_{\mathcal{N}})$ be a continuous and deterministic policy function. Then we can write the gradient of the expected cumulative reward for agent i, $J(\theta_i) = \mathbb{E}[R_i]$ as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\nabla_{\theta_i}\boldsymbol{\mu}_i(a_i \mid o_i)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}(s, a_1, \ldots, a_N)\big|_{a_i=\boldsymbol{\mu}_i(o_i)}\right] \tag{17}$$

Where $Q_i^{\mu}(s, a_1, ..., a_{\mathcal{N}})$ is a centralized action-value function that takes as input the actions of all agents, $a_1, ..., a_{\mathcal{N}}$ as well as some state information s and outputs the Q-value for agent i. In the simplest case, s could consist of the observations of all agents, $s = (o_1, ..., o_{\mathcal{N}})$. However, we may also include additional state information if available. Since each Q_i^{μ} is separately learned, agents can have arbitrary reward structures, including conflicting rewards in a competitive setting. Here the experience replay buffer $\mathcal{D}$ contains the tuples $(s^t, s^{t+1}, a_1^t, ..., a_{\mathcal{N}}^t, r_1^{t+1}, ..., r_{\mathcal{N}}^{t+1})$, recording experiences of all agents.

D. MADRP detailed description

We have designed the MADRP to be lightweight to ensure fast interaction with the environment. Also, we have designed the MADRP to ensure generality and then work in an unseen environment. Besides, MADRP has been designed to work independently from the number of slots, the size of the buffers, and the number of cells. 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 MADRP, including the state, the reward, and the action.

i) State: Let $\xi_{\mathcal{U},c}^t$ and $\xi_{\mathcal{D},c}^t$ denote the amount of traffic in the UL $\Psi_c^{\mathcal{U}}$ and the DL $\Psi_c^{\mathcal{D}}$ at the step t at the cell c, respectively. To ensure the generalization, we define the observation $\mathcal{O}_{\mathcal{U},c}^t$ and $\mathcal{O}_{\mathcal{D},c}^t$ of the UL and DL buffers as normalized values (Figure 6: 1). Formally, $\mathcal{O}_{\mathcal{U},c}^t = \frac{\xi_{\mathcal{U},c}^t}{|\Psi_c^{\mathcal{U}}|}$ and $\mathcal{O}_{\mathcal{D},c}^t = \frac{\xi_{\mathcal{D},c}^t}{|\Psi_c^{\mathcal{D}}|}$, respectively. The benefits of the normalization are twofold: i) It ensures generality by enabling the MADRP agent to be agnostic to the scenario scale in terms of the buffer capacity. It works similarly in different buffers with different sizes. The most important is to catch the buffer fullness ratio of $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$; ii) It is well known that the activation functions in the neural network work well for small values, which positively impacts MADRP's convergence. Moreover, to capture the traffic patterns, we define the state s_j^t of the agent j at time t as follows:

$$s_j^t = \bigcup_{k=0}^{\mathcal{N}} \hat{s}_{j,k}^t \tag{18}$$

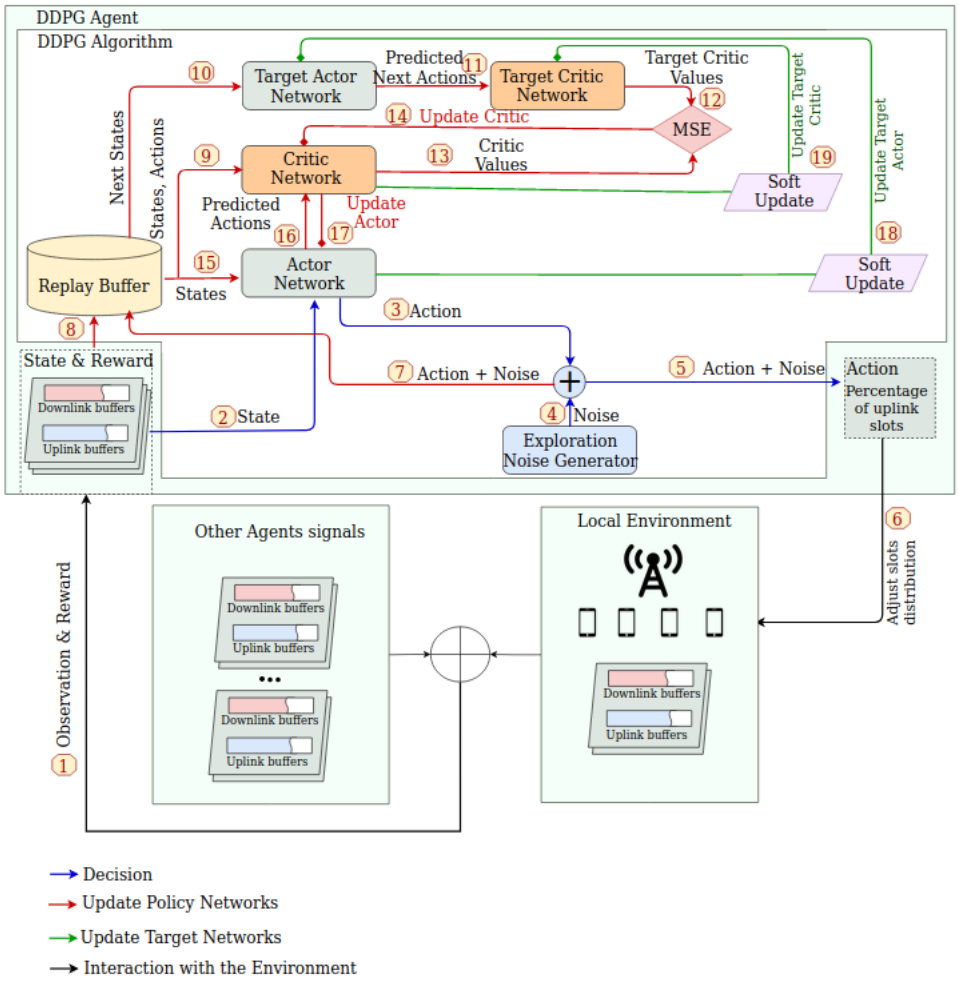


Figure 6: MADDPG architecture and workflow

Where $\hat{s}^t_{j,k}$ is defined as follows:

$$\hat{s}^t_{j,k} = \begin{cases} \left(\bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}^{t-i}_{\mathcal{U},\,j}, \bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}^{t-i}_{\mathcal{D},\,j} \right) & if\ k\ =\ j\ or\ \mathcal{F}(k,j) = 1 \\ \left(\bigcup_{i=0}^{\mathcal{K}-1} 0, \bigcup_{i=0}^{\mathcal{K}-1} 0 \right) & else \end{cases} \tag{19}$$

, whereby $\mathcal{F}(k,j) = 1$ when there is an interference between cells k and j (i.e., direct cross-link interference) or $\exists k' \in \mathcal{C}$ such that $\mathcal{F}(k,k') = 1$ or $\mathcal{F}(k',j) = 1$ (i.e., indirect cross-link interference)

In the state s^t_j, the MADRP agents, besides the current observation, consider $\mathcal{K}$ previous observations before taking any action. This enables capturing the behavior of both buffers and traffic before taking any action.

Besides, MADRP agents capture the behavior of the function $\mathcal{F}$ by communicating with each other. An agent j sends a message containing $\left(\bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}^{t-i}_{\mathcal{U},\,j}, \bigcup_{i=0}^{\mathcal{K}-1} \mathcal{O}^{t-i}_{\mathcal{D},\,j} \right)$ to agent k when agent j observes an edge user with the cell associated to agent k. This message is considered an interference alert. The operator can define a policy to trigger the interference alert, for example, when the number of edge users exceeds a predefined threshold. Each agent stores a list of agents with which it has interference, and each time it receives an interference alert, it broadcasts the received alert to its interference list. This mechanism allows MADRP to capture indirect interference (i.e., interference between two cells without edge users between each other but $\exists k \in \mathcal{C}$ such that there is interference between cell k and both cells).

ii) Action: We have only one continuous action a^t_j per agent j that presents the percentage of slots that should be reserved for the UL traffic at step t. Each MADRP agent $j \in \mathcal{G}$ enforces the taken decisions as depicted in Figure 6:6. Accordingly, the $\lfloor a^t_j \times \mathcal{T}^j_\delta \rfloor$ slots are reserved for the UL traffic, and $\lceil (1 - a^t_j) \times \mathcal{T}^j_\delta \rceil$ slots are reserved for the DL traffic. It is worth noting that the action is independent of the number of slots $\mathcal{T}^j_\delta$, which ensures the generality and enables the MADRP 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 it ends.

Let $\widehat{r^t_j}$ denotes the buffer reward of each agent j, defined in (20) as follows:

$$\widehat{r^t_j} = \begin{cases} \alpha \times (1 - \mathcal{O}^t_{\mathcal{U},\,j}) + (1 - \alpha) \times (1 - \mathcal{O}^t_{\mathcal{D},\,j}) & if\ |\mathcal{O}^t_{\mathcal{U},\,j}| < 1 \wedge |\mathcal{O}^t_{\mathcal{D},\,j}| < 1 \\ -\mathcal{M} & else \end{cases} \tag{20}$$

Equation (20) allows MADRP agents to get a positive reward

for each step, which succeeds in keeping the buffers $\Psi_c^{\mathcal{U}}$ and $\Psi_c^{\mathcal{D}}$ 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 force the MADRP agents to keep both buffers empty as much as possible and prevent their overflow, which positively impacts the Quality of Service (QoS). α is the priority between the UL and DL traffic.

Let $\overline{r_{j,\,k}^{t}}$ denotes the cross-link interference reward, defined in (21) as follows:

$$\overline{r_{j,\,k}^{t}} = \begin{cases} 0 & if\ \mathcal{F}(j,k)\ = 0\ \vee\ \mathcal{F}(j,k)\ = 1\ \wedge\ a_j^{t-1} = a_k^{t-1} \\ -\mathcal{M} & else \end{cases} \tag{21}$$

Equation (21) gives a penalty when two agents experience direct or indirect interference and do not choose the same TDD pattern (i.e., the same percentage of UL slots).

The agent reward r_j^t is the sum of the buffer and the cross-link interference reward, defined in (22) as follows:

$$r_j^t = \widehat{r_j^t} + \sum_{k\ \in\ \mathcal{C}} \left(\overline{r_{j,\,k}^{t}}\right) \tag{22}$$

As depicted in Figure 6, the MADRP system leverages the MADDPG algorithm and is executed on three different steps: i) Decision making (Figure 6: $1-6$) presented with blue color; ii) Updating policy networks (Figure 6: $7-17$) presented with red color; iii) Updating target networks (Figure 6: $18-19$) presented by green color. Each MADDPG agent has two networks: a) Policy networks that consist of the Actor and the Critic neural networks. These networks are used to predict the deterministic actions a_j^t. While the actor-network has as input the state s_j^t and it is used to predict the action a_j^t, the critic-network has as inputs $s^t = s_1^t,\ ...,\ s_{\mathcal{N}}^t$ and $a^t = a_1^t,\ ...,\ a_{\mathcal{N}}^t$ and returns the Q value that is used for criticizing the taken action; b) The target networks that consist of target actor-networks and target critic-network. These two networks are frozen and used to help the convergence of policy networks and stabilize their learning. In deep learning, the optimizer (e.g., 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 replay buffer sample, reducing the correlation between the agents' experiences.

Decision making: At the reception of the observation $(\mathcal{O}_{\mathcal{U},c}^t, \mathcal{O}_{\mathcal{D},c}^t, \bigcup_{i=0}^{\mathcal{N}} n_j^t)$, with n_j^t is the number of users at the edge between the current agent' cell c and agent j' cell. n_j^t is used to set the variable $\mathcal{F}(c,j)$, i.e., $\mathcal{F}(c,j) = 1$ if $n_j^t > 0$ or a message is received from agent j, $\mathcal{F}(c,j) = 0$ otherwise. Each MADRP agent generates the state s_j^t using the equation (19) (Figure 6: 2). The received state is used by the actor-network to predict the deterministic action a_j^t. In order to enable the MADRP agents to explore the environment, a noise $\mathcal{J}_j^t$ is added to the action. Accordingly, the UL and DL slots are reserved (Figure 6: $5-6$).

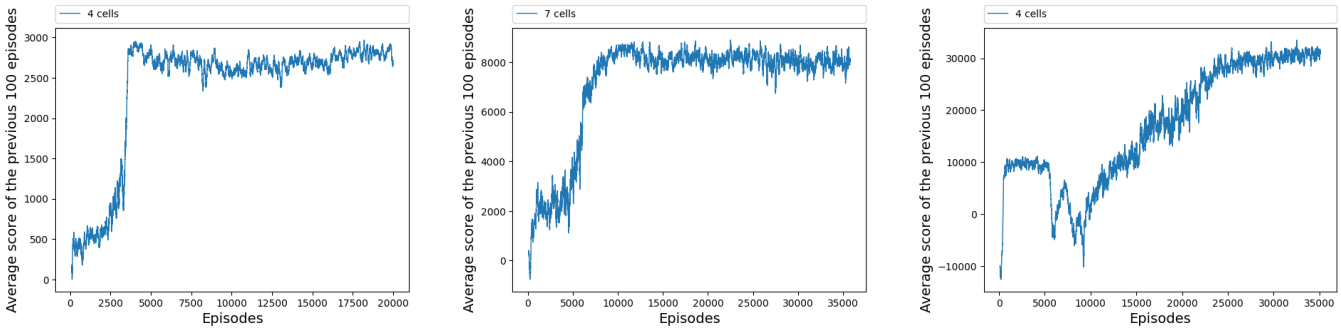
Algorithm 1 Multi-Agent Deep Deterministic Policy Gradient for $\mathcal{N}$ agents

1: **for** episode = 1 to $\mathcal{M}$ **do**
2: Initialize a random process $\mathcal{J}_i$ for action exploration for each agent i
3: Receive initial state s
4: **for** t = 1 to max-episode-length **do**
5: for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{J}_i^t$ w.r.t. the current policy and exploration
6: Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state s'
7: Store (s, a, r, s') in replay buffer $\mathcal{D}$
8: $s \leftarrow s'$
9: **for** agent i = 1 to N **do**
10: Sample a random minibatch of S samples $\left(s^j, a^j, r^j, s'^j\right)$ from $\mathcal{D}$
11: Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}\left(s'^j, a'_1, \ldots, a'_N\right)\Big|_{a'_k = \boldsymbol{\mu}'_k(o_k^j)}$
12: Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\mu}\left(s^j, a_1^j, \ldots, a_N^j\right)\right)^2$
13: Update actor using the sampled policy gradient: $\nabla_{\theta_i} J \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\boldsymbol{\mu}_i\left(o_i^j\right)\nabla_{a_i} Q_i^{\boldsymbol{\mu}}\left(s^j, a_1^j, \ldots, a_i, \ldots, a_N^j\right)\Big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$
14: **end for**
15: Update target network parameters for each agent i: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
16: **end for**
17: **end for**

Updating policy network: In order to take optimal actions, the policy network should be updated (Figure 6: $7-17$). The action taken by all the agents should be stored in the replay buffer (Figure 6: 7), as well as their corresponding state and reward (Figure 6: 8). First the critic-network is updated by leveraging a random batch sample $(s^t,\ s^{t+1},\ a_1^t + \mathcal{J}_1^t,\ ...,\ a_{\mathcal{N}}^t + \mathcal{J}_n^t,\ r_1^{t+1},\ ...,\ r_{\mathcal{N}}^{t+1})$ from the replay buffer (Figure 6: $9-14$). 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.

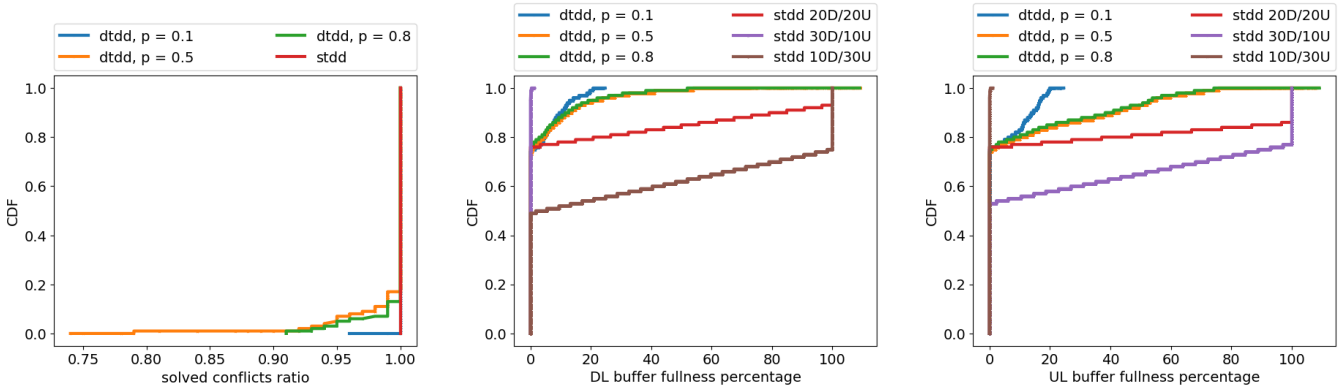
Updating target network: The target networks (actor and critic) should be updated slowly and periodically towards the policy networks using soft update (Figure 6: $18-19$). This strategy helps the Algorithm for providing optimal deterministic action. We utilize a distributed training procedure, where only the critic-networks are identical among the agents while the actor-networks are specific for each agent. Hence, the execution phase is distributed, with each agent making its own decision at each interval relying only on the specific observations it receives from the environment and its neighbors.

Overall, the MADDPG algorithm is summarized in Algo-



(a) 4 cells (b) 7 cells (c) 14 cells

Figure 7: Convergence evaluation of MADRP agent during the training mode



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 8: Performance evaluation of MADRP of 4 agents during the inference mode

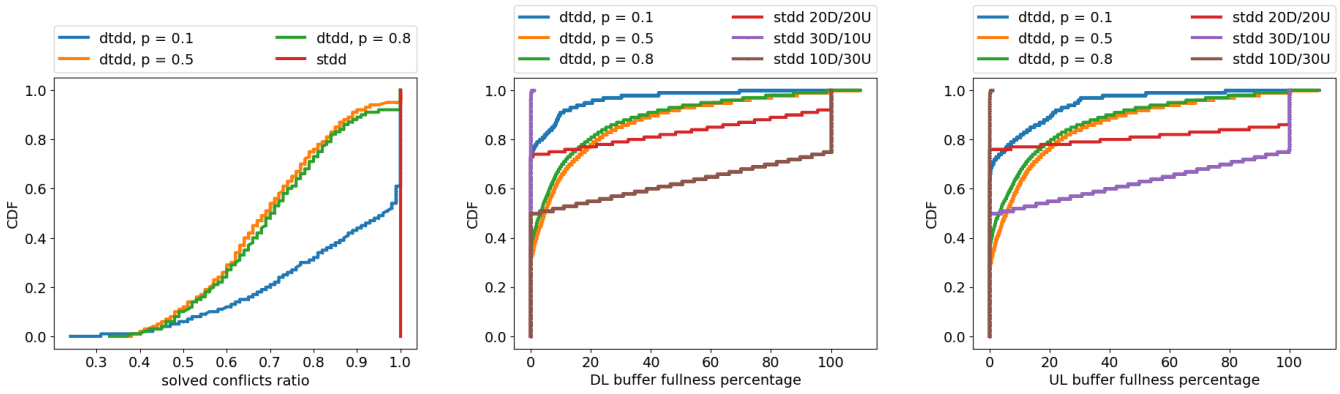
rithm 1 where $\mu' = (\mu'_1, ..., \mu'_{\mathcal{N}})$ is the set of target policies with delayed parameters θ'_i.

V. Performance Evaluation

We implemented our simulation environment using Python and Pytorch. We leveraged the simulator built in [5] and the open-source MADDPG implementation [7]. We used a physical machine with Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz with 64 GB of memory and NVIDIA GP102 GPU using Ubuntu 20 as an operating system. Following an empirical approach, we conducted several experiments with different neural network parameters and activation functions. We selected those which gave us the best performance. For all the agents, we employed two fully connected hidden layers of 400 and 300 nodes for both policy and target networks. We 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 employed a discount factor γ of 0.99, batch size of 1024, and the learning rates of the actor and critic-networks are set to 10^{-5} and 10^{-3}, respectively. We used the soft update with coefficient τ 0.001. Also, ADAM optimizer has been leveraged in both actor and critic-networks. Finally, the optimization problem is solved using Gurobi version 9.1.2. The absolute optimality gap is set to 10^{-8}.

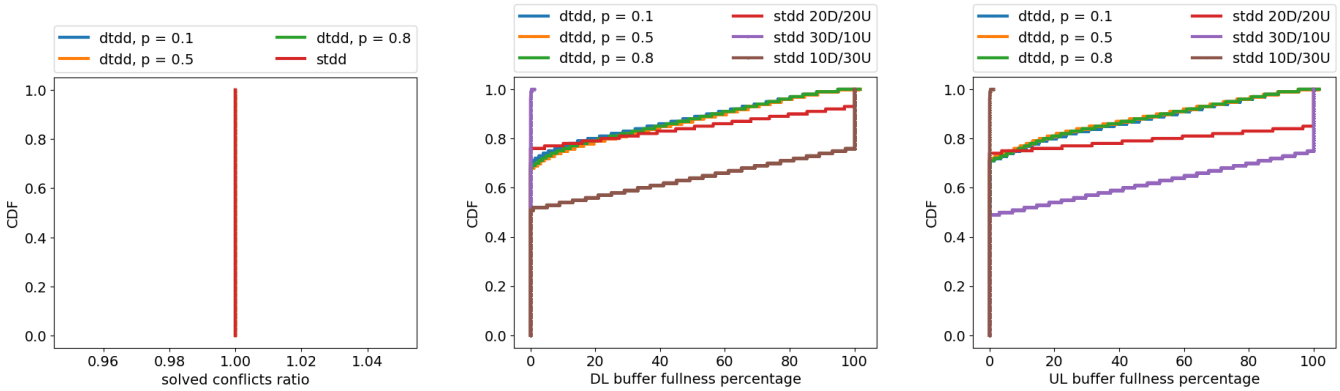
A. MADRP Training mode

As shown in Figure 7, we considered three different configurations: (i) 4 neighboring cells; (ii) 7 neighboring cells; (iii) 14 neighboring cells. For each configuration, we trained 4, 7, and 14 agents, respectively. We trained the MADRP agents using 20000, 35000, and 35000 independent episodes for configuration (i), (ii) and (iii), respectively. We set the maximum number of steps in each episode T to 200. We set the penalty $\mathcal{M}$ to -100, and the number of slots $\mathcal{T}_\delta^c$ to 40. We considered three successive observations (i.e., $\mathcal{K} = 3$). We used a Poisson distribution to generate the traffic in each cell for the UL and DL. The arrival rates for UL and DL are randomly chosen at each iteration $\lambda \in \{50, 100, 200, 300\}$ with unit u per δ. We assumed a fixed serving rate μ for each cell (i.e., the amount of data scheduled at each slot equals 14 times the unit u; 14 being the number of symbols per slot in 5G NR). Figures 7(a), 7(b), and 7(c) depict the evolution of the averaged 100 sum rewards of all the agents over time. We observe that MADRP converges at 5000, 15000, and 30000 episodes for configuration (i), (ii), and (iii), respectively. Thus, we observe that the more agents we add, the more episodes the agents need to converge.



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 9: Performance evaluation of MADRP of 7 agents during the inference mode



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

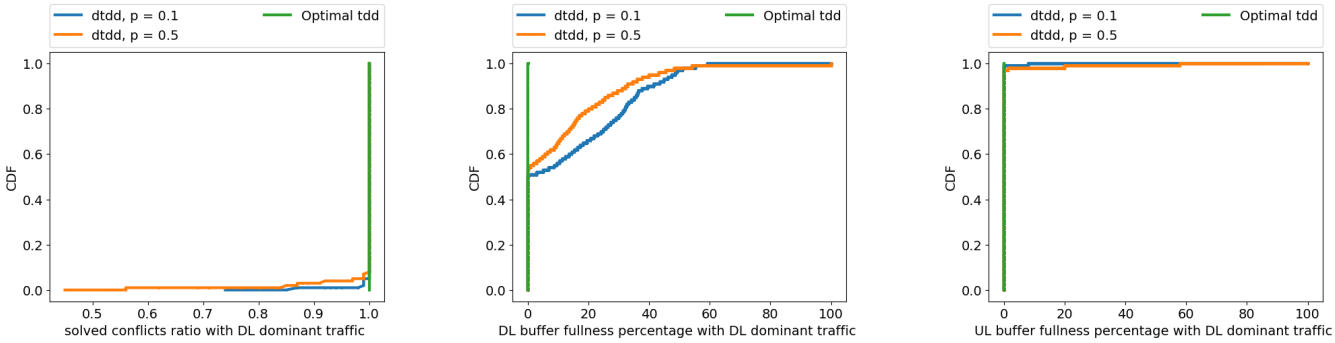
Figure 10: Performance evaluation of MADRP of 14 agents during the inference mode

B. MADRP Inference mode

For the three configurations and under different interference probabilities, we have evaluated the MADRP agents in terms of i) The percentage of the fullness of the UL and DL buffers; ii) The number of conflicts successfully mitigated. Let p be the probability that agent i has a direct cross-link interference with agent j (i.e., cell i has edges users with cell j). We recall that indirect cross-link interference can happen by transitivity even if there are no edge users between two cells. The terms dTDD and sTDD in the figures' legend stand for dynamic TDD and static TDD, respectively. It should be noted that we considered different static TDD configurations: (i) stdd 20D/20U: half of the TDD period slots (i.e., 20 slots) are dedicated to DL, while the other half are dedicated to UL; (ii) stdd 30D/10U: 75% of the slots are dedicated to DL (iii) stdd 10D/30U: 75% of the slots are dedicated to UL.

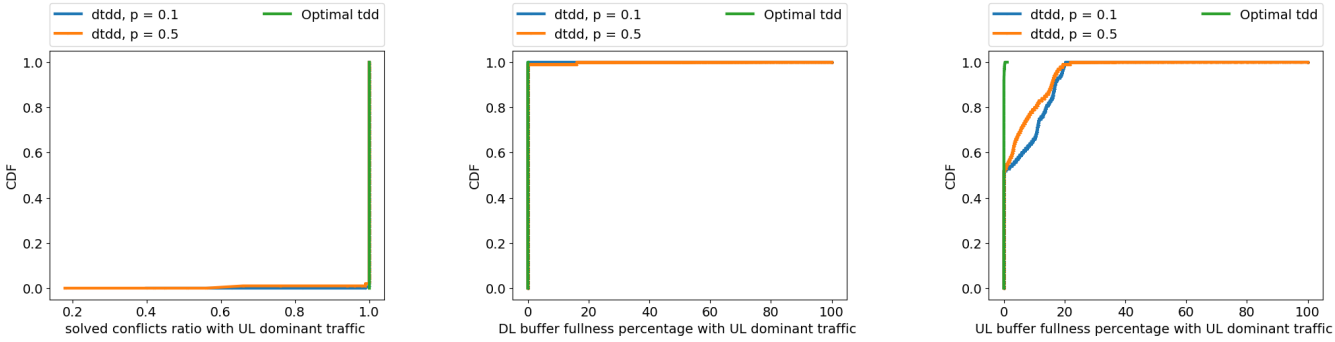
In Figure 8, we evaluated the performance of 4 agents during 1000 independent episodes, each of which with 200 iterations. We considered three different interference probabilities p. Figure 8(a) depicts the Cumulative Distribution Function (CDF) of the ratio of solved interference conflicts among all the interference cases. A solved interference conflict means two agents aligned their TDD pattern while there is direct or indirect interference between them. The x-axis represents the solved conflicts ratio, while y-axis represents its CDF. For $p = 0.1$ (i.e., agent i' cell has a probability of 0.1 to have an edge user with agent j' cell), we notice that MADRP is able to solve more than 96% of the conflicts. While for $p = 0.5$ and $p = 0.8$, MADRP is able to solve more than 80% and 92% of the conflicts, respectively. We noticed that with $p = 0.8$, there is always interference (i.e., direct + indirect interference). While in static TDD, no interference is present since all the cells share the same TDD pattern. For instance, we consider all the conflicts are solved since there is no unsolved conflict. We can see that MADRP outperforms all static TDD solutions. For the 20D/20D configuration, MADRP is better in UL and DL, while for the 30D/10U configuration, MADRP is better in UL, and for the 10D/30U configuration, MADRP is better in DL. Indeed, over 60% of the samples represent a buffer overflow for 30D/10U and 10D/30U static TDD in DL and UL, respectively. As a result, MADRP is able to balance DL and UL traffic dynamically.

In Figure 9, we evaluated the performance of 7 neighboring agents during 1000 independent episodes, each of which with 200 iterations. Figure 9(a) depicts the CDF of the ratio of solved interference conflicts among all the interference cases. For $p = 0.1$, MADRP solved more than 90% of the conflicts during 60% of the samples and more than 70% of the conflicts



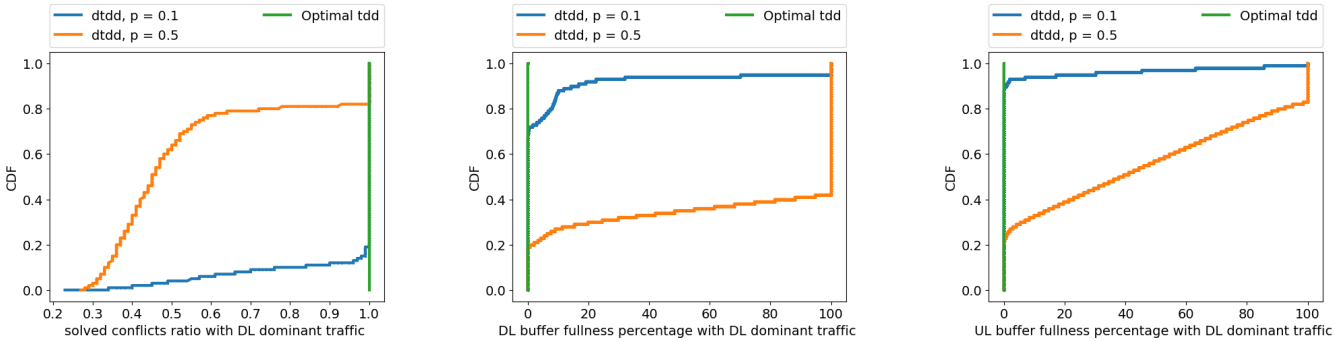
(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 11: Performance evaluation of MADRP of 4 agents during the inference mode with DL dominant traffic



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 12: Performance evaluation of MADRP of 4 agents during the inference mode with UL dominant traffic

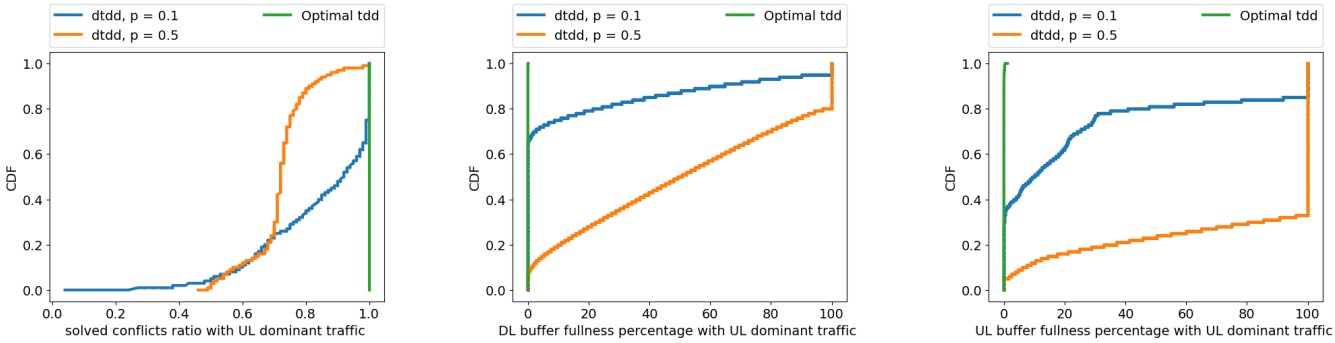


(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 13: Performance evaluation of MADRP of 7 agents during the inference mode with DL dominant traffic

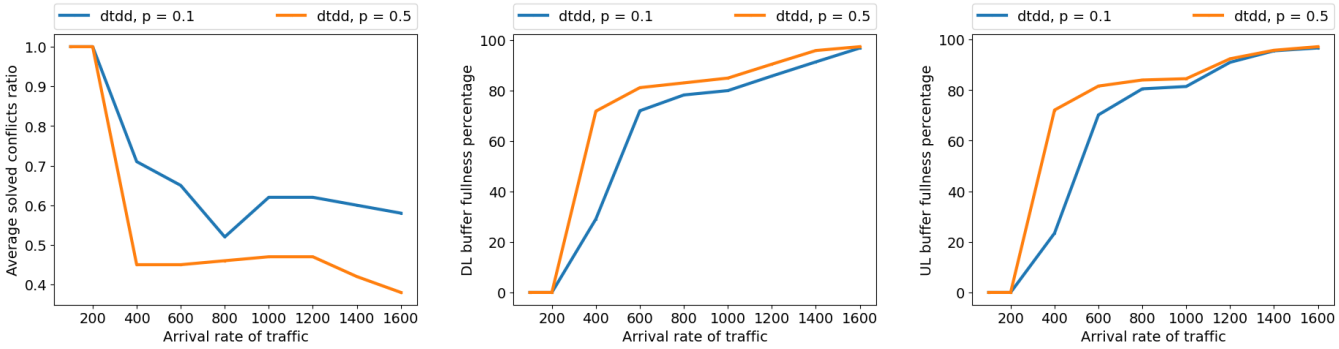
among the 80% of the samples. While for $p = 0.5$ and $p = 0.8$, we notice that MADRP solves more than 70% of the conflicts among 50% of the samples. Figures 9(c) and 9(b) depict the CDF of the UL and the DL buffers' fullness percentage, respectively. For $p = 0.1$, the buffer size is lower than 10% in 80% of the samples and higher than 20% in less than 2% of the samples. While for $p = 0.5$ and $p = 0.8$, the buffer size is lower than 20% in 70% of the samples and lower than 40% in 80% of the samples. In all cases, MADRP outperforms all static TDD solutions. In the case of static TDD 20D/20U, we observe that around 20% of samples represent a buffer overflow (i.e., the buffer is full) in both UL and DL. In contrast, in the 30D/10U and 10D/30U static TDD solutions, around 70% of samples represent buffer overflow in UL and DL, respectively.

In Figure 10, we evaluated the performance of 14 neighboring agents during 1000 independent episodes, each of which with 200 iterations. Figure 10(a) depicts the CDF of the ratio of solved interference conflicts among all the interference cases. We observe that all the conflicts are solved for all the conflict rates. This is because MADRP found that the configuration (50% DL slots, 50% UL slots) is the best due to the strong dynamics of the environment. Since the configuration is fixed among all the agents, all the interference



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 14: Performance evaluation of MADRP of 7 agents during the inference mode with UL dominant traffic



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 15: Performance evaluation of MADRP of 4 agents during the inference mode

cases are mitigated. Figures 10(c) and 10(b) depict the CDF of the UL and the DL buffer's fullness percentage, respectively. We observe that the buffer size is lower than 40% in 80% of the samples and lower than 80% in 90% of the samples. In all the cases, MADRP outperforms all the static TDD solutions. We observe that, in static TDD 20D/20U, around 20% of the samples represents buffer overflow (i.e., the buffer is full) in both UL and DL. While in static TDD 30D/10U and 10D/30U, around 70% of the samples represents buffer overflow in UL and DL, respectively.

Figures 11 and 13 illustrate the performance evaluation of MADRP when traffic is DL-dominant. That is, traffic arrives with higher arrival rates in the DL direction, while traffic arrives with lower arrival rates in the UL direction. We compared the MADRP solution with the optimization model solution.

In Figure 11(b), we see that the optimal solution empties the buffer continuously, whereas the MADRP solution keeps the buffer below 60% in 99% of the time. The gap between the optimal solution and the MADRP solution is 1%, representing the percentage of cases where the DL buffer was full. We recall that the aim of MADRP is to avoid buffer overflow without the knowledge of the arriving traffic model.

In Figure 13(b), we see that the optimal solution empties the buffer permanently, whereas the MADRP solution keeps the buffer below 20% in 90% of cases when the probability of interference is low, while the buffer overflows in 60% of cases when the probability of interference is high.

Figures 12 and 14 depict the performance evaluation of MADRP when the traffic is UL-dominant. We have compared the MADRP solution with the optimization model solution.

In Figure 12(c), we note that the optimal solution always empties the buffer, whereas the MADRP solution keeps the buffer below 20% in 99% of the time. In figure 14(c), the MADRP solution keeps the buffer empty in 40% of samples and below 40% in 80% of the time in the UL buffers when the probability of interference is low. We note that the buffer overflows in 60% of cases when the probability of interference is high.

Figures 15 and 16 depict the impact of arrival rates on the MADRP agents for configurations with 4 and 7 cells, respectively. Each plotted point represents the average of 1000 episodes, each of which with 200 steps. The first observation we can draw from this figure is the amount and the variance of the traffic handled by the MADRP agents. Figures 15(a) and 16(a) depict the average solved interference ratio variation over the arrival rates for different interference probabilities. The x-axis represents the arrival rate of the traffic, while the y-axis represents the average solved conflicts. We notice that while λ increases, the average ratio of solved conflicts decreases. We recall that the serving cell capacity is $\mathcal{T}_{\delta}^{c} \times \mu$ (i.e., 560). Hence, we notice that when $\lambda \geq \mu$, MADRP is able to solve around 30% and 60% of the interference. When $\frac{\mu}{2} \geq \lambda \leq \mu$ in the 7 agents scenario, MADRP is able to solve between 60% and

95% for $p = 0.1$ and between 50% and 60% of the interference for $p = 0.5$. While in the 4 agents scenario, MADRP is able to solve all the interferences. We conclude that when increasing λ, the traffic variance increases (i.e., due to Poisson distribution), which introduces more aggressive conflicts in the agents' observations. This will results in difficulties in aligning the TDD pattern between interfered neighbors while serving each cell's traffic.

Figures 15(b) and 16(b) depict the average DL buffer fullness percentage in the 4 cells and 7 cells environment, respectively. While figures 15(c) and 16(c) depict the average UL buffer fullness percentage in the 4 cells and 7 cells environment, respectively. We notice that while λ increases, the average buffer fullness ratio increases. When λ reaches 800 units, the average buffer size is 70% for $p = 0.1$ and 85% when $p = 0.5$. We conclude that even when the arrival rate is bigger than the serving rate, MADRP is able to balance the traffic between UL and DL.

Overall, MADRP is able to change the TDD pattern dynamically by allocating UL and DL slots to different cells while avoiding direct and indirect cross-link interference between cells. When there is a small number of neighbors (i.e., 4 cells), small direct interference probability (i.e., $p = 0.1$), and an arrival rate lower than a serving rate with a traffic variance lower than half of the arrival rate, MADRP is able to avoid more than 96% of the interference cases while serving different UL and DL traffic rates. Indeed, MADRP allocates the number of UL/DL slots needed for each cell in each iteration according to the UL/DL buffer size, and hence it avoids buffer overflow while avoiding cross-link interference. However, when we increase the number of cells, the probability of direct interference, or the variance of the traffic, MADRP starts finding difficulties in satisfying the traffic load in UL/DL while solving the interference issues. In all the cases, MADRP outperforms the static TDD solution by reducing the probability of buffer overflow in both UL and DL.

In conclusion, we propose to create small subsets of neighboring cells (e.g., 7 cells per subset) that use the same frequency band in order to make dynamic TDD more efficient.

VI. Conclusion

In this paper, we introduced MADRP, a Multi-Agent Deep Learning Reinforcement (DRL)-based solution that permits deriving and adjusting the TDD pattern in 5G NR while mitigating cross-link interference. MADRP approach consists in deploying a MADRP agent at each gNB serving a cell. Without prior knowledge of the UEs traffic model, each MADRP agent computes the number of slots dedicated to UL and DL in a TDD frame aiming at reducing both UL and DL buffers while avoiding cross-link interference with the neighboring cells. Simulation results clearly showed that MADRP could avoid buffer overflow and dynamically adapt to the cell traffic while avoiding cross-link interference. Further, the results showed that MADRP clearly outperforms the different static TDD configurations with different UL/DL proportions and the gap between the optimal solution and the MADRP solution is small in low-interference scenarios. Our future focus is on implementing MADRP on top of OpenAirInterface (OAI) 5G to demonstrate self-adapted and plug-and-play deployment of 5G and MADRP.

Acknowledgment

This work was partially supported by the European Union's Horizon Program under 6GBricks project (Grant No. 101096954).

References

[1] Shihao Shen et al. "EdgeMatrix: A Resource-Redefined Scheduling Framework for SLA-Guaranteed Multi-Tier Edge-Cloud Computing Systems". In: *IEEE Journal on Selected Areas in Communications* 41.3 (2023), pp. 820–834. DOI: 10.1109/JSAC.2022.3229444.

[2] Shaoyuan Huang et al. "Fine-Grained Spatio-Temporal Distribution Prediction of Mobile Content Delivery in 5G Ultra-Dense Networks". In: *IEEE Transactions on Mobile Computing* (2022), pp. 1–14. DOI: 10.1109/TMC.2022.3226448.

[3] NGMN Alliance. "6G Use Cases and Analysis". In: (2022).

[4] Karim Boutiba, Miloud Bagaa, and Adlen Ksentini. "On enabling 5G Dynamic TDD by leveraging Deep Reinforcement Learning and O-RAN". In: *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*. 2023, pp. 1–3. DOI: 10.1109/NOMS56928.2023.10154404.

[5] Miloud Bagaa, Karim Boutiba, and Adlen Ksentini. "On using Deep Reinforcement Learning to dynamically derive 5G New Radio TDD pattern". In: *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, pp. 1–6. DOI: 10.1109/GLOBECOM46510.2021.9685820.

[6] Florian et al. Kaltenberger. "The OpenAirInterface 5G new radio implementation: Current status and roadmap". In: *WSA 2019, 23rd ITG Workshop on Smart Antennas, Demo Session, 24-26 April 2019, Vienna, Austria*. 2019.

[7] Ryan Lowe et al. "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6382–6393. ISBN: 9781510860964.

[8] Samir Si-Mohammed et al. "UAV mission optimization in 5G: On reducing MEC service relocation". In: *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 2020, pp. 1–6. DOI: 10.1109/GLOBECOM42002.2020.9322304.

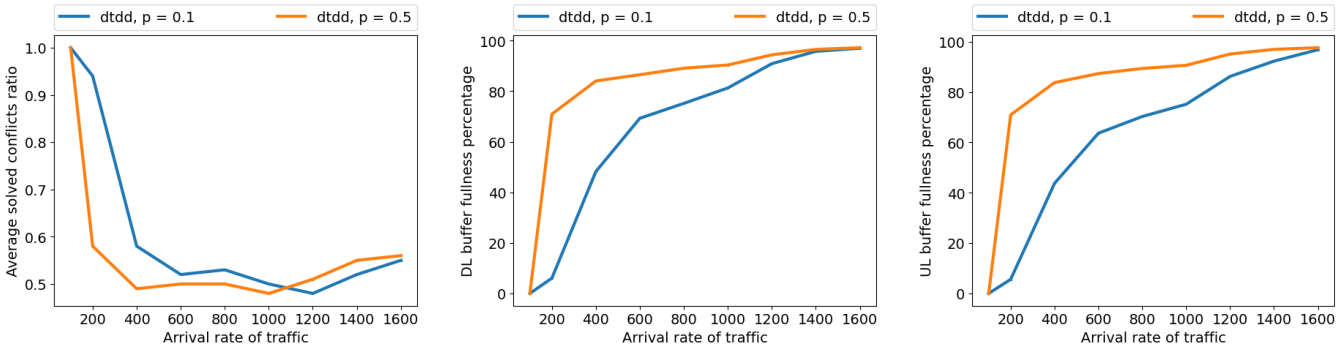
[9] 3GPP. *5G; NR; Radio Resource Control (RRC); Protocol specification*. Technical Specification (TS) 38.331. Version 15.3.0. 3rd Generation Partnership Project (3GPP), Oct. 2018.

[10] Fengxiao Tang, Yibo Zhou, and Nei Kato. "Deep Reinforcement Learning for Dynamic Uplink/Downlink Resource Allocation in High Mobility 5G HetNet". In: *IEEE Journal on Selected Areas in Communications* 38.12 (2020), pp. 2773–2782. DOI: 10.1109/JSAC.2020.3005495.

[11] Rudraksh Shrivastava, Konstantinos Samdanis, and Vincenzo Sciancalepore. "Towards service-oriented soft spectrum slicing for 5G TDD networks". In: *Journal of Network and Computer Applications* 137 (2019), pp. 78–90. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2019.01.009. URL: https://www.sciencedirect.com/science/article/pii/S1084804519300177.

[12] Shaozhen Guo, Xiaolin Hou, and Hanning Wang. "Dynamic TDD and interference management towards 5G". In: Apr. 2018, pp. 1–6. DOI: 10.1109/WCNC.2018.8377314.

[13] Yasar Sinan Nasir and Dongning Guo. "Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks". In: *IEEE Journal on Selected Areas in Communications* 37.10 (2019), pp. 2239–2250. DOI: 10.1109/JSAC.2019.2933973.



(a) Conflicts mitigation (b) DL buffer (c) UL buffer

Figure 16: Performance evaluation of MADRP of 7 agents during the inference mode

[14] Hyejin Kim, Jintae Kim, and Daesik Hong. "Dynamic TDD Systems for 5G and Beyond: A Survey of Cross-Link Interference Mitigation". In: *IEEE Communications Surveys and Tutorials* 22.4 (2020), pp. 2315–2348. DOI: 10.1109/COMST.2020.3008765.

[15] Cho-Hsin Tsai et al. "QoE-aware Q-learning based approach to dynamic TDD uplink-downlink reconfiguration in indoor small cell networks". In: *Wireless Networks* 25 (Aug. 2019). DOI: 10.1007/s11276-019-01941-8.

[16] Ali A. Esswie, Klaus I. Pedersen, and Preben E. Mogensen. "Online Radio Pattern Optimization Based on Dual Reinforcement-Learning Approach for 5G URLLC Networks". In: *IEEE Access* 8 (2020), pp. 132922–132936. DOI: 10.1109/ACCESS.2020.3011026.

[17] Zhi Yu et al. "Dynamic resource allocation in TDD-based heterogeneous cloud radio access networks". In: *China Communications* 13.6 (2016), pp. 1–11. DOI: 10.1109/CC.2016.7513198.

[18] Xiangyu Chen, Gang Chuai, and Weidong Gao. "Multi-Agent Reinforcement Learning Based Fully Decentralized Dynamic Time Division Configuration for 5G and B5G Network". In: *Sensors* 22.5 (2022). ISSN: 1424-8220. DOI: 10.3390/s22051746. URL: https://www.mdpi.com/1424-8220/22/5/1746.

[19] John Fearnley and Rahul Savani. "The Complexity of the Simplex Method". In: *CoRR* abs/1404.0605 (2014). arXiv: 1404.0605. URL: http://arxiv.org/abs/1404.0605.

[20] N. C. Luong et al. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". In: *IEEE Communications Surveys Tutorials* 21.4 (2019), pp. 3133–3174. DOI: 10.1109/COMST.2019.2916583.

[21] Kai Arulkumaran et al. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240.

[22] 3GPP. *5G; NGRAN; Xn Application Protocol (XnAP)*. Technical Specification (TS) 38.423. Version 16.5.0. 3rd Generation Partnership Project (3GPP), Apr. 2021.

Karim Boutiba received his Engineering degree in Computer Systems from the National School of Computer Science (ESI) , Algiers, Algeria, in 2020. Currently, he is pursuing a Ph.D. in the Communication Systems department at EURECOM, France, under the supervision of Pr. Adlen Ksentini. He is working towards enforcing Network Slicing in 5G networks and beyond. His research interests include Next-Generation Networking and Internet, 5G New Radio, Network Slicing, Open RAN, Optimization algorithms and Reinforcement Learning for 5G networks and beyond.

Miloud Bagaa currently is a professor in the department of electrical and computer engineering of UQTR. He received the engineer's, master's, and Ph.D. degrees from the University of Science and Technology Houari Boumediene, Algiers, Algeria, in 2005, 2008, and 2014, respectively. From 2009 to 2015, he was a researcher with the Research Center on Scientific and Technical Information, Algiers. From 2015 to 2016, he was postdoctoral researcher with the Norwegian University of Science and Technology, Trondheim, Norway. From 2016 to 2019, he was a postdoctoral researcher at Aalto University. From 2019 to 2020, he was a senior researcher at Aalto University. Last but not least, he was a visiting researcher at Aalto University and a senior system cloud specialist at CSC from Oct. 2020 until Nov. 2022.

Adlen Ksentini is a professor in the Communication Systems Department of EURECOM. He is leading the Network softwarization group. He is involved in several EU projects related to Network Slicing and 5G, such as 5G!Drones and MonB5G. He is leading the Network softwarization group activities related to Network Slicing and Edge Computing. He has been involved in several H2020 EU projects on 5G, such as 5G!Pagoda, 5GTransformer, 5G!Drones and MonB5G. Adlen Ksentini research interests are on Network Sofwerization and Network Cloudification focusing on topics related to: network virtualization, Software Defined Networking (SDN), Edge Computing, Network slicing for 5G and beyond networks. He is interested on both system and architectural issues, but also on algorithms problems related to those topics, using Markov Chains, Optimization algorithms and Machine Learning (ML). Adlen Ksentini has received the best paper award from IEEE IWCMC 2016, IEEE ICC 2012, and ACM MSWiM 2005 conferences, and has been awarded the 2017 IEEE Comsoc Fred W. Ellersick (best IEEE communications Magazine's paper).