Deep Reinforcement Learning for URLLC data management on top of scheduled eMBB traffic

1st Fabio Saggese
CNIT // Dept. of Information Engineering
University of Pisa
Pisa, Italy
fabio.saggese@phd.unipi.it

2nd Luca Pasqualini
Dept. of Information Engineering and Mathematics
University of Siena
Siena, Italy
pasqualini@diism.unisi.it

3rd Marco Moretti
CNIT // Dept. of Information Engineering
University of Pisa
Pisa, Italy
marco.moretti@unipi.it

4th Andrea Abrardo
CNIT // Dept. of Information Engineering and Mathematics
University of Siena
Siena, Italy
abrardo@dii.unisi.it

***Abstract*—With the advent of 5G and the research into beyond 5G (B5G) networks, a novel and very relevant research issue is how to manage the coexistence of different types of traffic, each with very stringent but completely different requirements. We propose a *Deep Reinforcement Learning* (DRL) algorithm to *slice* the available physical layer resources between ultra-reliable low-latency communications (URLLC) and enhanced Mobile BroadBand (eMBB) traffic. Specifically, in our setting the time-frequency resource grid is fully occupied by eMBB traffic and we train the DRL agent to employ Proximal Policy Optimization (PPO), a state-of-the-art DRL algorithm, to dynamically allocate the incoming URLLC traffic by puncturing eMBB codewords. Assuming that each eMBB codeword can tolerate a certain limited amount of puncturing beyond which is in outage, we show that the policy devised by the DRL agent never violates the latency requirement of URLLC traffic and, at the same time, manages to keep the number of eMBB codewords in outage at minimum levels, when compared to other state-of-the-art schemes.**

***Index Terms*—RAN Slicing, Deep Reinforcement Learning, URLLC, PPO.**

I. Introduction

Resource slicing of different kinds of traffic is a key enabler for 5G and B5G networks, allowing the coexistence on a common infrastructure of different services with different requirements such as eMBB and URLLC [1]. The two kind of traffic have different quality-of-service (QoS): eMBB users require high throughputs, while URLLC has strict low-latency and reliability constraints [2]. In particular, URLLC traffic is characterized by short packets that need to be transmitted and decoded in less than 1 ms [3], so that conventional channel-aware scheduling is generally not possible.

Addressing the problem of URLLC-eMBB scheduling, [2] compares the performance of different techniques in the uplink of a 5G system and lays the ground for the subsequent literature using either *puncturing*, orthogonal multiple access (OMA) and non-orthogonal multiple access (NOMA). Immediate scheduling of URLLC packets in combination with hybrid automatic repeat request (HARQ) is another approach investigated in [4]. In [5] eMBB codewords are punctured to accomodate URLLC traffic and the throughput loss for eMBB packets is evaluated under different models. In [6] the authors propose a resource allocation scheme for URLLC-eMBB traffic based on successive convex approximation and semidefinite relaxation of the general optimization problem.

Because of its ability of finding very good to optimal policies for systems that dynamically change through time [7], *Reinforcement Learning* (RL) is a natural choice to address the random dynamics of URLLC traffic. Usually, RL is employed in its *Deep* (DRL) formulation, where a multi-layer Neural Network (NN) is employed to extract features from states hardly enumerable in the simpler tabular approaches. Accordingly, in [8], and [9] the authors propose two RL algorithms based on Q-learning to multiplex eMBB and URLLC traffic employing OMA and NOMA, respectively. In [10] and [11], DRL approaches based on policy gradient algorithms are proposed to choose the resources punctured by the URLLC transmission. The former requires *a-priori* information to the number of resources to be punctured, the latter relies on the instantaneous channel state information (CSI).

Most of the recent literature [2], [5], [8]–[11] assumes that the URLLC packets are transmitted as soon as they arrive. However, as long as they satisfy their latency constraints, URLLC packets can tolerate a certain amount of delay and this (minimum) tolerable delay can be exploited to give the scheduler some degree of freedom. In this paper, we address the slicing problem by allowing some slack for URLLC transmissions to minimize their impact on eMBB traffic. We assume a system where all resources are already allocated to eMBB traffic, so that every time there is a URLLC transmission an eMBB packet needs to be punctured. The URLLC scheduler is a DRL agent, which select the resources for the URLLC packets with the goal of minimizing the outage of the eMBB traffic due to puncturing. To enforce slice isolation, the control planes of the different slices are kept to a minimum degree of interaction [5], and the URLLC the eMBB schedulers are two independent entities. Accordingly, the only information the URLLC scheduler needs to possess is the robustness of each eMBB codeword to puncturing. Moreover, because of the strict latency requirements, we assume that the URLLC scheduler does not have instantaneous channel state information. While our proposed codeword model resembles the *threshold model* described in [5], it retains two important differences. First, we consider a more realistic non-homogeneous situation where different puncturing policies can be adopted at different times. Then, we consider a threshold per codeword rather than per user. Nevertheless, the model is kept at a minimum complexity in order to preliminary investigate the capability of a DRL approach to automatically extract and employ the simulation's features to devise a generalized scheduling strategy acting independently with respect to the URLLC packet arrival and the eMBB scheduling process.

II. System Model

We consider a single cell scenario in which one base station (BS) serves a set of downlink user equipments (UE) with different requirements. We denote as U and E the number of URLLC and eMBB users, respectively. We consider a single coherence interval as time horizon, where the channel can be considered constant. The time axis is divided into Σ equally spaced time slots of fixed duration. To accommodate URLLC traffic, with its stringent latency requirements, slots are further divided into M minislots[1]. As for the frequency domain, the system bandwidth is divided into F orthogonal frequency resources (FR)[2].

We consider two different schedulers, one for each type of traffic, which operate separately and independently of each other. The *eMBB scheduler* is responsible for assigning time and frequency resources to eMBB users: each eMBB codeword can occupy any fraction of the total available number of minislots and FRs. As customary, eMBB scheduling is operated at the slot boundaries. At the same time, the *URLLC agent* operates on a per minislot basis with the possibility of puncturing some of the resources already assigned to eMBB users, if needed. In the following, we present a detailed description of how we model the traffics, and their coexistence.

A. The eMBB Scheduler

In this paper, we do not explicitly address the eMBB scheduling problem, but, rather, we assume that a proper radio resource allocations has been performed somehow and we can focus on the coexistence of URLLC traffic on top of eMBB. Nevertheless, we need to describe the main principles of the eMBB scheduling policy, which is to maximize a rate-dependent utility function, not considering any latency. Hence,

[1]In 3GPP, the formal term for a "slot" is eMBB Transmit Time Interval (TTI), and a "minislot" is a URLLC TTI [5].

[2]With "frequency resources" we refer to the abstract concept of bandwidth available in an OFDM system and we may refer to resource blocks or subcarriers, indifferently.

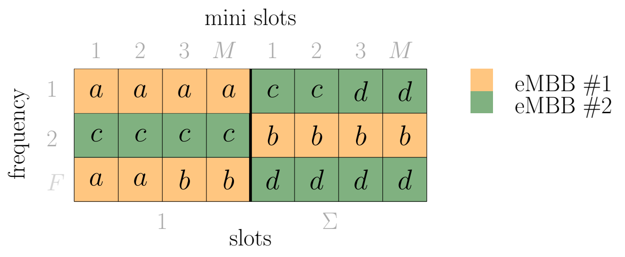

Fig. 1. Toy example of the resource allocation and codeword placement for the eMBB users, $F = 3$, $\Sigma = 2$, $M = 4$. Resources are allocated at slot boundaries, while codewords are $a, b \in \mathcal{W}_1$, $c, d \in \mathcal{W}_2$ and $|a| = |b| = |c| = |d| = 6$.

radio resources are allocated to the set of active users on a slot basis following the OMA paradigm. Moreover, since there is enough time to exchange channel quality information (CQI) before each scheduling decision, it is reasonable to assume perfect knowledge of CSI at the BS. Therefore, eMBB resource allocation can be performed following conventional methods such as the water-filling algorithm [12].

The scheduler has to further take into account that the eMBB packets might share the radio resources with URLLC traffic and in such event they should carry enough redundancy to be punctured without losing the entire packet. We denote as $\mathcal{W}$ the codebook at the BS. The BS will then select a subset $\mathcal{W}_e \subset \mathcal{W}$ containing all the codewords of user e. A single codeword intended for user e is denoted as $w \in \mathcal{W}_e$. The length in symbols $|w|$ of a codeword is always a multiple of the minislot length, i.e., each codeword spans an integer number of minislots. Finally, we denote with $w_{t,f}$ the codeword transmitted on the radio resource f during minislot t and with $\mathcal{W}_t = \bigcup_{\nu=1}^{F} w_{t,\nu}$ the set of all codewords transmitted during the minislot t. Figure 1 shows a toy example of a possible resource allocation and codeword placement for two eMBB users.

B. The URLLC DRL agent

Generally speaking, the QoS requirements of an URLLC user $u \in \mathcal{U}$ in a wireless network are specified as follows: a packet of size N_u bits must be successfully delivered to the receiver within an *end-to-end delay* of no more than $T_u^{\max}$ seconds with a probability of at least $1 - \epsilon_u$ [4]. Moreover, a URLLC packet may randomly arrive at the BS at any moment. In this work we will concentrate on the edge delay, i.e. the delay computed as the difference between the time the scheduler receives the packet and the time the packet is transmitted. This choice is justified by the fact that the backhaul delay is generally negligible [13], while UL queuing delay and transmission delay can be taken into account by reducing the value of the tolerable latency $T_u^{\max}$. Without loss of generality, we define the tolerable latency in terms of the maximum number $l_u^{\max}$ of minislots that can be waited before exceeding the latency constraint.

To simplify the description of the problem, our work will focus on URLLC packets of fixed length corresponding to a single minislot. The packets are generated following memoryless packet arrival distributions: Bernoulli with arrival probability p_u and Poisson with mean value λ_u. The packets are stored in a first-in first-out (FIFO) queue $\mathcal{Q}$ of infinite length. The DRL URLLC agent is responsible for taking the decision whether the oldest packet in the queue should be transmitted or not in the current minislot, and onto which frequency resource in the grid.

Owing to the stringent latency constraint, the CSI of URLLC users cannot be estimated. Hence, power adaptation during transmission is not possible and ARQ re-transmission mechanisms can be hardly acceptable. Accordingly, reliability can be expressed in terms of outage probability for a fixed predefined transmit power. As in previous works in literature [5], we assume that the URLLC transmit power is large enough so that the outage probability remains under an acceptable threshold.

C. URLLC and eMBB Coexistence

Coexistence of eMBB and URLLC is achieved by superposition coding or puncturing [1]- [5]. In this paper we consider a puncturing strategy, where the BS decides to use a certain resource for URLLC traffic regardless of any eMBB user already occupying it. To avoid any interference between the two types of traffic, the eMBB codeword is punctured, i.e., the transmit power of the eMBB user on the specific resource is set to zero. To tolerate puncturing, we assume that each eMBB codeword employs an inner erasure code with rate $1 - C_w/|w|$ [2], that allows to correct up to C_w erased minislots. The eMBB scheduler is in charge of determining the *class* C_w for each codeword. The class assignment is performed on a codeword basis, i.e., the BS can assign codewords with different C_w to the same user. Note that the algorithm implemented by the eMBB scheduler may be unknown by the URLLC traffic agent, as long as the latter is informed of the codewords class by the former.

III. Reinforcement Learning

RL is usually employed to solve a Markov Decision Process (MDP) defined over a real world task. A MDP is defined via a dynamic environment, a state space $\mathcal{S}$, an action space $\mathcal{A}$, and a reward function $R(a, s)$ with $a \in \mathcal{A}$, $s \in \mathcal{S}$ [7]. In a MDP, the decision maker, also referred to as *agent*, gets a reward from the environment upon taking an action. The action also causes the environment to change its internal state. At each time step t, the agent receives a state $S_t \in \mathcal{S}$ from the environment, and then selects an action $A_t \in \mathcal{A}$. The environment answers with a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and a next state S_{t+1}. This interaction gives a *trajectory* of random variables, also known as episode:

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots$$

The objective of RL is to find a policy that maximizes the expected discounted reward, where the discount factor $\gamma \in [0, 1)$ determines the length of the time horizon within each trajectory, so that $\gamma \to 1$ means an infinite horizon.

It's common knowledge in literature that RL is especially effective when paired with parametric function approximators for the policy $\pi(a|s)$, specifically multi-layer NNs (hence the name *deep*). That holds true especially for each task where the state space is too complex to be represented in tabular form, making it impossible for the algorithm to compute the estimated value of each state in a reasonable time [7]. Since the simulation's state observed by the proposed agent at each time step t is of combinatorial complexity, as shown in section III-A, in this paper we follow the DRL paradigm when training the agent, according to literature best practices.

A. System Model as a MDP

The application of RL to our task requires to formulate the URLLC scheduling problem as a MDP. Despite the task at hand being inherently not episodic, for convenience of operation we truncate it in multiple episodes of length T minislots, corresponding to the whole coherence interval of the channel. A minislot $t \in \{1, \ldots, T\}$ represents a time step in the episode. At the beginning of each episode, resource allocation and codeword placement for eMMB users is performed and then, at each time step, new URLLC packets are generated according to a certain distribution.

The DRL action consists in deciding whether the first URLLC packet in the queue should be transmitted in the current minislot or not and on which FR. The *possible actions* at time step t are collected in the set $\mathcal{A}_t = \{0, 1, \ldots, F\}$, where 0 means no transmission, while otherwise the action indicates the FR index for transmitting the URLLC packet. If the URLLC queue is empty, the only possible action is 0.

The *state* at each time step t is then represented by the set $\mathcal{S}_t = \{\mathcal{S}_t^{(u)}, \mathcal{S}_t^{(e)}\}$, where $\mathcal{S}_t^{(u)}$ and $\mathcal{S}_t^{(e)}$ collect the URLLC and eMBB information at step t, respectively. In particular, the 2-dimensional state $\mathcal{S}_t^{(u)}$ is

$$\mathcal{S}_t^{(u)} = \{Q_t, \Delta_t\} \tag{1}$$

where Q_t represents the length of the URLLC queue at step t, while $\Delta_t = l_u^{\max} - l_t^{\text{old}}$ represents the difference between the tolerable latency and the latency of the oldest packet in the queue at step t. The F-dimensional state $\mathcal{S}_t^{(e)}$ collects for each of the F frequency channels the variable $s_t(f)$, which tracks if the codeword transmitted on channel f is in outage ($s_t(f) = -1$) or not ($s_t(f) \geq 0$). A non-negative $s_t(f)$ stores the residual number of times that the codeword can be punctured without being in outage. Let $\rho_t(w)$ denote the number of times the codeword w has been punctured from the beginning of the episode, $s_t(f)$ is computed as

$$s_t(f) = \max\left\{C_{w_{t,f}} - \rho_t(w_{t,f}), -1\right\}, \tag{2}$$

remembering that $w_{t,f}$ is the codeword placed on resource f and minislot t. Once a codeword is in outage, its state variable is set to -1 and does not change anymore regardless of the times is further punctured.

B. Reward Computation

In a RL problem choosing the reward is an empirical process: a good reward function should capture the essence of the task at hand. In this case the objective is to minimize the number of eMBB codewords in outage while keeping the latency of URLLC packets below the given threshold. With this goal in mind, we introduce the eMMB penalty function $e_t(w)$

$$e_t(w) = \begin{cases} -1, & \mathcal{C}_w - \rho_{t-1}(w) \geq 0 \cap \mathcal{C}_w - \rho_t(w) < 0, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

which takes value -1 only if the chosen action causes the outage of the codeword w. Furthermore, since $\Delta_t < 0$ signals the violation of the latency constraint, we introduce the following URLLC penalty function

$$L_t = \begin{cases} 0, & \Delta_t \geq 0, \\ -\frac{\alpha T}{F+1}, & \Delta_t < 0. \end{cases} \quad (4)$$

The heuristic value $-\frac{\alpha T}{F+1}$ is empirically chosen so that the violation of the latency constraint for an URLLC packet results in a larger negative contribution than the outage penalty for eMBB traffic. In the worst case, every punctured resource will lead to an outage event and the total maximum contribution of the eMBB penalty is $-T$. However, T is a large value which can lead to numerical instabilities within the learning process, and it is not highly probable that an eMBB outage event occurs every step. Hence, we normalize T by $(F+1)/\alpha$ ($\alpha \geq 1$) which is a term that accounts for to the fact that the larger is the number of frequency resources, the less probable are outage events. The value of α is devised by means of trial and error, according to best practices of reward engineering in RL [7]. Finally, the reward at time t can be expressed by

$$R_t = \sum_{w \in \mathcal{W}_t} e_t(w) + L_t, \quad (5)$$

Eventually, when $\Delta_t < 0$ the episode is considered finished.

C. Algorithm and Neural Network (NN) Architecture

Among the possible DRL techniques, we consider a Policy Gradient (PG) algorithm called Proximal Policy Optimization (PPO) [14]. PPO aims at taking the biggest possible improvement step on a policy without ending too far from the previous one, thus avoiding the risk of performance collapse. While we know that many papers in literature focus on well known methods as Deep Q-Learning (DQL) or Vanilla Policy Gradient (VPG), both presents huge drawbacks w.r.t. PPO. Specifically, DQL cannot scale to large action spaces required by complex tasks as the one at hand, while VPG is extremely unstable during training. PPO is considered state-of-the-art for PG methods in current RL literature, by being an efficient approximation of Trust Region Policy Optimization (TRPO) [15], the latter being proved to guarantee a monotonic improvement over the expected discounted reward w.r.t. the training steps.

PPO is an actor-critic algorithm [14], where two different neural networks are required. To this respect, we consider two completely separated subnetworks, one for the *value function* (the critic, with output estimated value of current state) and one for the *policy function* (the actor, with output the current strategy). Both policy and value function subnetworks have three dense layers with 128, 64, and 32 neurons, respectively. All of them operate a rectified linear activation function (ReLU). Furthermore, the policy subnetwork has a dense fourth layer with $F+1$ neurons to choose the actions, while the value subnetwork has a dense fourth layer with 1 neuron and no activation to estimate the value. Finally, all layers are initialized using Xavier initialization. For additional details refer to the GitHub repository [16].

IV. RESULTS

We consider a scenario, where the slot duration and the coherence time of the channel are set to 1 and 10 ms, respectively. Each slot is further divided in $M = 14$ minislots. The number of frequency resources is $F = 12$. The length of an episode corresponds to the coherence time of the channel so that the number of time slots for each episode is $\Sigma = 10$, for a total of $T = 140$ minislots. We consider a single URLLC user, i.e. $U = 1$, and the number of eMBB users is $E = 10$. We further set the maximum delay constraint to $l_u^{\max} = M/2 = 7 = 0.5$ ms. We consider only codewords of class $C_w \in \{0, 1\}$, i.e., codewords that can be punctured zero or one times before being in outage.

Regarding PPO, we use an instance of PPO-Clip with a clip ratio equal to 0.2, and an early stopping strategy if the mean KL-divergence of the new policy from the old one grows beyond a given threshold, set as $1.5 \cdot 10^{-2}$, as described in [17]. To reduce the variance of the states' values stored into the various trajectories we feed the NN with, we use the generalized advantage estimation approach with $\gamma_{\text{GAE}} = 1$ and $\lambda_{\text{GAE}} = 0.97$, as proposed in [18]. The value of α in (4) is 3.

To have a fair performance comparison, we consider four alternative URLLC scheduling algorithms:

- *Aggressive*. The URLLC packet is transmitted immediately on a randomly chosen frequency.
- *Threshold Proportional (TP)*. The URLLC packet is transmitted immediately on the frequency resource occupied by the codeword with the highest puncturing threshold, given by (2). TP has almost optimal performance when the URLLC is forced to transmit immediately upon arrival, i.e., $l_u^{\max} = 1$, and in case of low average URLLC load [5].
- *TP-lazy*. As long as $\Delta_t > 0$, the packet is transmitted only if $\sum_{w \in \mathcal{W}_t} C_w - \rho_t(w) \geq \sum_{w \in \mathcal{W}_{t+1}} C_w - \rho_t(w)$, i.e. if the present state is somehow better (or equal) than the next one. If $\Delta_t = 0$, the transmission is forced in the present minislot. In any case, the choice of the frequency is made according to the TP scheme. This heuristic combines the advantage of the TP transmission policy with the possibility of waiting before puncturing eMBB resources.

- *TP-smart*. This heuristic immediately transmits the URLLC packet. If there is an eMBB codeword already in outage the scheduler transmits on the resources occupied by that codeword, otherwise TP-smart acts as TP. Note that this scheme results optimal, in terms of impact on eMBB codewords, for immediate transmissions in this scenario.

During the *learning phase* of the PPO agent, the parameters related to eMMB resource allocation and URLLC traffic generation are randomized on a per episode basis. While this is not mandatory to train a functioning agent, it is crucial to help the agent to learn a generalized strategy that is not specific either for a particular eMMB allocation policy or a particular URLLC traffic load. Since the MDP formulation considered in this paper models a not episodic task, to compute the expected discounted reward we set $\gamma = 0.99$. To further simulate a continuous task, we initialize each episode with a random number of URLLC packets in the queue. The number of packets generated in this way is always smaller than $l_u^{\max}$ to avoid that the episode starts with $\Delta_t < 0$. After training the agent with Bernoulli distribution of packets, we show the results obtained by running the RL-based agent in *inference mode*, providing comparisons with the considered heuristic schemes.

A. Bernoulli Distribution

Here we discuss the results obtained for the Bernoulli distribution.

Table I shows the total episode reward $\sum_{t=1}^{T} R_t$ as a function of p_u, for $T = 140$. It is worth noting that the PPO agent trained as proposed in this paper, outperforms all the other schemes for every value of p_u. In Table II, we show the average number of packets remaining in the URLLC queue at the end of each episode for different p_u. The results for aggressive, TP-smart and TP are omitted since the URLLC packets are promptly transmitted upon arrival. At the opposite, with the TP-lazy scheme a non-negligible amount of traffic remains unserved at the end of an episode. We can see that the PPO agent is able to devise a new policy in-between the two. It's worth noting that while TP, TP-lazy, aggressive and TP-smart are all designed to to *never violate the URLLC latency constraints* for Bernoulli distribution, the PPO agent learns this on its own.

The subdivision of the task into episodes may somehow distort the correct evaluation of the algorithms' performance. To simulate a longer time horizon is of critical importance to the real word task. To address this issue, we repeated our tests scaling up the length of each episode by one order of magnitude, *without retraining the agent*.

Fig. 2 shows the percentage of eMMB codewords in outage at the end of each episode for the various schemes, with $T = 1400$, while the class of each codeword is again randomly chosen. Also in this case, the PPO agent outperforms all the other schemes.

TABLE I
TOTAL REWARD VERSUS ACTIVATION PROBABILITY p_u.

p_u	0.1	0.2	0.3	0.4	0.5	mean
aggressive	-4.60	-9.04	-14.01	-18.55	-23.34	-13.91
TP	-1.31	-3.23	-6.71	-9.85	-15.78	-7.38
TP-lazy	-1.18	-3.18	-5.72	-9.25	-14.63	-6.79
TP-smart	-0.77	-1.64	-2.34	-3.28	-4.23	-2.43
PPO	-0.48	-1.18	-1.955	-2.69	-3.77	-2.03

TABLE II
AVERAGE NUMBER OF URLLC PACKETS NOT SERVED BEFORE THE END OF THE EPISODE.

p_u	0.1	0.2	0.3	0.4	0.5
TP-lazy	0.597	1.222	1.790	2.416	3.015
PPO	0.030	0.068	0.081	0.136	0.210

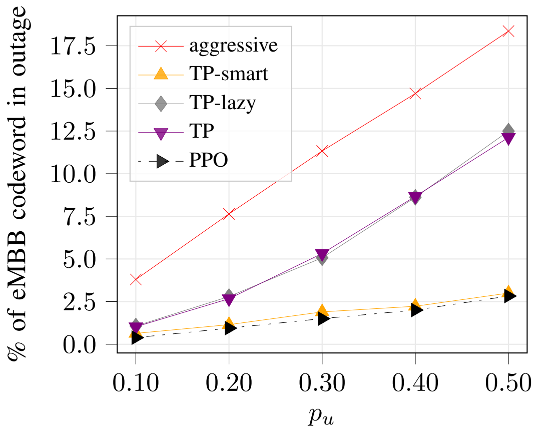

Fig. 2. Percentage of eMBB codeword in outage versus activation probability p_u, $T = 1400$.

Finally, Fig. 3 shows the percentage of eMBB codewords in outage for different compositions of eMBB codewords classes $D = [\Pr\{\mathcal{C}_0\}, \Pr\{\mathcal{C}_1\}]$ and $p_u = 0.3$. The PPO agent outperforms all the heuristic schemes, including the one we considered optimal at one step. Among the other things, we can see that PPO has good performance even when $D = [1, 0]$, i.e. there are only codewords without an inner erasure code.

B. Poisson Distribution

In this section, we discuss the results obtained for the Poisson distribution, collected for $T = 140$. To assess the generalization capabilities of PPO, we use the same network trained on the environment with Bernoulli distribution.

In Fig. 4, we show the performance of the various schemes in terms of eMBB codewords' outages. Also in this case, the proposed PPO agent attains best performance. It's worth noting that TP-lazy heuristic scheme is omitted since it's not able to deal with the bursts of packets generated under the Poisson assumption, thus consistently violating URLLC latency requirements. All other schemes, *PPO included*, never violate them.

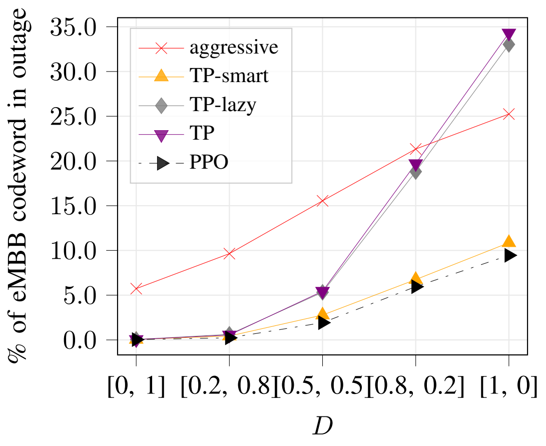

Fig. 3. Percentage of eMBB codewords in outage versus the different percentage of classes of codeword for probability of activation $p_u = 0.3$.

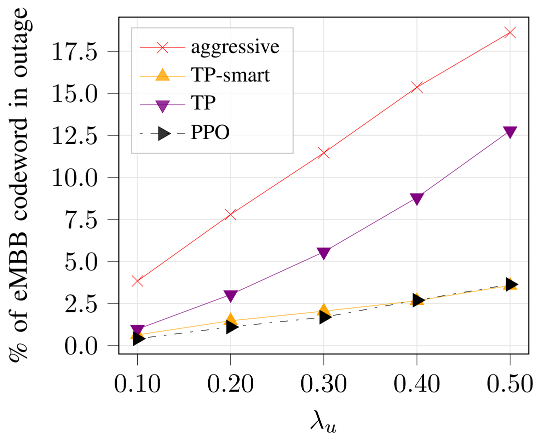

Fig. 4. Percentage of eMBB codeword in outage versus Poisson rate λ_u.

V. Conclusions

We proposed a DRL approach to train an agent acting as a scheduler able to dynamically manage the coexistence of the URLLC traffic on top of the eMBB traffic. The agent is trained using PPO, a state-of-the-art DRL algorithm, and once trained it can decide where to execute puncturing with average time complexity within the range of $[0.3, 3.0]$ milliseconds, depending on the underlying hardware. It also supports parallelization by means of autonomous decisions over multiple simulations at once. The trained RL agent overall outperforms all the other schemes on multiple performance metrics, being capable of noteworthy generalization over the complementary task of never violating URLLC latency requirements while minimizing eMBB codewords' outages. Our approach is highly scalable with respect to the length of each simulation and the arrival packet distribution, without retraining the agent. This is of critical importance since the real world task we modeled is inherently not episodic and the URLLC packets' arrival distribution is not known a priori or it could be subject to changes.

We believe this work to be a promising foundation w.r.t. the general research task of solving the eMBB-URLLC resource slicing problem. Our future works will build on top the results of this paper, retaining the approach and its scalability while we increase the realism of the simulation. Specifically, we plan to focus on: taking into account the reliability of URLLC user; addressing the transmission over multiple frequency resources; enabling non-orthogonal multiple access communication; testing the agent with different arrival packets distributions.

Acknowledgement

The work of Fabio Saggese and Marco Moretti is partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

References

[1] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G RAN slicing for verticals: Enablers and challenges," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 28–34, 2019.

[2] P. Popovski, K. Trillingsgaard, O. Simeone, and G. Durisi, "5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View," *IEEE Access*, vol. 6, pp. 55 765–55 779, 2018.

[3] C. She, C. Yang, and T. Q. S. Quek, "Radio Resource Management for Ultra-Reliable and Low-Latency Communications," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 72–78, 2017.

[4] A. Anand and G. de Veciana, "Resource Allocation and HARQ Optimization for URLLC Traffic in 5G Wireless Networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2411–2421, 2018.

[5] A. Anand, G. de Veciana, and S. Shakkottai, "Joint Scheduling of URLLC and eMBB Traffic in 5G Wireless Networks," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 477–490, 2020.

[6] J. Tang, B. Shim, and T. Q. S. Quek, "Service Multiplexing and Revenue Maximization in Sliced C-RAN Incorporated With URLLC and Multicast eMBB," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 4, pp. 881–895, 2019.

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

[8] M. Elsayed and M. Erol-Kantarci, "AI-Enabled Radio Resource Allocation in 5G for URLLC and eMBB Users," in *2019 IEEE 2nd 5G World Forum (5GWF)*, 2019, pp. 590–595.

[9] Y. Li, C. Hu, J. Wang, and M. Xu, "Optimization of URLLC and eMBB Multiplexing via Deep Reinforcement Learning," in *2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops)*, 2019, pp. 245–250.

[10] Y. Huang, S. Li, C. Li, Y. T. Hou, and W. Lou, "A Deep-Reinforcement-Learning-Based Approach to Dynamic eMBB/URLLC Multiplexing in 5G NR," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6439–6456, 2020.

[11] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," *IEEE Trans. Wireless Commun.*, pp. 1–1, 2021.

[12] P. He, L. Zhao, S. Zhou, and Z. Niu, "Water-Filling: A Geometric Approach and its Application to Solve Generalized Radio Resource Allocation Problems," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3637–3647, 2013.

[13] G. M. S. Rahman, M. Peng, K. Zhang, and S. Chen, "Radio Resource Allocation for Achieving Ultra-Low Latency in Fog Radio Access Networks," *IEEE Access*, vol. 6, pp. 17 442–17 454, 2018.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[15] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1889–1897.

[16] L. Pasqualini and F. Saggese, "Deep Reinforcement Learning for URLLC data management on top of scheduled eMBB traffic," GitHub repository, 2021, https://github.com/InsaneMonster/telerl2021.

[17] OpenAI, "Proximal Policy Optimization," OpenAI web site, 2018, https://spinningup.openai.com/en/latest/algorithms/ppo.html.

[18] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv:1506.02438*, 2015.