Efficient 5G Resource Block Scheduling Using Action Branching and Transformer Networks

Sylvain Nérondat*†, Xavier Leturc*, Philippe Ciblat†, Christophe J. Le Martret*

*Thales SIX GTS SAS, France
†LTCI, Télécom Paris, IP Paris, 91120 Palaiseau, France
firstname.lastname@{*thalesgroup.com, †telecom-paris.fr}

Abstract—**This paper presents a deep reinforcement learning-based scheduling solution tailored for 5G networks. The proposed neural network architecture, utilizing an encoder-only transformer and action branching, is designed to handle large action spaces for resource block allocation in wireless environments. By training on variable number of user equipment scenarios, the solution generalizes well across different configurations. Experimental results in Nokia's wireless suite environment demonstrate superior performance in packet loss, compared to heuristics.**

Index Terms—**Action branching, deep reinforcement learning, scheduling, transformer, wireless suite.**

I. Introduction

Packet scheduling is a critical mechanism in communication systems, determining how data packets are prioritized and transmitted to ensure efficient resource utilization, low latency, and high-quality service in increasingly demanding network environments.

Until recently, scheduling algorithms have relied on heuristics. These heuristics were designed to achieve specific objectives: round-robin (RR) aims to distribute resources equally across all links, proportional fair (PF) seeks to distribute rates fairly across all links, while the log-rule, exp-rule, and modified largest weighted delay first (MLWDF) aim to guarantee bounded delays. These heuristics operate with limited information on flows and buffers, such as: i) average or instantaneous data rate, ii) head of line (HoL) delay, or number of bits.

Currently, approaches utilizing neural network (NN) architectures trained with deep reinforcement learning (DRL) have emerged to tackle the scheduling challenge. A key advantage of NN architectures is their flexibility in incorporating numerous features, making it easier to expand features at the NN input compared to heuristics. Additionally, DRL is model-free, allowing for customized objectives through the reward function.

Each user equipment (UE) is assigned specific quality of service (QoS) associated with first in, first out (FIFO) buffers with finite size, in which the packet data flows are stored. Packets may be lost when new packets arrive and the buffer is full, a situation referred to as buffer overflow (BO). Each packet in the buffer is characterized by its age, i.e. the time spent in the buffer since its arrival, expressed in time transmission interval (TTI), which is the smallest unit of time for the scheduling process. According to the QoS, packets are assigned time duration constraints called packet delay budget (PDB). A packet exceeding the PDB is referred to as delay violation (DV). The corresponding packet is kept in the buffer until transmission. Each link from the base station (BS) to the UE is characterized by a link quality given by the channel quality indicator (CQI). The number of bits that can be scheduled is determined from the CQI.

NN architectures utilize input vectors from the state space that are defined by the designer and are crucial for system performance. Most schedulers in existing literature, whether heuristic or NN-based, primarily consider HoL, number of packets (NP), or both. An effective NN scheduler should, after training, possess these properties: i) it should be permutation equivariant (PE) with respect to the inputs, enhancing learning and inference performance [1], ii) it should adapt to a variable number of links, accommodating temporal fluctuations, a property called number of links independent (NLI), iii) it should collectively evaluate all users' buffer data for scheduling to achieve a global, potentially optimal solution, termed global buffer management (GBM). Note that while NLI can handle a varying number of links, good performance is not guaranteed. This depends on factors such as training methods, hyperparameters, datasets, and the architecture's generalizability, and thus requires validation.

The scheduling operation assigns UEs to a set of resource blocks (RBs) at each slot. Two approaches can be used for this assignment: i) RBs are assigned sequentially, one by one, or ii) RBs are assigned all at once. We refer to approach i) as sequential RB scheduling (SRS), and ii) as global RB scheduling (GRS). Approach i) can be seen as a local optimization, while approach ii) represents a global optimization. The solution proposed in this paper follows the GRS approach.

This paper proposes a novel NN architecture for a GRS packet scheduler trained with a deep Q-learning (DQL) algorithm [3]. Although GRS is a joint approach (thus presumably better than SRS), it suffers from the *curse of dimensionality*, as the number of possible actions grows exponentially with the number of RBs, rendering this approach inapplicable with conventional NN architectures. To mitigate this issue, we propose using the action branching (AB) architecture concept [4], which involves i) implementing a NN fed by the state vectors, and ii) distributing the output of the first NN to N_f parallel branches, each performing dedicated processing, where N_f is the number of RBs. The output of each branch gives the Q-values of the possible actions (number of UEs), from which we deduce the index of the scheduled UE using an argmax. For part i), we use an encoder-only transformer (EOT) architecture that possesses the three aforementioned properties: PE, NLI, and GBM.

The proposed solution is evaluated using Nokia's wireless suite (WS) environment, which implements a simplified version of the 5G downlink scheduling mechanism and is available online [2]. The objective of this paper is to experiment with the new EOT-AB architecture using the WS environment as is, without modifying the framework, and in particular the state space information and reward.

The main contributions of this paper are: i) the proposal of a DRL scheduling solution performing a GRS approach, based on an AB architecture, and using an EOT for the first NN, ii) the development of a masking procedure adapted to the AB architecture, and iii) the performance evaluation of the proposed architecture, and comparison with three heuristics implemented in the WS environment, in terms of lost packets and packet delays.

The rest of this paper is organized as follows. Section II provides an analysis of the state of the art (SotA). Section III depicts the system model. Section IV describes the proposed solution. Section V presents and analyzes the numerical results. Section VI draws concluding remarks.

II. State of the art solutions with DRL

We analyze the SotA of solutions based on DRL aiming at scheduling several RBs. Regarding the GRS, one can find two kinds of approaches: i) the architecture outputs give the indices of the UEs to be scheduled for each RB, ii) the architecture outputs give a proportion of RBs to be allocated per UEs. In that case, a non-linear post-processing is needed to convert the proportions into the number of UEs per RB. In the following, we refer approach i) to as GRS index (GRS-I) and ii) to as GRS proportion (GRS-P). Notice that due to the non-linear post-processing in the GRS-P approach, the proportions cannot be exactly respected, thus we conjecture that this approach is suboptimal compared to the GRS-I one.

The NLI, PE, and GBM properties are deduced from the NN architecture characteristics and the formatting of input state vectors. The state vectors feeding the NN architecture can be: i) a single vector stacking all the UEs' states, or ii) a matrix where columns represent individual UE state vectors. We refer to i) as vector state input (VSI) and ii) as matrix state input (MSI). NN architectures in the literature are either fully connected (FC) or architectures involving memory mechanism like long-short term memory (LSTM). Regarding the NLI property, the VSI format does not fulfill it because the input vector size is fixed, depending on the number of UEs, whereas MSI one is inherently NLI. For the PE property, FC combined with MSI achieves it, as reordering the input vectors produces the same outputs, and architectures like EOT without positional encoding also fulfill this. The GBM property holds in the VSI format since it encompasses all UE features. The MSI format, along with a FC, cannot be GBM because state input vectors the processing are processed piecewise, whereas an architecture with memory fulfills it.

Following the classification presented in the previous paragraph, we summarize the analysis of the publications identified in Table I based on the following features: i) the allocation approach, SRS, GRS-P or GRS-I, ii) the architecture property, NLI, PE, and GBM. In [5], [6], authors use an SRS approach, while in [7], [8], they use the GRS-P approach, and in [9], [10] the GRS-I is employed. It is worth noting that none of the identified references verify the three properties, except for the solution proposed in this paper.

TABLE I: Summary of SotA scheduling solutions with DRL.

Ref.	Allocation mode	NLI	PE	GBM
[5]	SRS			✓
[6]	SRS	✓		✓
[7]	GRS-P	✓	✓	
[8]	GRS-P			✓
[9]	GRS-I	✓	✓	
[10]	GRS-I	✓		✓
Ours	GRS-I	✓	✓	✓

III. System model

We implement our architecture within the Nokia's WS environment that is suited for reinforcement learning (RL) algorithms evaluation since its implementation follows the conventional *state, action, reward* of the RL scheme [11]. We focus in this paper on the *TimeFreqResourceAllocation-v0* (TFRA) challenge presented in [2], and referred to as WS-TFRA in the sequel. In the following sections, we describe the WS-TFRA environment and the corresponding communication model, followed by the implemented RL model.

A. WS environment and communications model

WS-TFRA implements a BS designed to send traffic to K UEs, which are uniformly distributed within a 1000-meter square around it. The UEs move according to a random walk at constant speeds, according to a normal distribution. Free space propagation is assumed along with shadowing. The total bandwidth W is divided into N_f RBs, which are allocated by the scheduler to UEs at each TTI. The data to be scheduled to the different UEs are stored in finite length buffers at the BS, each buffer containing at most B packets. There are two kinds of QoS classes: guaranteed bit rate (GBR) and non-GBR, and four types of traffic: voice, video, delay critical (DC), and web, each with a specific QoS. Voice, video, and DC are GBR, while web is non-GBR. WS-TFRA ensures that the traffic types are equally represented, which requires that the number of UEs is a multiple integer of four, i.e. $K = 4p$, with $p \in \mathbb{N}^*$. In the following, our developments are presented in a more general manner for possible future extensions, considering that the number of QoS levels is equal to n_Q, while keeping the constraint $K = pn_Q$. Each UE is assigned a single service, which is uniformly randomly drawn from $\mathcal{Q} := \{1, \dots, n_Q\}$. The traffic characteristics are defined by the inter-arrival time τ_q, i.e. the time between the arrival of two consecutive packets in the buffer, and the incoming packet size $\mathcal{B}_q$, where $q \in \mathcal{Q}$. The QoS corresponding to traffic q is given by the PDB D_q.

The scheduling sequence is depicted as follows. At each TTI t, the scheduler selects the different UEs to be served on the N_f RBs of the slot. To do so, the BS selects one UE per RB, with the possibility of allocating multiple RBs to the same UE. The number of bits delivered by the BS to each selected UE depends on the CQI index $n_k^c \in \{0, \dots, n_{\text{CQI}}\}$ with $n_{\text{CQI}} = 15$, where 0 represents the worst channel quality, and 15 represents the best. Depending on the number of allocated RBs, the CQI and the packet sizes, a packet may be not completely transmitted, resulting in partial packet transmission. The age of a packet continues to be incremented as long as it is not fully transmitted. Once the bits have been extracted from the buffers, the age of the packets is incremented. New packets arrive in the buffers according to τ_{q_k} and $\mathcal{B}_{q_k}$, which depends on the traffic type of UE k. If a new packet arrives while the corresponding buffer is full, a BO occurs, and the packet is lost. Finally, the UEs move, and the sequence is repeated at each TTI.

B. Reinforcement learning model

In this section, we describe the state, action, and reward used in our architecture, which are derived from those of WS-TFRA. Several *adaptations* are required to convert the outputs of WS-TFRA to fit our architecture: i) going from sequential to joint RB allocation, and ii) adapting the state vector format to the MSI format.

Next, for the state, action, and reward, we first review the outputs of WS-TFRA, followed by a description of the proposed adaptations.

1) State space: The state space implemented in WS is the concatenation of the features associated with each buffer and the RB index in the slot. Each packet j of UE k is characterized by its number of bits $b_{j,k}$ and its age $d_{j,k}$. The feature associated with the buffer of UE k for RB n during time slot t is denoted by $\mathbf{f}_{k,t}^n$ and is defined as:

$$\mathbf{f}_{k,t}^n := [\mathbf{b}_k, \mathbf{d}_k, n_k^c, \mathbf{h}_k]^T, \tag{1}$$

where T is the transposition operator, $\mathbf{b}_k := [b_{1,k}, \dots, b_{B,k}]$ and $\mathbf{d}_k := [d_{1,k}, \dots, d_{B,k}]$ are the vectors containing the number of bits and the age of each packet, respectively. Let $q_k \in \mathcal{Q}$ be the QoS of UE k and $\mathbf{h}_k$ its corresponding one-hot encoded vector, i.e. $\mathbf{h}_k$ is a vector with all entries equal to zero except the q_kth entry, which is equal to one. The number of entries in the vector $\mathbf{f}_{k,t}^n$ is equal to $(2B + n_Q + 1)$. The resulting state provided by the WS-TFRA at RB n of slot t writes

$$\mathbf{s}_{t,WS}^n := [\mathbf{f}_{1,t}^{n\ T}, \dots, \mathbf{f}_{K,t}^{n\ T}, n]^T, \tag{2}$$

The number of entries of vector $\mathbf{s}_{t,WS}^n$ is equal to $K(2B + n_Q + 1) + 1$.

Proposed adaptations. Since we propose a joint UE allocation for all the RBs in the slot, we only need to consider the state at the beginning of the time slot, i.e., $\mathbf{f}_{k,t}^1$. In addition, the state vector as defined in (1) was found not to achieve very good performance. Therefore, we propose a new one:

$$\tilde{\mathbf{f}}_{k,t}^1 = [\mathbf{h}_k \otimes [\mathbf{b}_k / \mathcal{B}_{q_k,th}, \mathbf{d}_k / D_{q_k}], n_k^c / 15]^T \tag{3}$$

where $\otimes$ represents the Kronecker product, and $\mathcal{B}_{q_k,th}$ and D_{q_k} are the maximum values allowed for $b_{j,k}$ and $d_{j,k}$ respectively. The length of vector $\tilde{\mathbf{f}}_{k,t}^1$ is equal to $(2Bn_Q + 1)$. This new definition considers the same entries as in (1), but arranged differently by inserting zeros, and normalized to 1. Moreover, since we use a NLI architecture that can handle a variable number of UEs, we need MSI inputs. Thus, we have to reshape (2) with $n = 1$, and replacing $\mathbf{f}_{k,t}^1$ by $\tilde{\mathbf{f}}_{k,t}^1$, into a matrix where each column correspond to $\tilde{\mathbf{f}}_{k,t}^1$:

$$\mathbf{s}_{t,a} := [\tilde{\mathbf{f}}_{1,t}^1, \dots, \tilde{\mathbf{f}}_{K,t}^1] \tag{4}$$

which has dimensions $(2Bn_Q + 1) \times K$.

2) Action: At RB n in slot t, WS-TFRA takes as input a scalar action:

$$a_{t,WS}^n = i \tag{5}$$

where i is the index of the buffer to be scheduled.

Proposed adaptations. The joint RB allocation requires to adapt the action format to be taken as:

$$\mathbf{a}_{t,a} := (i_t^1, \dots, i_t^{N_f}) \tag{6}$$

where i_t^j corresponds to the buffer to be scheduled during the jth RB of slot t.

It is worth noting that the proposed modification results in an action space with a dimension of K^{N_f}. For instance, in WS, where $N_f = 25$, setting $K = 32$, we obtain $K^{N_f} > 10^{37}$, which confirms the curse of dimensionality discussed in Section I, and highlights the need for the AB architecture to mitigate it.

3) Reward: The reward implemented in WS is given by:

$$r_{t,WS}^j = \begin{cases} 0, & \text{if } j < N_f \\ -n_{d,t} - n_{b,t}, & \text{if } j = N_f \end{cases} \tag{7}$$

where $n_{d,t}$ and $n_{b,t}$ denote the total number of bits subject to DV in the buffers and the number of bits in the non-GBR buffers, respectively. The reward is computed after the RB allocation and before new packets arrive. The objective of this reward is thus to minimize the number of bits in the buffers subject to DV for all traffic types, plus the number of bits in the non-GBR buffers.

Proposed adaptations. For the GRS-I approach, we propose to use the reward (7) at $n = N_f$. In addition, since the WS-TFRA reward is a non-positive integer, we propose mapping it to the range $[0, 1]$ using the exponential function:

$$r_{t,a} = e^{r_{t,WS}^{N_f}}. \tag{8}$$

IV. Proposed solution

The scheduling problem, with the action space, state space and reward defined in Section III, can be modeled as a Markovian decision process (MDP), not presented here due to space limitation. In that context, the DQL is known to find the optimal policy, and thus the optimal scheduler, which maximizes the long-term average discounted reward [11].

We now expose the proposed NN architecture, and then present the implementation of the masking mechanism.

A. NN architecture

The proposed architecture follows the AB architecture philosophy as described in [4] along with the use of an EOT NN, referred to as EOT-AB in the sequel, and illustrated in Fig. 1.

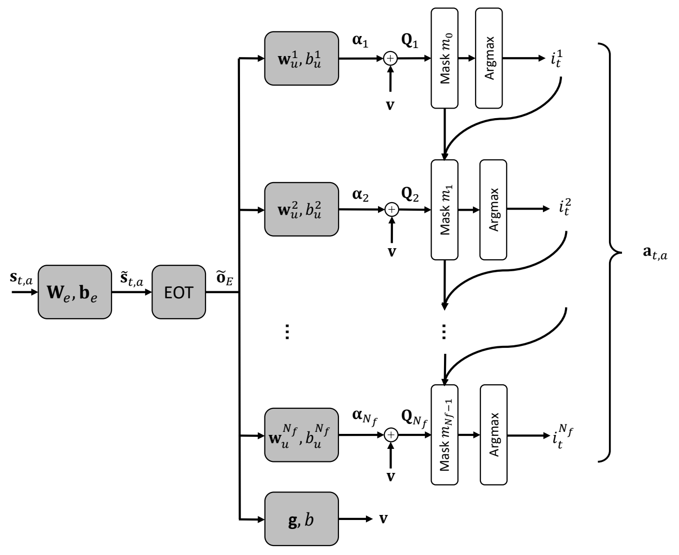


Fig. 1: Proposed EOT-AB architecture combining an EOT and an AB structure. Gray blocks are learned.

First, the state space, $\mathbf{s}_{t,a}$, undergoes an affine embedding, $\tilde{\mathbf{s}}_{t,a} := \mathbf{W}_e\mathbf{s}_{t,a} + \mathbf{b}_e$, where $\mathbf{W}_e$ is a matrix with dimensions $d_e \times (2Bn_q + 1)$, and $\mathbf{b}_e$ is a vector of dimensions $d_e \times 1$. Next, $\tilde{\mathbf{s}}_{t,a}$ is processed by an EOT module, without positional encoding [12], which has the three following properties: NLI, GBM, and PE. The implementation details for the EOT can be found in [13], and it is available in conventional machine learning frameworks like PyTorch. The EOT output is a $d_e \times K$ matrix, denoted by $\tilde{\mathbf{o}}_E$. In a typical EOT setting, the output would pass through an affine unembedding. To integrate the sequence of an EOT with the AB architecture, we propose to distribute the unembeddings across each branch. For jth branch, the affine unembedding is a $1 \times K$ vector that writes $\boldsymbol{\alpha}_j := \mathbf{w}_u^j\tilde{\mathbf{o}}_E + b_u^j$, where $\mathbf{w}_u^j$ is a $1 \times d_e$ vector, and b_u^j is a scalar. The kth entry of $\boldsymbol{\alpha}_j$ represents the advantage function of action k for RB j [4]. Additionally, following [4], dueling is applied to estimate the value function represented by a $1 \times K$ vector that writes $\mathbf{v} = \mathbf{g}\tilde{\mathbf{o}}_E + b$, where $\mathbf{g}$ is a $1 \times K$ vector, and b is a scalar. The kth entry of $\mathbf{v}$ represents the estimated value for UE k. Vector $\mathbf{v}$ is then added on each branch to $\boldsymbol{\alpha}_j$ to produce $\mathbf{Q}_j$, an estimate of the Q-value for each action. To prevent suboptimal decisions due to selection of an empty buffer, action masking [14] is applied on each branch. Detailed mask implementation is provided in Section IV-B. Finally, the action i_t^j is selected by taking the argmax on the masked vector $\mathbf{Q}_j$. As highlighted in gray in Fig. 1, $\mathbf{W}_e$, $\mathbf{b}_e$, EOT, $\mathbf{w}_u^j$, b_u^j, $\mathbf{g}$, and b are learned during the training phase.

B. Action masking procedure

The action masking procedure [14] aims to prevent the selection of "invalid" actions, i.e. actions that are not possible or allowed. In [14], authors i) theoretically validate this approach, ii) demonstrate that action masking improve training convergence and inference performance compared to more conventional methods, such as *invalid action penalty*. In the scheduling problem, there is no invalid actions, but suboptimal ones when an empty buffer is selected. Here, we propose using action masking for empty buffers. Once the actions $\mathbf{a}_{t,a}$ have been determined, the RBs are filled with the bits taken from the selected buffers. During this process, no new bit arrives, the buffer contents evolve only due to bits extraction. The RBs are filled sequentially, and for each RB, the bits are extracted from the UE buffer (the selection order of the RBs is not significant). As a result, a UE may be selected multiple times, reflected by multiple occurrences of the same index in $\mathbf{a}_{t,a}$.

Two situations can lead to the selection of an empty buffer: i) the first time a UE is selected when its buffer is already empty, ii) a UE is selected multiple times, eventually emptying the buffer. Therefore, we need to track each buffer's status (empty or not) after each RB allocation, and, when an empty buffer is detected, set its mask to empty. This sequential adaptive masking procedure, to the best of our knowledge, is not covered in the literature and is illustrated in Fig. 1. In this figure, the RBs are selected sequentially from RB #1 to RB #N_f (top to bottom). The initial mask, m_0, is set based on the buffer statuses at the beginning of the slot. For RB #1, the buffer i_t^1 is selected among the non-empty buffers, thanks to the mask m_0, which is applied to the Q-values in $\mathbf{Q}_1$. After bits are extracted from buffer i_t^1, we update the mask if the buffer becomes empty. This corresponds to the curved arrow pointing from i_t^1. We then update the mask, resulting into m_1, apply m_1 to the second branch, and repeat the process until the last branch.

V. Numerical results

A. Communication setup

We use the default parameters of the WS-TFRA: a bandwidth of $W = 5$ MHz, $N_f = 25$ RBs, and $B = 8$. The parameters of the four traffics and QoS parameter D_q are reported in Tab. II. For GBR traffics, both $\mathcal{B}_q$ and τ_q are fixed, while for the non-GBR traffic, they are drawn according to a geometric distribution $\mathcal{G}$:

$$\mathcal{B}_4 = \min\left(\max\left(1, \mathcal{G}(1/20\,000)\right), 41\,250\right) \tag{9}$$

$$\tau_4 \sim \mathcal{G}(1/\beta) \tag{10}$$

with $\beta = 10$, the default parameter in WS-TFRA.

TABLE II: QoS and traffic parameters.

Service	QoS class	q	D_q	$\mathcal{B}_q$	τ_q
Voice	GBR	1	100	584	20
Video	GBR	2	150	41 250	33
DC	GBR	3	30	200	20
Web	non-GBR	4	300	Eq. (9)	Eq. (10)

TABLE III: Lost packets due to BO and DV wrt K.

K	32	36	40
PF	1816	18 650	143 048
BA	19 751	90 955	493 521
KP	3	5	200
EOT-AB	**0**	**3**	**42**

B. Training and inference setup and benchmark algorithms

The proposed scheduler is trained over 2000 episodes of 2622 steps each (i.e. 65 536 RBs) with a fixed $\beta = 10$. The number of UEs varies from episode to episode in $\mathcal{K} := \{32, 36, 40\}$. Our objective is to train a single NN achieving good performance across different values of K, avoiding the need to implement a dedicated NN for each possible K. The inference is conducted over 1000 episodes of 2622 steps for each $K \in \mathcal{K}$.

We compare the performance of the proposed approach with the heuristics implemented in WS and optimized for WS-TFRA: i) *ProportionalFairChannelAwareAgent*, which is a PF algorithm, ii) *BoschAgent*, which is described in [15] and referred to as Bosch agent (BA) hereafter, and iii) *Knapsackagent* an algorithm called knapsack (KP), for which [2] provides no bibliographic reference. Notice that we were unable to compare our approach with those listed in Tab. I due to insufficient information in the papers to implement them.

C. Performance metrics

We compare the solutions in terms of the following performance metrics, collected at the end of the inference episodes: i) the number of packets exceeding their PDB, ii) the number of packets lost due to BO, and iii) the time spent by a packet in the buffer between arrival and departure in TTI, referred to as packet delay (PD) in the sequel. Since the age of a packet is determined by the age of the last transmitted bit: i) a packet is said to exceed its PDB as soon as, at least one bit of the packet exceeds the PDB, ii) the PD is evaluated as the number of slots between the arrival of the packet in the buffer and the departure of the last bit of this packet from the buffer.

D. Performance results

1) Inference performance on the training setup: We set $\beta = 10$ as during training, evaluate the inference performance of the learned NN for $K \in \mathcal{K}$, and compare it with that of the heuristics. Table III provides the number of packets lost due to BO and DV with respect to (wrt) K. Several observations can be made: i) the PF performs better than BA, but still loses a significant number of packets, ii) both KP and EOT-AB outperform BA and PF, iii) EOT-AB outperforms KP for all values of K, achieving zero packet loss for $K = 32$, while KP has a loss of 3, iv) for $K = 36$, performance is similar, with a slight advantage for EOT-AB, v) for $K = 40$, KP loses almost five times more packets than EOT-AB. In conclusion, EOT-AB outperforms all the heuristics.

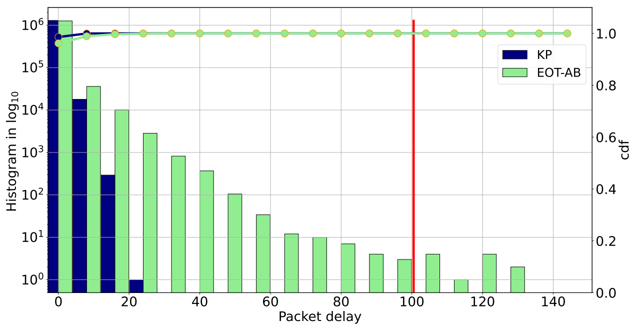
Fig. 2 depicts both the cumulative distribution function (cdf) and the histograms of the PD for each service, for KP and EOT-AB, only for $K = 40$ due to space limitation. The other heuristics are not displayed since they yield significantly worse performance. The histograms are plotted on the $\log_{10}$ scale because of the high dynamic range of the values, allowing both high and low numbers of packets to be displayed simultaneously. The red vertical line shows the PDB value. The cdf shows the amount of received packets vs. the packet delay. From the histograms, we observe that: i) some voice and DC packets exceed their PDB when using EOT-AB, while no packet exceeds the PDB for KP, ii) the PD spread for the GBR traffics is larger with EOT-AB than with KP, iii) the PD spread for non-GBR traffic is identical for both EOT-AB and KP. The cdf shows that, for non-GBR traffic, although the PD spread is similar, the EOT-AB cdf increases faster than the KP cdf. Specifically, 90% of packets are received within 7.6 TTIs for EOT-AB, whereas it takes 22 TTIs for KP to achieve the same result, which is approximately three times longer.

These observations can be explained by the design of the reward (7). Indeed, $n_{d,t}$ represents the number of packets in buffers exceeding the PDB for all traffic types, while $n_{b,t}$ represents the number of packets in buffers that do not exceed the PDB, only for non-GBR traffic. As a consequence, $n_{b,t}$ incentivized EOT-AB to empty the non-GBR buffers, whereas KP lacks such an incentive. Moreover, for $\beta = 10$, the non-GBR traffic has the highest bit load among all services, which reinforces the bias in favor of the non-GBR traffic at the expense of GBR traffic types.

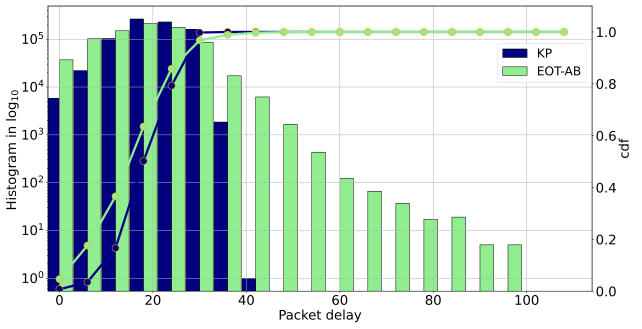
2) Generalization vs. β : We evaluate here the generalization capability of EOT-AB and KP vs. β. The metric considered here is the number of packets contributing to the packet error rate, as specified in the 5G standard, i.e., the sum of the number of packets lost due to BO and the ones exceeding the PDB for the DC traffic. We call this metric *PER lost packets*. Fig. 3 plots the number of lost packets wrt β in the range $[7.5, 10]$, the lower β, the higher the arrival rate. The metric is averaged over 1000 inference episodes, only for $K = 32$ due to space limitation. One can observe that EOT-AB shows a slight increase of lost packets for $\beta = 7.5$, whereas KP loses progressively more packets as β decreases, reaching ten times more lost packets at $\beta = 7.5$. This demonstrates the good generalization capability of the proposed approach in more stringent conditions.

VI. Conclusion

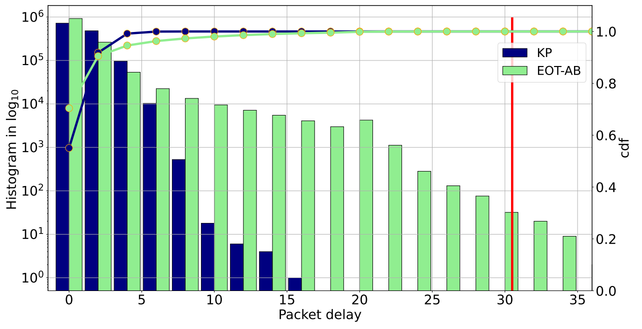
We have proposed in this paper a new architecture for packet scheduling suitable to the 5G context. The proposed solution is based on an AB architecture capable to handle the very large action space induced by the joint RB allocation, and



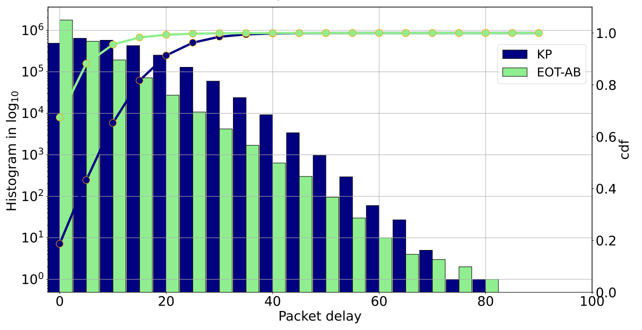
(a) Voice (GBR).



(b) Video (GBR).



(c) Delay critical (GBR).



(d) Web (non-GBR).

Fig. 2: CDF and histogram of packet delay for the different services for $K = 40$.

leverages the EOT architecture to handle a varying number of UEs, and is in addition GBM and PE. The proposed solution has been tailored to the WS-TFRA, using the environment outputs. Simulations in this framework show that the EOT-AB outperforms the heuristics implemented in WS in terms of number of lost packets, which is encouraging. However, it fails

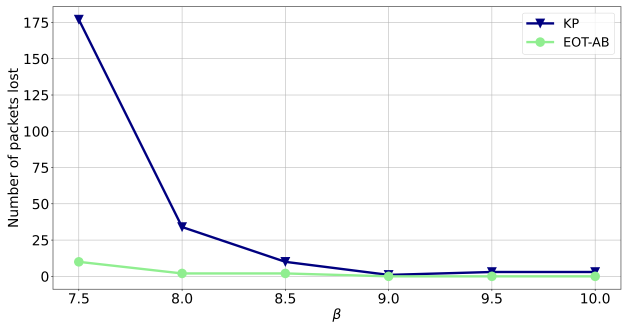


Fig. 3: Total number of PER lost packets vs. β for $K = 32$.

to outperform KP for the GBR traffics in terms of PD. This is because the WS reward is actually not well-suited to DRL architectures, and in particular to EOT-AB, to accommodate all traffic types, and will thus need to be adapted in future work.

References

[1] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, “Deep sets,” *Advances in neural information processing systems*, vol. 30, 2017.

[2] A. Valcarce. Wireless suite: A collection of problems in wireless telecommunications. [Online]. Available: https://github.com/nokia/wireless-suite

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] A. Tavakoli, F. Pardo, and P. Kormushev, “Action branching architectures for deep reinforcement learning,” in *Proceedings of the aaai conference on artificial intelligence*, vol. 32, no. 1, 2018.

[5] E.-M. Bansbach, V. Eliachevitch, and L. Schmalen, “Deep reinforcement learning for wireless resource allocation using buffer state information,” in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[6] F. Al-Tam, A. Mazayev, N. Correia, and J. Rodriguez, “Radio resource scheduling with deep pointer networks and reinforcement learning,” in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2020.

[7] A. Paz-Pérez, A. Tato, J. J. Escudero-Garzás, and F. Gómez-Cuba, “Flexible reinforcement learning scheduler for 5G networks,” in *2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*, 2024, pp. 566–572.

[8] Z. Gu, C. She, W. Hardjawana, S. Lumb, D. McKechnie, T. Essery, and B. Vucetic, “Knowledge-assisted deep reinforcement learning in 5G scheduler design: From theoretical framework to implementation,” *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, 2021.

[9] J. S. Shekhawat, R. Agrawal, K. G. Shenoy, and R. Shashidhara, “A reinforcement learning framework for QoS-driven radio resource scheduler,” in *IEEE Global Communications Conference (GLOBECOM)*, 2020.

[10] A. Robinson and T. Kunz, “Downlink scheduling in LTE with deep reinforcement learning, LSTMs and pointers,” in *IEEE Military Communications Conference (MILCOM)*, 2021.

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

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[13] M. Phuong and M. Hutter, “Formal algorithms for transformers,” *arXiv preprint arXiv:2207.09238*, 2022.

[14] S. Huang and S. Ontañón, “A closer look at invalid action masking in policy gradient algorithms,” *arXiv preprint arXiv:2006.14171*, 2020.

[15] P. M. de Sant Ana and N. Marchenko, “Radio access scheduling using CMA-ES for optimized QoS in wireless networks,” in *2020 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2020, pp. 1–6.