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

Fabio Saggese[1], Luca Pasqualini[2], Marco Moretti[1], Andrea Abrardo[2]

[1]Dip. di Ingegneria dell'Informazione, Università di Pisa
[2]Dip. di Ingegneria dell'Informazione e Scienze Mathematiche, Università di Siena

GTTI workshop "Wireless Intelligence: From Reconfigurable Surfaces to Edge/Cloud Communication"

UNIVERSITÀ DI PISA

UNIVERSITÀ DI SIENA 1240

Outline

Aim

- Finding a solution for the RAN slicing of eMBB and URLLC traffics;
- The solution should exploit the dynamic nature of URLLC traffic.

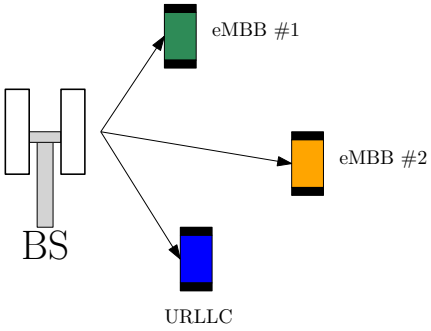
Why Reinforcement Learning?

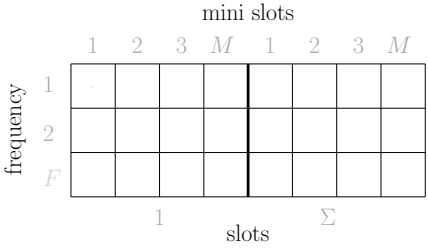
- Performing slicing of different type of traffic is a hard task;
- RL is able to find very good policies for systems that dynamically change through time;
- differently from other machine learning techniques, RL does not require collected data for training.

Outline

Scenario & Resources

We consider a downlink transmission for single cell scenario with two set of users: *eMBB* and *URLLC* users.



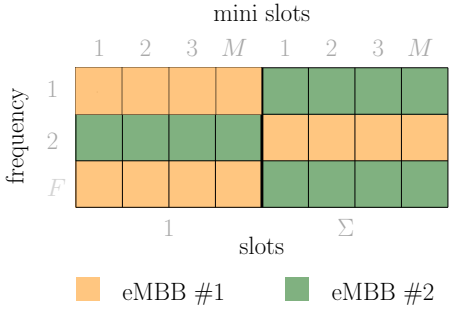


- F frequency resources;
- time resources are divided in Σ time slots;
- each time slot is divided in M mini slots.

enhanced Mobile BroadBand (eMBB)

Specifications:

- high throughput;
- no latency requirement;
- resource allocation is made on a *time slot* basis;
- perfect knowledge of channel state information (CSI) at the BS.



eMBB resource allocation can be performed following conventional methods (e.g.water-filling, wMMSE, etc.)

Ultra Reliable Low-Latency Communication (URLLC)

Specifications:

- low outage probability ($\leq 10^{-5}$);
- stringent latency requirement (≤ 1 ms);
- resource allocation is made on a *mini slot* basis

In our setting, each packet:

- has length equal to a mini slot;
- is generated in each mini slot following a Bernoulli with probability p_u;
- has a strict latency requirement $l_u^{\max}$;
- is stored in a in FIFO queue Q of infinite length;

URLLC and eMBB coexistence

eMBB codewords:

- the BS distributes di erent codeword for each user;
- codeword length is multiple of a mini slot;
- each codeword is protected by a code able to recover D erased mini slots.

URLLC puncturing:

- each URLLC packet is transmitted through puncturing selecting a time-freq resource;
- a transmitted URLLC packet is always successfully decoded

Outline

RL in a nutshell

Reinforcement Learning

Reinforcement Learning is a eld of Machine Learning studying the behaviour (policy) of a certain agent (model) acting in an environment (in which Markov property holds).

The agent accumulates a discounted return $G_t = \sum_{k=0}^{1} {}^{k} R_{t+k+1}$ with 2 [0; 1);

The probability distribution of G_t depends on the policy (ajs) that chooses the action a for each possible states;

The objective of RL is to nd a policy that maximizes the expected discounted return.

Model as a Markov Decision Process

Each step t of the simulation is a mini slot of the allocation grid.

Actions: $A_t = \{0, 1, \ldots, F\}$, where 0 means no transmission, while otherwise the action indicates the FR index.

State: $S_t = \{S_t^{(u)}, S_t^{(e)}\}$ where

$S_t^{(u)} = \{Q_t, \ldots_t\}$

$S_t^{(e)} = \{s_t(f)\}_{f=1}^{F}$ tracking how much the eMBB codeword on (t, f) is protected.

Reward: $R_t = \sum e_t(w) + L_t$

$$e_t(w) = \begin{cases} 1, & \text{if } A_t \text{ causes the outage of } w, \\ 0, & \text{otherwise,} \end{cases} \qquad L_t = \begin{cases} 0, & \ldots_t \quad 0, \\ \frac{3T}{F+1}, & \ldots_t < 0. \end{cases}$$

PPO and NN Architecture

Proximal Policy Optimization (PPO):

aims to the biggest possible improvement step on a policy without ending too far from the starting point one, thus avoiding the risk of performance collapse;

is an actor-critic algorithm ! two di erent neural networks are required.

NN architecture:

Two completely separated subnetworks:

value function: three ReLU dense layers with 128, 64, and 32 neurons + fourth with single neuron to estimate the value with no activation;

policy function: three ReLU dense layers with 128, 64, and 32 neurons + dense fourth layer with $F + 1$ neurons to choose the actions with softmax activation.

Outline

Simulation parameters

Resource grid: $F = 12$, 10 time slot, $M = 14$ mini slot each ! $T = 140$;

Users: 1 URLLC user, 10 eMBB users;

Maximum delay constraint to $I_u^{max} = 7$;

Degree of protection of eMBB codewords: $D \in \{0, 1\}$.

Learning phase

the parameters related to eMMB resource allocation and URLLC tra c generation are randomized on an episode basis: random allocation, codeword placement, protection of each codeword, and probability of URLLC packet.

We initialize each episode with a random number of URLLC packets in the queue (always smaller than I_u^{max}).

Comparison schemes

Aggressive The URLLC packet is transmitted immediately on a randomly chosen frequency.

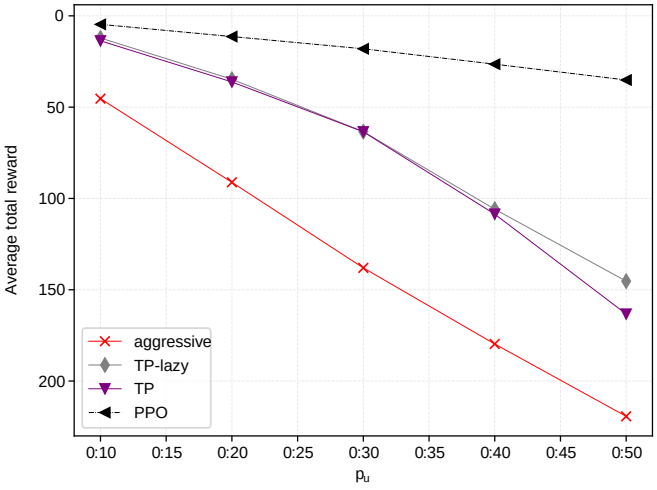
Threshold Proportional (TP) . The URLLC packet is transmitted immediately on the frequency resource occupied by the codeword with the highest puncturing threshold[1].

TP-lazy. As long as $_t > 0$, the packet is transmitted only if the present state is somehow better (or equal) than the next one. If $_t = 0$, the transmission is forced in the present mini slot. In any case, the choice of the frequency is made according to the TP scheme.

[1] A. Anand, G. de Veciana, and S. Shakkottai. Joint Scheduling of URLLC and eMBB Tra c in 5G Wireless Networks . In: IEEE/ACM Transactions on Networking 28.2 (2020), pp. 477 490 .

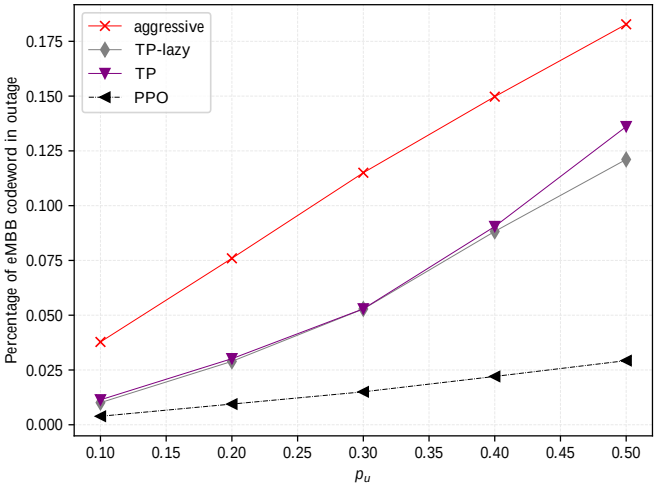
Average total reward, T = 140

Average total reward, $T = 1400$

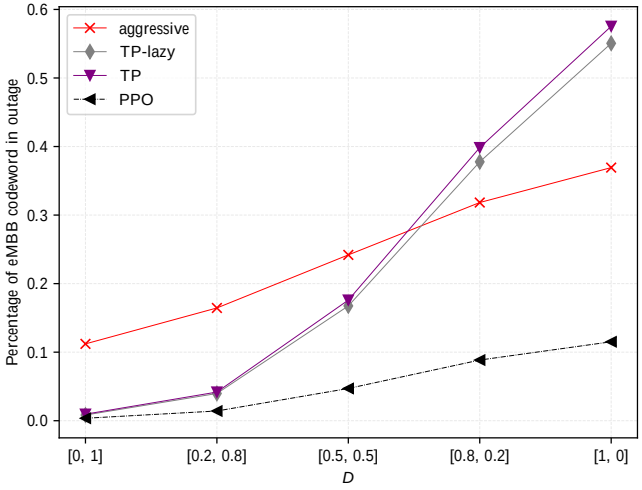


The same NN trained for $T = 140$ is used here, proving the generalization capacity of the agent.

eMBB codeword in outage vs p_u, $T = 1400$



eMBB codeword in outage vs degree of protection



Outline

Conclusions

- We proposed a Deep Reinforcement Learning approach to the slicing task with respect to eMBB and URLLC traffic;
- Our agent learns from scratch a policy outperforming all the man-made schemes over multiple performance metrics;
- The agent's learned policy is agnostic to the length of each episode allowing for fast self-training while still being applicable to the real world task.

Future works

- Taking into account the reliability of the URLLC user;
- Adopting a Poissonian distribution to better simulate the generation of URLLC packets;
- the agent should be able to transmit more than one packet over multiple frequencies;
- Enabling the Power Domain Non-Orthogonal Multiple Access (NOMA) communication.

Bibliography & Resources

- **Paper**: Fabio Saggese, Luca Pasqualini, Marco Moretti, and Andrea Abrardo. *Deep Reinforcement Learning for URLLC data management on top of scheduled eMBB tra c*. 2021. arXiv: 2103.01801 [eess.SP]
- **Github**: `https://github.com/InsaneMonster/telerl2021`
- **Framework**: `https://github.com/InsaneMonster/USienaRL`