NOMA-based scheduling and offloading for energy harvesting devices using Reinforcement Learning

Ibrahim Djemai*, Mireille Sarkiss*, Philippe Ciblat†
*SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 91120 Palaiseau, France
{ibrahim.djemai, mireille.sarkiss}@telecom-sudparis.eu
†LTCI, Télécom Paris, Institut Polytechnique de Paris, 91120 Palaiseau, France
{philippe.ciblat}@telecom-paris.fr

***Abstract*—We consider a joint optimization problem of resource scheduling and computation offloading in a Mobile-Edge Computing (MEC) system where User Equipments (UEs) or devices have energy harvesting functionalities. The UEs can either execute locally the data packets or offload them to a nearby MEC server for remote processing. The main objective is to minimize the overall packet losses of the UEs under strict delay constraints imposed by applications. Non-Orthogonal Multiple Access is enabled to allow UEs sending their data packets simultaneously. The problem is formulated as a Markov Decision Process and is solved using Proximal Policy Optimization, a Deep Reinforcement Learning algorithm. The numerical results show the efficiency of such an algorithm in reducing the packet loss as well as the energy consumed during testing compared to some naive heuristics.**

***Index Terms*—Scheduling, Offloading, Energy Harvesting, NOMA, PPO**

I. Introduction

Sequential Decision Making refers to well-established techniques to develop policies that optimize the behavior of a given agent in a stochastic environment by maximizing rewards (or minimizing costs). More specifically, Reinforcement Learning (RL) has shown great promise to resolve complex tasks in large environments and obtain good approximations of optimal policies by a trial-and-error approach. The integration of deep neural networks with RL has produced a suite of Deep Reinforcement Learning (DRL) algorithms with even more scalability and a wide range of applications.

In this paper, we focus on using DRL techniques to investigate joint scheduling and offloading in Mobile Edge Computing (MEC) systems under strict delay constraints. We aim at finding optimal policies to process users' data by minimizing data loss while allowing Non-Orthogonal Multiple Access (NOMA) technique and Energy Harvesting (EH) capability at User Equipments (UEs). On one hand, benefiting from the proximity of MEC servers at the base stations allows users to offload their heavy tasks to be remotely processed, extending thus the computational capacity at end-devices. On the other hand, harvesting energy from external sources promises to reduce carbon fingerprint of current IoT devices and mobile networks. The collected energy is stored in batteries and can serve to power network operations, improving hence devices lifetime. Moreover, enabling simultaneous access of users to wireless channels via NOMA provides higher spectral efficiency than Orthogonal Multiple Access (OMA) techniques, satisfying thus high date-rate and latency requirements of future networks but participates to the curse of dimensionality which can reduce the aforementioned advantages.

The goal of our work is then to leverage on RL techniques to propose efficient policies by solving a Markov Decision Process (MDP) of the optimization problem. More specifically, we use one of the best performing policy-gradient DRL algorithms, namely the Proximal Policy Optimization (PPO) [1]. The designed policy is able to determine the processing type (local or remote), which user(s) to offload, and the number of packets to process, depending on the channel conditions, the data buffers and energy buffers states at UEs. Moreover, we show that NOMA provides some benefits.

Related works were proposed in the literature. In [2], the authors presented an RL-based approach for computation offloading and energy transmission in a MEC-EH based system. They minimized the energy consumption and execution delay by decomposing the problem into sub-problems to overcome the dimensionality curse. Extension to MIMO-MEC has been done in in [3] by using Deep Q-Network [4] and PPO. In [5], the authors used PPO in a multi-user environment with multiple MEC servers. However, unlike our work, the mentioned works did not consider a strict delay neither a simultaneous transmission with NOMA. Here, we extend the previous work in [6] that considers only a single user, or equivalently an OMA, scenario.

The paper is organized as follows: we introduce the system model in Section II. We formulate the problem and describe its solution in Section III. Simulation results are drawn in Section IV, and we conclude in Section V.

II. System Model

We consider a MEC server deployed near a BS serving two UEs with limited battery. UEs have EH ability. They also have small sized data buffer. The BS is the centralized decision center that dictates the type of processing for each UE and the number of packets to be executed at the beginning of every time slot. In addition, NOMA is enabled whenever the decision involves both UEs offloading.

A. Energy & Battery Model

We suppose that both EH devices $\text{UE}^{(1)}$ and $\text{UE}^{(2)}$ are equipped with batteries of finite capacity of $\mathcal{N}^b$ energy units.

Each energy unit corresponds to $\mathcal{S}^e$ Joules. The arrival of energy packets is modeled as an independent and identically distributed (i.i.d.) random Poisson process with mean μ^b. At each time step t, the captured energy, e_t, is stored in the battery while the excess energy is discarded. The battery level is denoted by $b_t^e \in [0, \mathcal{N}^b)$. The probability distribution is given by:

$$p(e_t = \mathcal{E}) = e^{-\mu^b} \cdot \frac{(-\mu^b)^{\mathcal{E}}}{\mathcal{E}!}. \tag{1}$$

B. Data Buffer Model

The data buffer at each UE$^{(1)}$ and UE$^{(2)}$ stores the packets awaiting their execution. It is modeled as a vector $\mathbf{d}$ of size $\mathcal{S}^d$ where each component d^k, $k \in [0, \mathcal{S}^d)$, represents a data packet by its age (-1 is for an empty buffer slot). The data packets are ordered in a descending order w.r.t their age. The number of packets in the buffer at a given time step t is denoted by $\mathcal{N}_t^d \leq \mathcal{S}^d$. The arrival of data packets is also following an i.i.d. random Poisson process with mean μ^d. The probability distribution is given by:

$$p(a_t = \mathcal{D}) = e^{-\mu^d} \cdot \frac{(-\mu^d)^{\mathcal{D}}}{\mathcal{D}!} \tag{2}$$

where a_t is the number of arrived packets at time step t. If a_t exceeds the available $\mathcal{S}^d - \mathcal{N}_t^d$ slots in the buffer, **buffer overflow** occurs and the excess will be dropped. If the packets present in the buffer reach a maximum pre-fixed delay, ∇, due to strict delay constraint, **delay violation** occurs and these packets will be dropped as well.

C. Channel Model and Multiple Access

We consider a Rayleigh fading channel for data transmission at UEs, with $\mathcal{W}^{ul}$ and $\mathcal{W}^{dl}$ denoting the uplink and downlink bandwidths respectively. The channel gain is denoted by $x_t = |h_t|^2$ with h_t its complex amplitude. It is assumed constant during a time slot and varies independently through time following the exponential distribution with mean μ^c:

$$p(x_t = \mathcal{C}) = \frac{1}{\mu^c} e^{\frac{-\mathcal{C}}{\mu^c}}. \tag{3}$$

The noise is a Additive White Gaussian Noise (AWGN) with spectral density $\mathcal{N}_o$. In addition, we assume that channel estimation is performed at the UEs and only a quantized version $\underline{x}_t = \mathcal{F}(x_t)$ is transmitted to the BS, $\mathcal{F}$ being the quantization function. We use $\underline{x}_t$ and $\overline{x}_t$ as the respective lower and upper values of the discrete set of channel gains $\mathcal{X}$ at BS.

When both UEs can offload to the MEC server, power domain NOMA is considered to share time and frequency resources between users. It consists in allocating different power levels to UEs for transmission and using Successive Interference Cancellation (SIC) at the reception. Without loss of generality, let UE$^{(1)}$ has a better channel gain than UE$^{(2)}$, $x_t^{(1)} > x_t^{(2)}$ at time step t. (In simulations any UE can have a better channel). The received signal at the BS y_t^{bs} is

$$y_t^{(bs)} = \sum_{j=1}^{2} \sqrt{p_t^{\mathbf{o},(j)}} \cdot h_t^{(j)} \cdot s_t^{\mathbf{o},(j)} \tag{4}$$

where $s_t^{\mathbf{o},(j)}$ and $p_t^{\mathbf{o},(j)}$ are the transmitted signal and its corresponding offload power at UE$^{(j)}$. The SIC decoder first proceeds into decoding UE$^{(1)}$'s signal while considering UE$^{(2)}$'s signal as interference. Then, it subtracts the decoded signal from the received one to decode interference-free UE$^{(2)}$'s signal. The resulting optimal rates using quantized channel gains are as follows:

$$\mathcal{R}_{noma,t}^{ul,(1)} = \mathcal{W}^{ul} \cdot \log\left(1 + \frac{p_t^{\mathbf{o},(1)} \cdot \underline{x}_t^{(1)}}{p_t^{\mathbf{o},(2)} \cdot \overline{x}_t^{(2)} + \mathcal{W}^{ul} \cdot \mathcal{N}_o}\right) \tag{5}$$

$$\mathcal{R}_{noma,t}^{ul,(2)} = \mathcal{W}^{ul} \cdot \log\left(1 + \frac{p_t^{\mathbf{o},(2)} \cdot \underline{x}_t^{(2)}}{\mathcal{W}^{ul} \cdot \mathcal{N}_o}\right). \tag{6}$$

Note that the interference term $p_t^{\mathbf{o},(2)} \cdot \overline{x}_t^{(2)}$ in uplink rate considers the upper value of the quantization interval to account for worst case scenario in decoding UE$^{(1)}$ signal.

In the downlink, the BS controls the power allocation at time step t, by allocating $\delta p_t^{\mathbf{b},(bs)}$ to UE$^{(1)}$ and $(1-\delta)p_t^{\mathbf{b},(bs)}$ to UE$^{(2)}$, where $p_t^{\mathbf{b},(bs)}$ is the total power and δ is the power correlation coefficient. The received signals at UEs are

$$y_t^{(1)} = h_t^{(1)} \cdot s_t^{\mathbf{b},(bs)} \tag{7}$$

$$y_t^{(2)} = h_t^{(2)} \cdot s_t^{\mathbf{b},(bs)} \tag{8}$$

where $s_t^{\mathbf{b},(bs)} = \sqrt{\delta p_t^{\mathbf{b},(bs)}} \cdot s_t^{\mathbf{b},(1)} + \sqrt{(1-\delta)p_t^{\mathbf{b},(bs)}} s_t^{\mathbf{b},(2)}$ is the broadcasted signal from the BS to UEs. Then we perform SIC decoding at the strongest UE, namely UE$^{(1)}$. Therefore, UE$^{(2)}$'s signal is first decoded subject to interference from UE$^{(1)}$, the decoded signal is subtracted, and then UE$^{(1)}$'s signal is decoded. Meanwhile, UE$^{(2)}$ directly performs decoding with interference from UE$^{(1)}$. The resulting optimal rate expressions are as follows:

$$\mathcal{R}_{noma,t}^{dl,(1)} = \mathcal{W}^{ul} \cdot \log\left(1 + \frac{\delta p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(1)}}{\mathcal{W}^{dl} \cdot \mathcal{N}_o}\right) \tag{9}$$

$$\mathcal{R}_{noma,t}^{dl,(2)} = \mathcal{W}^{ul} \cdot \log\left(1 + \frac{(1-\delta) p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(2)}}{\delta p_t^{\mathbf{b},(bs)} \cdot \overline{x}_t^{(2)} + \mathcal{W}^{dl} \cdot \mathcal{N}_o}\right). \tag{10}$$

III. Problem Formulation and Resolution

We firstly describe the decisions that can be made at the UEs and their resulting cost in terms of consumed energy. At the beginning of each time slot t of duration $\mathcal{T}$ ms, the UE$^{(j)}$ with $j \in \{1, 2\}$ can decide between three actions: staying *idle*, *local processing* or *offloading* with a given number of packets $m_t^{(j)}$ to be processed.

- *Idle*: UE$^{(j)}$ does not consume any energy,

$$\mathcal{E}_t^{\mathbf{i},(j)} = 0. \tag{11}$$

- *Local Processing*: UE$^{(j)}$ has the computational capacity to execute locally $m_t^{(j)} \leq \mathcal{M}^{\mathbf{l}}$ packets with power $p_t^{\mathbf{l},(j)}$ per processed packet. It consumes then

$$\mathcal{E}_t^{\mathbf{l},(j)} = m_t^{(j)} \cdot p_t^{\mathbf{l},(j)} \cdot \mathcal{T}. \tag{12}$$

- *Offloading*: UE$^{(j)}$ can offload $m_t^{(j)} \leq \mathcal{M}^{\mathbf{o}}$ packets with power $p_t^{\mathbf{o},(j)} \leq \mathcal{P}^{\mathbf{o}}$. If only one UE is offloading, the consumed energy at UE$^{(j)}$ for either $j = 1$ or 2 is

$$\mathcal{E}_t^{\mathbf{o},(j)} = m_t^{(j)} \cdot \Big(\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(j)}}{\mathcal{R}_{su,t}^{ul,(j)}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\mathcal{R}_{su,t}^{dl,(j)}} + \rho \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{su,t}^{dl,(j)}} \Big) \quad (13)$$

where $\mathcal{L}^{ul}$ and $\mathcal{L}^{dl}$ are the lengths in bits of the transmitted data packets in uplink and downlink, respectively. $\mathcal{T}^{\mathbf{w}}$ is the waiting time at UEs for receiving the results. $p^{\mathbf{w}}$, $p^{\mathbf{r}}$ and $p^{\mathbf{d}}$ are the consumed powers for waiting, reception and decoding. ρ is a scaling factor for efficient decoding. The optimal rates for such a single-user offloading are expressed as follows:

$$\mathcal{R}_{su,t}^{ul,(j)} = \mathcal{W}^{ul} \cdot \log \left(1 + \frac{p_t^{\mathbf{o},(j)} \cdot \underline{x}_t^{(j)}}{\mathcal{W}^{ul} \cdot \mathcal{N}_o} \right) \quad (14)$$

$$\mathcal{R}_{su,t}^{dl,(j)} = \mathcal{W}^{dl} \cdot \log \left(1 + \frac{p_t^{\mathbf{b},(bs)} \cdot \underline{x}_t^{(j)}}{\mathcal{W}^{dl} \cdot \mathcal{N}_o} \right). \quad (15)$$

On the other hand, when both UEs offload at the same time slot, NOMA is used to allow simultaneous transmission. The expressions at both UEs of the consumed energy are

$$\mathcal{E}_t^{\mathbf{o},(1)} = m_t^{(1)} \cdot \Big(\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(1)}}{\mathcal{R}_{noma,t}^{ul,(1)}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\min\{\mathcal{R}_{noma,t}^{dl,(1)}, \mathcal{R}_{noma,t}^{dl,(2)}\}} + \rho \cdot \big(\frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(1)}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(2)}} \big) \Big) \quad (16)$$

$$\mathcal{E}_t^{\mathbf{o},(2)} = m_t^{(2)} \cdot \Big(\frac{\mathcal{L}^{ul} \cdot p_t^{\mathbf{o},(2)}}{\mathcal{R}_{noma,t}^{ul,(2)}} + \mathcal{T}^{\mathbf{w}} \cdot p^{\mathbf{w}} + \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{r}}}{\mathcal{R}_{noma,t}^{dl,(2)}} + \rho \cdot \frac{\mathcal{L}^{dl} \cdot p^{\mathbf{d}}}{\mathcal{R}_{noma,t}^{dl,(2)}} \Big). \quad (17)$$

Notice that offloading decisions only occur when the transmission, waiting, reception and decoding are completed within the time slot duration $\mathcal{T}$. Forcing equality yields the maximal offloading power for each UE$^{(j)}$ $\mathcal{P}^{\mathbf{o}}$ with the maximal number of packets to offload. In addition in Eq. (16), the reception time is tuned according to the slowest communication and the decoding time has two terms since SIC is applied.

A. Markov Decision Processes

As the data buffer, channel and energy buffer dynamics satisfy Markov property, we can formulate the problem as a Markov Decision Process. Any MDP can be described with a state space $\mathbf{S}$, an action space $\mathbf{A}$, a transition model $\mathbf{T}$, and a reward model $\mathbf{R}$. We describe these elements for our system.

- *State space*: It is defined as a vector of data buffer states, channels states and energy states at both users.

$$\mathbf{S} = \{\mathbf{d^{(1)}}, \mathbf{d^{(2)}}, \underline{x}^{(1)}, \underline{x}^{(2)}, \mathcal{N}^{e,(1)}, \mathcal{N}^{e,(2)}\} \quad (18)$$

This state space is of size:

$$|\mathbf{S}| \leq \Big[|\mathcal{X}| \cdot (\nabla + 2)^{\mathcal{S}^d} \cdot (\mathcal{N}^b + 1) \Big]^2 \quad (19)$$

However, imposing ordering on the packets in the data buffer, the state space can be reduced.

- *Action space*: It is an ordered list that contains all the possible actions (a combination of the decisions and the number of packets to execute). The size of the space is defined by $|\mathbf{A}| = \mathcal{M}^{\mathbf{o}} + \mathcal{M}^{\mathbf{l}} + 1$.
- *Reward model*: It is defined as the negative sum of the number of dropped packets during a time slot for both UEs. Therefore we have

$$\mathbf{r}_t = - \sum_{j \in \{1,2\}} (c_{1,t}^{(j)} + c_{2,t}^{(j)}) \quad (20)$$

where c_1 is the number of packets dropped due to delay violation, while c_2 is the number of packets dropped due to a buffer overflow.

Moreover, we setup our problem as an Infinite Discounted Horizon problem, with γ being the discount factor that determines how relevant future rewards are compared to the present one. The overall reward function following a policy π is defined as follows.

$$\mathbf{R}^{\pi} = \lim_{\mathbf{T} \to \infty} \mathbb{E}^{\pi} \left[\sum_{t=0}^{\mathbf{T}} (\gamma)^t \mathbf{r}_t \right]. \quad (21)$$

From this, we define an optimal policy, named π^o as the set of actions that maximizes the overall reward function:

$$\pi^o = \arg\max_{\pi} \mathbf{R}^{\pi}. \quad (22)$$

Finding the exact solution to this optimization problem is hard due to the dimensionality of our problem. That is the goal of next Subsection.

B. Proximal Policy Optimization

RL techniques, which are model-free, are aimed at producing policies without knowledge of the environment transition and reward models (only instantaneous reward $\mathbf{r}_t$). Algorithms based on RL to obtain a relevant policy closed to π^o rely on trial-and-error to built an understanding of the environment, with a balance between exploration and exploitation.

Policy Gradient algorithms are a family of RL methods, that attempt to find the policy directly by optimizing an iterative algorithm. In order to win the curse of dimensionality, this policy is a parametrized function with parameters $\mathbf{w}$. Therefore, the output of the kind of approaches is an policy $\pi^{\mathbf{w}}(a|s)$. Using an objective function $\mathcal{L}^{\mathbf{w}}$, we can measure the quality of the current policy, and update the parameters $\mathbf{w}$ of the policy with Gradient Ascent, in the direction of the gradient $\nabla_{\mathbf{w}} \mathcal{L}^{\mathbf{w}}$.

In addition to the policy update stage, we can also analyze the value function to give us more insights into finding a good policy. Value functions $V^{\pi}(s)$ assesses the quality of being in a certain state s by playing the policy π. This function is helpful for knowing which states are good in the environment, and

which are not. More precisely, this function is the expectation of the sum of rewards $\mathbf{R}_s^a$ following a policy π during an episode where R_s^a is the reward after being in state s and taking an action a. The function has to satisfy the so-called Bellman equation as follows:

$$V^{\pi}(s) = \sum_{a \in \mathbf{A}} \pi(a|s) \left(R_s^a + \gamma \sum_{s' \in \mathbf{S}} P_{s,s'}^a V^{\pi}(s') \right) \quad (23)$$

where γ is the discount factor and $P_{s,s'}^a$ the probability to go from state s to state s' by acting a.

In the following, by a small abuse of notations, we will denote $V^{\pi^{\mathbf{w}}}$ by $V^{\mathbf{w}}$.

The Proximal Policy Optimization (PPO) [1] is related to an Actor-Critic scenario where the actor part is the policy estimation with parameters $\mathbf{w}$, and the critic part is the value function with other parameters $\mathbf{w}'$. More precisely, PPO uses a policy gradient method where Neural Networks approximate the policy and the value function with weights $\mathbf{w}$ and $\mathbf{w}'$ respectively. This algorithm is built upon a previous work called Trust Region Policy Optimization (TRPO) [7] which goal was to limit the update for the policy and so to improve the training stability by forcing the Kullback-Leibler (KL) divergence between the old and the new policies to be smaller than a pre-defined threshold. PPO takes this concept but simplifies it, by replacing the constraint on KL divergence with a clipping.

Mathematically speaking, the TRPO objective function is defined as follows:

$$\mathcal{L}_{TRPO}^{\mathbf{w},\mathbf{w}'} = \hat{\mathbb{E}}_t \left[\frac{\pi^{\mathbf{w}}(a_t|s_t)}{\pi^{\mathbf{w}_{old}}(a_t|s_t)} \hat{A}_t^{\mathbf{w}'} \right] \quad (24)$$

with $\hat{\mathbb{E}}_t$ being the expectation over batch of samples, and $\hat{A}_t^{\mathbf{w}'}$ being the advantage function derived from the value function calculating with weights $\mathbf{w}'$. This advantage function is defined as

$$\hat{A}_t^{\mathbf{w}'} = \sum_{\tau=t}^{T-1} (\gamma\lambda)^{\tau-t} \left(R_{s_\tau}^{a_\tau} + \gamma V^{\mathbf{w}'}(s_{\tau+1}) - V^{\mathbf{w}'}(s_\tau) \right) \quad (25)$$

where λ is the so-called generalized advantage estimation factor. The last term in brackets is related to Bellman equation where $s' = s_{\tau+1}$. Then the TRPO update works as follows

$$\begin{array}{ll} \max_{\mathbf{w},\mathbf{w}'} & \mathcal{L}_{TRPO}^{\mathbf{w},\mathbf{w}'} \\ \text{subject to} & \hat{\mathbb{E}}_t \left[\mathrm{KL} \left(\pi^{\mathbf{w}_{old}}(a_t|s_t), \pi^{\mathbf{w}}(a_t|s_t) \right) \right] \leq \delta. \end{array} \quad (26)$$

The PPO update works as follows

$$\max_{\mathbf{w},\mathbf{w}'} \mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}'} \quad (27)$$

with

$$\begin{aligned} \mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}'} &= \hat{\mathbb{E}}_t \left[\min \left(q_t^{\mathbf{w}} \hat{A}_t^{\mathbf{w}'}, \mathrm{clip}\left(q_t^{\mathbf{w}}, 1-\epsilon, 1+\epsilon\right) \hat{A}_t^{\mathbf{w}'} \right) \right] \\ q_t^{\mathbf{w}} &= \frac{\pi^{\mathbf{w}}(a_t|s_t)}{\pi^{\mathbf{w}_{old}}(a_t|s_t)}. \end{aligned} \quad (28)$$

The PPO compared to the TRPO integrates the constraint into the objective function by clipping the ratio between both policies up to ϵ.

Based on the idea, the PPO algorithm works as follows: we first fill up a memory buffer with data from the transitions of the agent when navigating the environment for a certain number of timesteps. This means that we start with an state s_t, based upon which we sample an action $a_t \sim \pi^{\mathbf{w}_{old}}$ using the actor network, obtain a next state s_{t+1} and a corresponding reward r_{t+1}. Calculating the value function $V^{\mathbf{w}'}(s_t)$, and store the tuple $\left(s_t, a_t, r_{t+1}, s_{t+1}, V^{\mathbf{w}'}(s_t)\right)$ in the memory buffer. The next timestep will have s_{t+1} as the current state, and we repeat the operation. When the determined number of timesteps is reached, the obtained sequence corresponds to an episode. And we start with a new randomly chosen initial state. Once the memory buffer is filled with tuples, it plays the role of batch for training the actor and the critic networks. Given this batch, we build non-overlapping mini-batchs of a given size by picking up tuples randomly.
Looping through all the mini-batches, we estimate the advantage functions $\hat{A}_{b_i}^{\mathbf{w}'}$ for a certain mini-batch b_i, as well as the ratios between the old policy and the new one $q_{b_i}^{\mathbf{w}}$. Then we can calculate our objective function $\mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}',b_i}$ associated with the mini-batch b_i (actually this function is obtained as in (28) where the expectation has been replaced with the expectation over the tuples in the considered mini-batch b_i) and use it to update the weights of both networks using gradient ascent :

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \nabla_{\mathbf{w}} \mathcal{L}_{PPOa}^{b_i} \quad (29)$$

$$\mathbf{w}' \leftarrow \mathbf{w}' + \alpha \nabla_{\mathbf{w}'} \mathcal{L}_{PPOa}^{b_i} \quad (30)$$

where α is the learning rate and $\mathcal{L}_{PPOa}$ is the *augmented* objective function defined as:

$$\mathcal{L}_{PPOa}^{b_i} = \mathcal{L}_{PPO}^{\mathbf{w},\mathbf{w}',b_i} + \beta_1 \mathcal{L}_{VF}^{\mathbf{w}',b_i} + \beta_2 \mathcal{S}^{\mathbf{w},b_i} \quad (31)$$

with $\mathcal{L}_{VF}^{\mathbf{w}',b_i}$ a squared error term for the value function, and $\mathcal{S}^{\mathbf{w},b_i}$ the entropy loss that encourages exploration. β_1 and β_2 are some hyperparameters to tune.

After visiting all the mini-batches b_i, we repeat the training operation on the same batch and the same mini-batchs for $\mathcal{I}$ iterations, and afterwards we wipe out the memory buffer and start the process filling up the memory buffer, and training the agent again (this corresponds to a new epoch), with the obtained policy becoming the old one. After a certain number of epochs, the algorithm converges to a policy $\pi^{\mathbf{w}}(a|s)$ which will be applied during the test phase.

IV. Simulation Results

We first describe the simulation setup. The duration of the timeslot is $\mathcal{T} = 1$ms. The data buffer size is $\mathcal{S}^d = 6$. The maximum delay is $\nabla = 3$ms. The energy unit is $\mathcal{S}^e = 250$nJ, and the battery can accumulate $\mathcal{N}^e = 4$ energy units. The energy unit arrival rate for EH device is $\mu^e = 0.5$. The channel from UEs to MEC server is quantized into 5 discrete states $\mathcal{X} = [-20, -4.437, -1.487, 0.253, 1.492]$dB. The bandwidth in the uplink is $\mathcal{W}^{ul} = 1$MHz and in the downlink $\mathcal{W}^{dl} = 5$MHz. The AWGN spectral density is $\mathcal{N}_0 = -87$dBm/Hz. The power allocation coefficient for downlink NOMA is set to $\delta = 0.5$. The decoding efficiency factor is $\rho = 1$. The

maximum number of offloaded packets is $\mathcal{M}^{\mathbf{o}} = 4$, and of local-processed packets is $\mathcal{M}^{\mathbf{l}} = 2$. The packets sizes are $\mathcal{L}^{ul} = 1000$bits for the uplink and $\mathcal{L}^{dl} = 100$bits for the downlink. The maximum offloading power is $\mathcal{P}^{\mathbf{o}} = 2$mW, and the power used for local operations is $p^{\mathbf{l}} = 150$mW, waiting power is $p^{\mathbf{w}} = 0.1$mW with $\mathcal{T}^{\mathbf{w}} = 0.1$ms, base station transmit power is $p^{\mathbf{b},(bs)} = 50$W, reception power is $p^{\mathbf{r}} = 3$mW, and decoding power is $p^{\mathbf{d}} = 5$mW. This setup leads to a number of states approximately equal to 27×10^6.

To train the PPO, we use a shared layer architecture for both neural networks, where the first two layers are shared between the actor and critic networks, while two other layers are independent for each network. We used 128 nodes for each layer in the network with ReLU activation function in the hidden layers. We train the agent for 10^5 epochs. For each epoch, we have 128 episodes of 128 timesteps, leading to a batch of 16384 tuples. The mini-batch size was set to 128. The number of iterations per batch is set to $\mathcal{I} = 10$. A fixed learning rate $\alpha = 5\times10^{-5}$ was used. The clip factor is $\epsilon = 0.2$, the discount factor is $\gamma = 0.99$ and the generalized advantage estimation factor was set to $\lambda = 0.97$.

We compare the PPO to some naive heuristics, namely the Naive Offload (NO) that chooses only offloading actions, Naive Local (NL) that chooses only local operations, Immediate scheduler (IMM) that chooses the operation (local or offload) enabling the instantaneous maximum number of processed packets, and Naive Random (NR) that follows a random process for choosing its actions.

In Fig. 1, we plot the average percentage of dropped packets versus the packet arrival rate. The test has been done over 1000 episodes of 1000 timesteps. In the top figure (numbered (1)), NOMA has been considered while on the bottom figure (numbered (2)), only OMA/TDMA is considered. In any case, PPO approach outperforms all the naive methods. The comparison between NOMA and TDMA shows that NOMA is much better which means that the added complexity of using NOMA compared to TDMA due to the state space size (and possible loss in performance due to this complexity) is completely compensated for the better transmission speed with NOMA due to simultaneous users scheduling.

In Fig. 2, we plot the average energy consumption (through the number of consumed energy units) per episode vs the packet arrival rates. The PPO approach is also better than the naive methods. So the proposed method consumes less energy and loses less packets.

V. Conclusion

We propose a joint scheduling and offloading optimization problem when NOMA transmission is allowed. Despite of the added complexity which requires to use Deep Reinforcement Learning (via the PPO approach), we show that NOMA is much better than OMA in terms of packet error rate.

References

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *Preprint arXiv:1707.06347*, 2017.

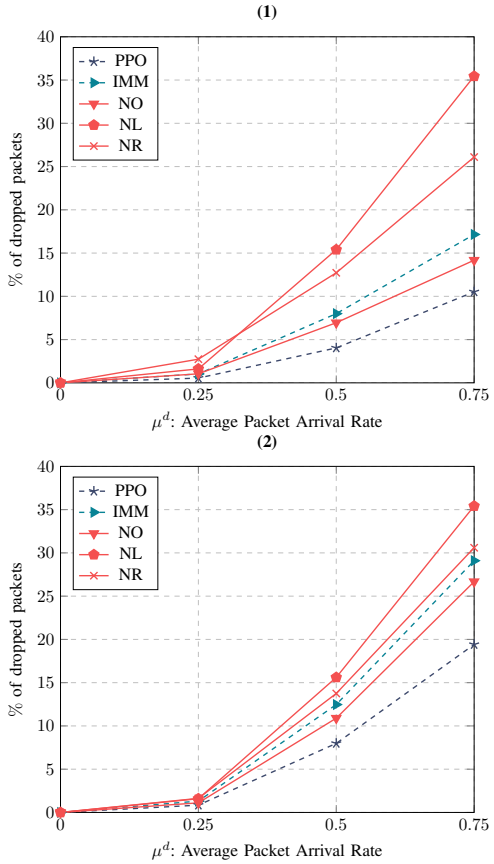

Fig. 1. Percentage of dropped packets vs μ^d for NOMA (1) and TDMA (2)

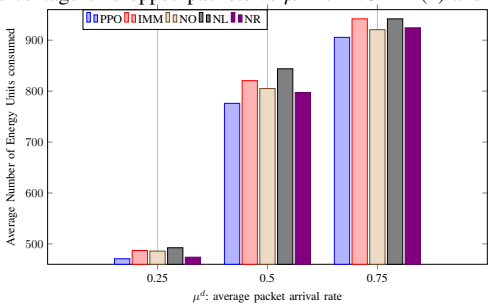

Fig. 2. Number of consumed energy units vs μ^d with NOMA

[2] Zheyuan Hu, Jianwei Niu, Tao Ren, Bin Dai, Qingfeng Li, Mingliang Xu, and Sajal K Das. An efficient online computation offloading approach for large-scale mobile edge computing via deep reinforcement learning. *IEEE Transactions on Services Computing*, 15(2):669–683, 2021.

[3] Abdeladim Sadiki, Jamal Bentahar, Rachida Dssouli, Abdeslam En-Nouaary, and Hadi Otrok. Deep reinforcement learning for the computation offloading in MIMO-based edge computing. *Ad Hoc Networks*, page 103080, 2023.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[5] Lingling An, Zhuo Wang, Jiahao Yue, and Xiaoliang Ma. Joint task offloading and resource allocation via proximal policy optimization for mobile edge computing network. In *IEEE International Conference on Networking and Network Applications (NaNA)*, pages 466–471, 2021.

[6] Ibrahim Fawaz, Mireille Sarkiss, and Philippe Ciblat. Delay-optimal resource scheduling of energy harvesting-based devices. *IEEE Transactions on Green Communications and Networking*, 3(4):1023–1034, 2019.

[7] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.