Mean-Field Multi-Agent Contextual Bandit for Energy-Efficient Resource Allocation in vRANs

Jose A. Ayala-Romero*, Leonardo Lo Schiavo†, Andres Garcia-Saavedra*, Xavier Costa-Perez‡*
*NEC Laboratories Europe GmbH, Germany, ‡i2CAT Foundation and ICREA, Spain
†Universidad Carlos III de Madrid & IMDEA Networks Institute, Spain

***Abstract*—Radio Access Network (RAN) virtualization, key for new-generation mobile networks, requires Hardware Accelerators (HAs) that swiftly process wireless signals from Base Stations (BSs) to meet stringent reliability targets. However, HAs are expensive and energy-hungry, which increases costs and has serious environmental implications. To address this problem, we gather data from our experimental platform and compare the performance and energy consumption of a HA (NVIDIA GPU V100) vs. a CPU (Intel Xeon Gold 6240R, 16 cores) for energy-friendly software processing. Based on the insights obtained from this data, we devise a strategy to offload workloads to HAs opportunistically to save energy while preserving reliability. This offloading strategy, however, needs to be configured in near-real-time for every BS sharing common computational resources. This renders a challenging multi-agent collaborative problem in which the number of involved agents (BSs) can be arbitrarily large and can change over time. Thus, we propose an efficient multi-agent contextual bandit algorithm called ECORAN[1], which applies concepts from mean field theory to be fully scalable. Using a real platform and traces from a production mobile network, we show that ECORAN can provide up to 40% energy savings with respect to the approach used today by the industry.**

I. Introduction

Driven by the O-RAN Alliance [1], RAN *virtualization* (vRAN) has gained the attention of the industry to shift from hardwired BSs to inexpensive general-purpose computing platforms [2], [3]. Despite the rapid success of dense vRANs, the industry today is concerned about the energy consumption of such systems. Verizon and Vodafone have set targets to reach net zero energy emissions by 2040 [4]. China Mobile committed to reduce energy consumption and carbon emission intensity by no less than 20% by the end of 2025 [5]. Indeed, even before the latest surge in energy prices, the energy-related expenditures in mobile networks are one of the predominant factors of their costs [6]. vRAN solutions on the market today are not well-suited to achieving these targets.

In contrast to more conventional network functions such as network switches or firewalls, RAN functions have stringent latency constraints to process wireless signals. Violating processing deadlines, which span between 1 and 3 ms depending on the scenario [7], may result in users losing wireless synchronization with the BS, which leads to dropping connectivity [8]. Hence, an industry-grade BS must respect such deadlines with 99.999% probability to provide reliability [7]. However, to process wireless signals, a BS has to execute compute-intensive operations such as decoding forward error correction (FEC) codes [8], which prevents conventional virtualization platforms based on general-purpose CPUs from attaining the required reliability.

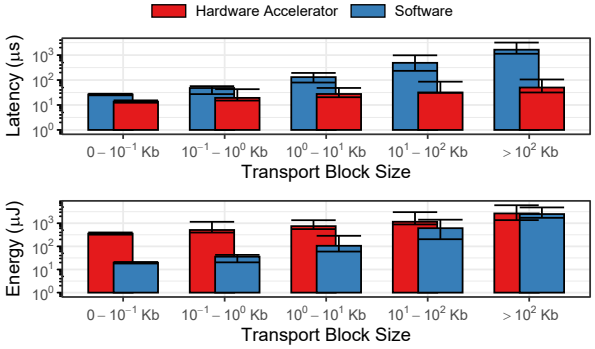

Fig. 1: Latency and energy consumption to process 5G signals with a CPU (software) and a GPU (hardware accelerator). Experimental results obtained using a platform introduced in §VII. Bars show median values. Error bars show 10^{th} and 90^{th} percentiles.

Consequently, today's industry relies on *hardware accelerators* (HAs) to offload the most intensive FEC processing operations. HAs are GPUs, FPGAs, or ASICs that are specialized in specific operations (FEC, in the case of vRANs) and help attain industry-grade reliability. Fig. 1 (top) shows that a HA (a GPU in this case) can process large chunks of data (called transport blocks) an order of magnitude faster than processing this data via software. Unfortunately, HAs are expensive (between 5 and 50× more expensive than a CPU core[2]), and consume a lot of energy (between 30 and 80% of the overall consumption in a commodity server).

A natural approach to reduce the energy toll of these systems is to use HAs *opportunistically* and rely on software as much as possible. Fig. 1 (bottom) shows that small and medium-sized transport blocks (TBs) can be decoded in software within common deadlines (1-3 ms) with an order of magnitude lower energy consumption than a HA. However, to design a policy that determines when to use a HA, three challenges emerge:

1) *Complexity*: Balancing the workload between software (CPUs) and hardware (HAs) while preserving reliability is particularly challenging. This is because the time required to process signals is hard to predict since it depends, not only on the size of the transport block as depicted in Fig. 1,

[1]Our solution is publicly available at https://github.com/jaayala/ECORAN

[2]An NVIDIA GPU V100 required to accelerate FEC operations costs around $10K, in contrast to $150 of an Intel Xeon CPU core.

but also on many other features such as, e.g., signal-to-noise ratio (SNR). The error bars in Fig. 1 hide away the impact of these additional features, which must be taken into account to attain industry-grade reliability.

2) *Speed*: BSs must adapt their configuration to the changing environment dynamics, but wireless conditions change very fast in real mobile networks [7] due to the high mobility of users and reflectors in the environment. Therefore, a useful policy must operate at the same time granularity.
3) *Scalability*: Every transmission time interval (TTI$\leq$1 ms), a large number of signals from multiple users and cells have to be processed by the vRAN, e.g., NTT Docomo famously centralized up to 48 cells in a single platform [9], and each cell may handle dozens of users concurrently. Hence, a practical policy must reliably operate at scale.

To address these challenges, we propose an Energy-aware learning strategy for Computing Offloading in vRANs (ECORAN). ECORAN consists in (i) a simple and fast threshold-based offloading rule (ECORAN-R) to operate in real-time (< 1 ms), and (ii) a multi-agent contextual bandit algorithm (ECORAN-P) to optimally configure the offloading rule for each BS in near-real-time ($\sim$100 ms).

We hence face a problem where multiple BSs (learning agents) need to find the optimal configuration of their offloading strategy as a function of their context (traffic load). However, the offloading strategy of each BS affects all the others as they all share the same computing resources, which renders a multi-agent collaborative problem. As mentioned before, the number of BSs involved can be arbitrarily large, which increases the complexity of the solution.

To address this problem, we propose a novel multi-agent learning algorithm that uses concepts from mean field theory to make it fully scalable. Importantly, we build ECORAN to be standard-compliant such that our solution can be smoothly transferred to the industry. To this end, each agent is hosted by an O-RAN xApp in a near-real-time RAN intelligent controller (Near-RT RIC), as defined by O-RAN [1].

To evaluate our solution, we have integrated ECORAN in a real O-RAN system in a lab environment and experimentally assessed its performance using traces collected from a real operational RAN. We verified that the execution time of our solution meets the time constraints of the platform with a negligible energy footprint. We also evaluated its convergence with up to 60 learning agents, which is aligned with today's deployments, showing a fast convergence rate independent of the number of agents. We also compared ECORAN with other benchmarks in scenarios with a dynamic number of BSs, revealing up to 41% energy savings with respect to using dedicated HAs, which is the industry standard today. To summarize, the main contributions of this work are:

- We built a fully-fledged experimental platform and performed an analysis of the performance and energy consumption of a real vRAN system.
- Based on the gathered data, we devised the structure of a strategy to improve the system's efficiency substantially based on opportunistic offloading.
- We formulated the problem as a multi-agent contextual bandit and proposed an efficient learning algorithm based on concepts from mean field theory to make it scalable.
- We evaluated our approach in our experimental platform using traces from a real operational radio access network.

II. Related Work

A. Energy Efficiency in Mobile Networks

Energy efficiency in mobile networks is extensively researched, with the primary focus on the energy consumed by BSs for amplifying wireless signals (a.k.a. transmission power) [10], [11], [12]. Numerous studies and literature address this issue, delving into analytical models and deployment planning tools derived from these models [10], [11], [12].

The current trend in network densification involves deploying a higher number of small base stations with lower transmission power and higher data rates [13]. Due to network virtualization, the computing capacity needed to process high-bitrate signals is now a significant factor in the Radio Access Network's (RAN) energy consumption [14], [15]. While a few studies focus on virtualized RANs, such as a sequential decision-making algorithm for CPU resource allocation in the RAN [16] and a Bayesian online learning algorithm to optimize energy consumption in virtualized RANs [15], [17], none of these consider hardware acceleration, crucial for industry-grade systems [18], [19].

The use of HAs brings new challenges as these high-performing devices are energy-hungry and costly. Hence, we address this unexplored challenge and propose a novel strategy to share HAs in order to minimize energy consumption while meeting the performance targets. To the best of our knowledge, this is the first work that addresses this problem.

B. Multi-Agent Learning for Mobile Networks

Previous Multi-Agent Reinforcement Learning (MARL) methods often handle only a small number of agents due to the exponential increase in complexity with the number of agents, known as *the curse of dimensionality* [20]. A few works tackle the scalability issue by using concepts from Mean Field Theory [21], [22]. In these studies, interactions within the agent population are approximated by those between a single agent and the population's average effect. While these techniques were applied in fields like the control of Unmanned Aerial Vehicles (UAVs) [23] and the management of ride-sharing platforms [24], [22], they were never applied to mobile networks to the best of our knowledge.

In mobile networking, Single-agent Reinforcement Learning (RL) finds extensive use in spectrum management [25], network diagnostics [26], software-defined networking [27], among others. In mobile network problems, the Markov Decision Process (MDP) is often particularized with a *contextual bandit* due to several reasons. First, these problems typically have an infinite horizon, with the goal of optimizing long-term performance, leading to a zero discount factor. Second, state transitions are often independent of the selected action. Finally, the reward observation is usually not delayed. Consequently,

numerous works express mobile network challenges through contextual bandits [28], [29], [16], [15], [17], [30].

In this work, we deal with an arbitrarily large number of agents solving a contextual bandit collaboratively. To the best of our knowledge, such a setting has not yet been considered in the literature on mobile networks. To address it, we introduce a novel algorithm that combines a multi-agent contextual bandit approach with ideas from mean field theory.

III. Background

In 5G systems, Base Stations (BSs) exchange modulated radio signals with User Equipment (UE) to wirelessly transfer data, following protocols outlined by the *New Radio* (NR) interface. During each Transmission Time Interval (TTI), a variable amount of data bits is packaged into a *transport block* (TB) for each active UE. The TTI duration ranges from 125 μs to 1 ms, depending on the BS configuration. The TB size (in bits) depends on factors like the BS's scheduler, modulation and coding scheme (MCS), signal-to-noise ratio (SNR), etc. Fig. 2 depicts a BS receiving data from three mobile users.

At the transmitter, each transport block (TB) undergoes operations like modulation or rate matching for conversion into radio signals [7]. The reverse process at the receiver extracts data from these signals. We specifically focus on forward error correction (FEC) among these operations, as it is the most compute-intensive task [8], typically offloaded to a HA for fast execution. FEC-decoding involves an iterative *belief propagation* algorithm, with the number of required iterations (and computing operations) depending on factors like TB size, MCS, or SNR. Refer to [31] for more details.

The novel O-RAN architecture for mobile systems defines a computing platform known as O-Cloud to offload signal processing workloads from virtualized BSs. An O-Cloud provides signal processors comprised of general-purpose CPUs (software processing) and HAs such as FPGAs, GPUs, or ASICs. Each processor queues FEC processing requests in a first-in-first-out (FIFO) queue and, once processed, the resulting TB data is sent back to the associated BS (see Fig. 2). In O-RAN, BSs are controlled by a near-real-time RAN intelligent controller (Near-RT RIC) using apps, known as *xApps*, which operate in the timescale of $\sim 10 - 100$ ms (i.e., 10x or 100x longer than a TTI). In this paper, we deploy a data-driven policy in a Near-RT RIC to control the offloading strategy of BSs deployed over a prototype O-Cloud platform.

IV. Problem formulation

In this section, we formulate an opportunistic HA offloading problem for 5G signal processing workload as a discrete-time decision-making problem where each time step k corresponds to a TTI. We let $\mathcal{B}_k$ denote the set of B_k BSs that share the same O-Cloud platform at time step k. Note that the number of active BSs may change over time. Every time step k, each BS $b \in \mathcal{B}_k$ receives from its users a set of encoded TBs $\mathcal{T}_k^b$ that must be FEC-decoded by the O-Cloud.

Every TB $d_i \in \mathcal{T}_k^b$ is characterized by its SNR (c_i), MCS (m_i), and the amount of data bits it carries (l_i). BSs

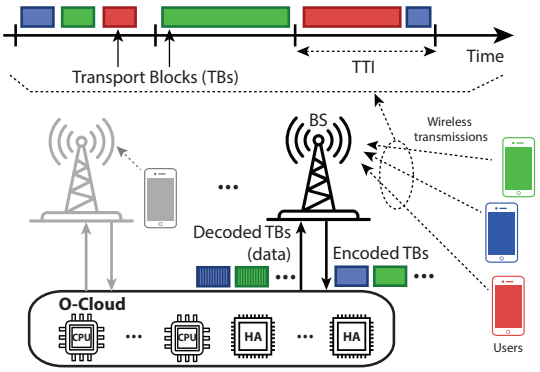

Fig. 2: System architecture. A BS with three mobile users is shown. At the top, we depict each TTI with different TBs. The size of each TB and the associated radio resources are decided by the BS's scheduling algorithm. The (encoded) TBs are sent to the BS, which uses the O-Cloud to decode them to extract the data.

must assign each encoded TB to one type of processor: a software processor (no offloading) or a hardware accelerator (offloading). We let $\mathcal{A}_k^b$ denote the set of assignment actions taken by the BS $b \in \mathcal{B}_k$ at time k. Each individual action $\alpha_i \in \mathcal{A}_k^b$ is associated with a TB $d_i \in \mathcal{T}_k^b$. Therefore, both sets have the same cardinality ($|\mathcal{A}_k^b| = |\mathcal{T}_k^b|$), which may change over time steps k, i.e., the number of TBs generated by a BS can change for different TTIs. Evidently, the individual action $\alpha_i \in \mathcal{A}_k^b$ takes binary values. When $\alpha_i = 0$, $d_i \in \mathcal{T}_k^b$ is assigned to a CPU queue (software processing), and when $\alpha_i = 1$, the TB is assigned to the HA queue (for hardware acceleration). We also let $\mathcal{A}_k$ denote the joint set of actions, i.e., $\mathcal{A}_k := \{\mathcal{A}_k^b \mid b \in \mathcal{B}_k\}$.

We now let $\gamma(\alpha_i, d_i)$ and $q(\alpha_i, d_i)$ denote the processing and waiting/queuing time of d_i, respectively. Note that $\gamma(\cdot)$ and $q(\cdot)$ not only depend on the selected resource for processing (software or HA) as we show in Fig.1, but also on the features associated with the TB , i.e., c_i, m_i, and l_i. We will study this in more detail later. In line with the industry standards, we assume that all the CPU cores are identical as well as all the HAs in the O-Cloud.

According to the specifications, every TB in the system older than a time deadline τ is discarded, incurring data loss. A TB can be discarded while waiting in a queue ($q(\alpha_i, d_i) > \tau$) or after being processed ($\gamma(\alpha_i, d_i) + q(\alpha_i, d_i) > \tau$), whatever happens first. We assume a long enough queue in the HA pool to avoid TB losses due to queue overflow.

Note that, although $\gamma(\cdot)$ and $q(\cdot)$ are continue-valued, i.e., a processing task can finish between two consecutive time steps, the decision intervals (every TTI) are discrete, and hence the problem can be formulated in discrete time. However, as processing a task can take more than one time step, the actions taken in k may affect the system state in future time steps.

We now let $E_k := \sum_{b \in \mathcal{B}_k} \sum_{d_i \in \mathcal{T}_k^b} e(\alpha_i, d_i)$ denote the energy consumed by the O-Cloud due to the transport blocks generated at time step k, where $e(\alpha_i, d_i)$ corresponds to the energy consumed by the processor selected by action α_i to process TB d_i. Similarly, we define the ratio of TBs successfully processed in the O-Cloud within the deadline

during the time step k as $\zeta_k \in [0, 1]$. We then formulate our opportunistic HA offloading problem as follows:

Problem 1.

$$\min_{\{\mathcal{A}_k\}_{k=1}^{K}} \quad \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} E_k$$

$$\text{s.t.} \quad \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \zeta_k \geq 1 - \epsilon.$$

where ϵ sets the target reliability (e.g., $\epsilon = 10^{-5}$ in [7]).

Problem 1 assumes that the O-Cloud system is dimensioned with sufficient computing resources (HAs and CPUs) to operate *always* within the problem's feasibility region, but this may not necessarily hold always. If that is the case, the goal is to maximize throughput regardless of energy consumption:

Problem 2.

$$\max_{\{\mathcal{A}_k\}_{k=1}^{K}} \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{T} \zeta_k$$

Solving these problems is highly intricate. Firstly, O-Cloud energy consumption relies on computing resource utilization (CPU and HAs), influenced by the number of generated TBs from the BSs, the selected actions $\mathcal{A}_k$, and TB characteristics (c_i, m_i, and l_i) at time step k and previous steps. Notably, a TB lost at time k might have been assigned in a prior time step (TTI $< \tau$), and queued TBs add up waiting time to TBs in the near future. Secondly, TB processing time and energy consumption involve a random component, as observed in our experiments in §V. This randomness pertains to how bits are ordered within a TB and how an FEC decoder extracts information (see [31]). Thirdly, both TB energy consumption and processing time depend on O-Cloud hardware specifics and software processing implementation, which can vary across platforms or evolve over time due to hardware/software upgrades. Fig. 3 illustrates processing time variations for different software and HA decoder implementations and configurations. Lastly, determining which problem to solve (Problem 1 or 2) is inherently challenging as it depends on unknown system dynamics.

Therefore, our main problem cannot be solved analytically. One typical approach is to model it as a Markov Decision Process (MDP) and employ Deep Reinforcement Learning (DRL). However, this is not feasible due to time constraints. *xApps* make decisions every 100 ms, while computing resource allocations need to be made every TTI $\leq$ 1ms, as discussed in §III. Even with relaxed time constraints, the problem remains challenging due to a vast action space, involving $2^{|\mathcal{A}_k|}$ possible actions, where $|\mathcal{A}_k|$ can be arbitrarily large. Additionally, the action space dynamically changes over time, contrasting with the typical RL assumption of a fixed action space. Furthermore, our primary goal is to minimize the system's energy cost. Using large models to solve our problems may induce a noticeable energy burden on the system. To avoid this, we must design a lightweight, practical, and scalable solution capable of real-time operation at Base Stations (BSs) with negligible energy impact.

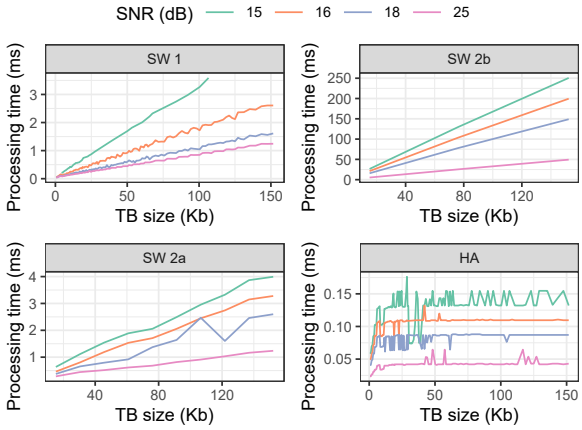

Fig. 3: Processing time of a TB as a function of its size for different SNR values and MCS index 15. We evaluate a HA, two different implementations of the software decoder (SW 1 and 2), and two configurations of the second software decoder (SW 2a and 2b).

V. An Efficient Offloading Strategy

We propose a practical and efficient energy-aware offloading strategy for vRANs. We rely on the insights from our experimental campaign and we exploit the structure of the problem described in §IV to devise an efficient solution.

Minimizing the use of HAs can bring important energy savings. However, an excessive use of CPUs may induce throughput loss as they are a magnitude slower than energy-consuming HAs (see Fig. 1). To get additional insights, we implemented an O-RAN-compliant O-Cloud platform comprised of an NVIDIA GPU V100 as HA and up to 16 Intel CPU cores for signal processing; see §VII for more details. Fig. 3 shows the processing time $\gamma(\alpha_i, d_i)$ of a TB d_i with different TB sizes and SNR values, for 3GPP MCS index 15 [32]. The "SW" plots show the latency incurred by different implementations of a software processor on the same CPU ($\alpha_i = 0$), and the "HA" plot that by the GPU HA ($\alpha_i = 1$).

We observe a key structural difference between software and HA processing. The processing time of a CPU is highly dependent on l_i (TB size), while such dependency practically vanishes in the case of the HA. This structure holds even across different software implementations. Similar behavior can be observed for energy consumption as it is proportional to relative processor busy time. Based on these observations, we propose an intuitive strategy by which small TBs are processed in software, to avoid an early saturation of the CPU. Conversely, as the processing time of the HA is not sensible to the bit size of the TB, large TBs are offloaded. To this end, we define l_{th} as the TB size threshold or *bit threshold* that delimits two operational regions. When $l_i < l_{\text{th}}$, d_i is processed in software, otherwise the HA is used.

In order to evaluate the impact of l_{th} on the performance of the network, we set up a scenario with up to 10 BSs sharing our O-Cloud experimental platform with 4 Intel CPU cores dedicated to software processing and the NVIDIA GPU V100 as HA. We assume (for now) that the workload generated by all the BSs is independent and identically distributed (i.i.d), which implies that the l_{th} is common to all the BSs and

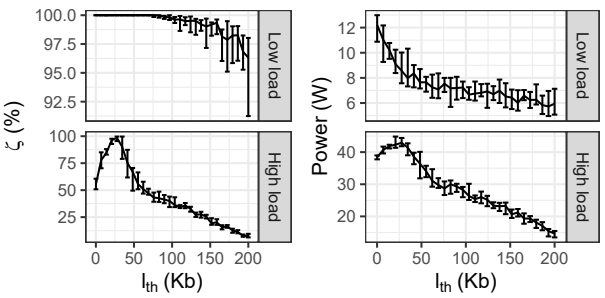

Fig. 4: Ratio of successfully processed TBs (left) and power consumption (right) for low (top) and high (bottom) traffic loads.

simplifies analysis. We will relax these assumptions later. In this way, each BS handles the traffic of 3 homogeneous users, which generate data following a Poisson process with mean 70 Mb/s, and we simulate a Rayleigh wireless channel model with a random mobility pattern with 20 Km/h velocity [33]. We assess two scenarios: a high-load scenario with 10 BSs, and a low-load scenario with 2 BSs. For each scenario, we evaluate 30 different values of l_{th} during 0.5s and for 10 independent runs. Fig. 4 depicts the mean and the 10^{th} and 90^{th} percentiles of the system reliability ζ (left) and the power consumption (right) for both low-load (top) and high-load (bottom).

We observe in the top part of Fig. 4 that, when $l_{\text{th}} < 75{\cdot}10^3$, the ratio of processed TBs (ζ) is maximum. This means that only the TBs with a smaller size than the selected l_{th} are processed using software and the rest of them are offloaded into the HA. For higher values of the bit threshold $l_{\text{th}} > 75{\cdot}10^3$ the CPU gets saturated and some TBs are dropped ($\zeta < 1$). On the other hand, the more we use the CPU (higher values of l_{th}), the higher the energy savings. In that case, as the system is not saturated, we are solving Problem 1 and the optimal point (energy saving maximization without data losses) is around $l_{\text{th}} = 75 \cdot 10^3$. In the bottom part of Fig. 4 we can observe that the higher traffic saturates the O-Cloud, i.e., $\zeta < 1 \forall l_{\text{th}}$. In this case, we need to solve Problem 2, and find the configuration that maximizes ζ. In that case, the optimal configuration is around $l_{\text{th}} = 25 \cdot 10^3$. Note that due to the higher performance of this configuration, the energy consumption is also maximized.

From these experiments, we can extract an important conclusion. *The optimal value of l_{th} depends on the traffic conditions*. In consequence, we need to devise a strategy to find the optimal bit threshold of each BS depending on the system state in terms of traffic conditions. This strategy needs to be learned for each specific deployment, as it also depends on the hardware in the O-Cloud and the software implementation of the decoders (see Fig. 3). Thus, the learned strategy can be deployed at the near-real-time RIC, providing different configurations according to traffic variations.

Nevertheless, this problem is still very challenging to solve in a centralized way for several reasons. The O-Cloud workload is not only defined by the load of TBs (number of TBs per second), but also by the parameters c, m, and l of each TB, which have an impact on the processing time. Second, as the O-Cloud is common for all the BSs considered in the problem, we face a collaborative problem in which the decisions of each agent (BS offloading configuration) affect all the others. For example, a poor action by one BS may saturate the CPU, increasing energy consumption or even causing throughput loss in some other BS sharing the O-Cloud. Third, the number of BSs can be arbitrarily high, especially in highly populated areas. Moreover, we consider the case where the BSs can be switched off and on dynamically, which is a popular feature in 5G [34]. This defines state and action spaces with changing dimensionality over time. For these reasons, we next formulate the problem as a Multi-Agent Contextual Bandit, and propose a practical algorithm using ideas from mean field theory.

VI. Multi-Agent Formulation

A. Background: Markov Games

A Markov game is defined by the tuple $\Gamma = \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where $\mathcal{N}$ is the set of players or agents, $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R}$ is the set of reward functions, and $\mathcal{P}$ is the set of transition probability functions.

Let $\mathcal{N}_t \subseteq \mathcal{N}$ denote the set of N_t agents active by decision period t. Thus, the state observation of each agent n is given by $o_t^n \in \mathcal{S}^n$, and the system state by $\mathbf{s}_t = (o_t^1 \dots o_t^{N_t}) \in \mathcal{S}$. The joint action of all agents is denoted by $\mathbf{a}_t = (a_t^1 \dots a_t^{N_t}) \in \mathcal{A}$. At decision period t, each agent selects the actions based on a deterministic policy $\pi^n : \mathcal{S}^n \mapsto \mathcal{A}^n$. We let $\pi = (\pi^1 \dots \pi^{N_t})$ define the joint policy. In our particular case, the transition probabilities do not depend on the selected actions, i.e., $\mathcal{P}_t^n(\mathbf{s}_t, \mathbf{a}_t) = \mathcal{P}_t^n(\mathbf{s}_t)$ as we consider a contextual bandit formulation. Thus, the state transition is given by $o_{t+1}^n \sim \mathcal{P}_t^n(\mathbf{s}_t)$, where $\mathcal{P}_t^n \in \mathcal{P} : \mathcal{S} \times \mathcal{S} \mapsto [0, 1]$.

Then, at the end of each decision period t, the agents receive a reward $r_t^n(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{R}_t^n(\mathbf{s}_t, \mathbf{a}_t)$, where $\mathcal{R}_t^n \in \mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. We can then write the performance objective of agent n with policy π^n as an expectation:

$$J^n(\pi^n \mid \pi^{-n}) = \int_{\mathcal{S}} \rho(s)\, r^n(\mathbf{s}, \pi(\mathbf{s}))\, \mathrm{d}\mathbf{s} = \mathbb{E}_{\mathbf{s}\sim\rho}[r^n(\mathbf{s}, \pi(\mathbf{s}))], \tag{1}$$

where $\rho(s)$ is the stationary state distribution and π^{-n} is the set of all policies except the one of agent n.

B. Multi-Agent Contextual Bandit

Let us now particularize the above Markov game formulation for our case. As detailed in §V, the bit threshold rule operates at each TTI granularity. However, the configuration of this threshold does not need to be done at each TTI. In fact, in the standard O-RAN architecture, the xApps in the near-RT RIC make decisions every 100 ms. To be standard compliant, we adopt this time granularity to configure the bit threshold as a function of traffic load variations.

Thus, we make decisions at each decision period t, which comprises a set of TTIs $\mathcal{K}_t$. We associate each BS $b \in \mathcal{B}$ with one learning agent $n \in \mathcal{N}$. Hence, we denote $\mathcal{T}_t^n = \{\mathcal{T}_k^b \mid k \in \mathcal{K}_t\}$ as the set of TBs generated by the BS b corresponding the learning agent n during the decision period t.

The state observation $o_t^n \in \mathcal{S}^n$ for agent n aims to characterize the traffic conditions of the TBs generated by its associated BS. Hence, we define the state $o_t^n = \Phi(\mathcal{T}_t^n, D)$ as the 3-dimensional histogram of the TB's features, i.e., MCS, SNR, and size, where D is the number of bins of this histogram in each dimension. Finally, the action selected by agent n determines the bit threshold of its associated BS, i.e., $a_t^n \in \mathcal{A}^n = [l_{\text{min}}, l_{\text{max}}]\ \forall n \in \mathcal{N}_t$, where l_{min} and l_{max} are the minimum and maximum TB size values.

We now let $E_t(\mathbf{s}_t, \mathbf{a}_t)$ denote the energy consumed by the O-Cloud platform (considering both the CPU resources and the hardware accelerator) during period t. We also (re-)define the ratio of successfully processed TBs of the O-Cloud as $\zeta_t(\mathbf{s}_t, \mathbf{a}_t)$. Note that both E_t and ζ_t depend on the global state and the joint actions taken, indicating that both the input traffic and the offloading strategy of all the BSs have a joint impact on the performance metrics of the system. Now, using the definition of expectation in eq. (1), we can define the following constrained decision-making problem:

$$\begin{aligned} \min_{\pi} \quad & \mathbb{E}_{\mathbf{s}\sim\rho}\left[E(\mathbf{s}, \pi(\mathbf{s}))\right] \\ \text{s.t.} \quad & \mathbb{E}_{\mathbf{s}\sim\rho}\left[\zeta(\mathbf{s}, \pi(\mathbf{s}))\right] \geq 1 - \epsilon. \end{aligned} \tag{2}$$

We approximate this constrained problem with an unconstrained one by defining the following reward function:

$$r_t(\mathbf{s}_t, \mathbf{a}_t) = -\left(E_t(\mathbf{s}_t, \mathbf{a}_t) + \lambda\left(1 - \zeta_t(\mathbf{s}_t, \mathbf{a}_t)\right)\right), \tag{3}$$

where λ is a constant that weights the penalty incurred when failing the constraint and the minus sign converts a cost into a reward to be maximized. Using this reward function in eq. (1), we define the optimal policy for agent n as

$$\pi_*^n = \underset{\pi^n}{\text{argmin}}\ J^n(\pi^n \mid \pi^{-n}). \tag{4}$$

At this point, we would like to remark two practical considerations. First, the decision periods defined in this section are several orders of magnitude higher than the TTI (time slot k defined in §IV). Thus, the effect of $\mathbf{a}_{t-1}$ on $r_t(\mathbf{s}_t, \mathbf{a}_t)$ is negligible and therefore each decision period is independent of each other. For that reason, instead of formulating the problem as a multi-agent reinforcement learning (where actions affect future states and rewards), we do it as a multi-agent contextual bandit. This formulation simplifies the problem, allowing us to design a lighter and more effective solution.

Second, the action $a_t^n = \pi^n(o_t^n)$ used by the agent n during the decision period t is computed based on the distribution of the incoming traffic characterized by the observation o_t^n. Due to the nature of the system, the distribution of the incoming traffic during the decision period t is unknown at the beginning of this decision period. For that reason, we use the observation of the previous decision period $t-1$ to make decisions at t, assuming that traffic distribution changes slower than the time granularity of the decision periods.

C. Mean Field Multi-Agent Solution

A common approach in multi-agent learning is to use the *centralized training with decentralized execution* framework [20], [35] to let actor-critic policy gradient methods include information about other agents. The critic of agent n approximates its reward function and is denoted by $R^n(\mathbf{s}, \mathbf{a} \mid \theta_R^n)$, where θ_R^n are the weights of its function approximator. The loss of the critic is given by

$$L(\theta_R^n) = \mathbb{E}_{\mathbf{s}_t\sim\rho, a_t\sim\pi', r_t}\left[\left(R^n(\mathbf{s}_t, \mathbf{a}_t \mid \theta_R^n) - r_t(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right] \tag{5}$$

where the expectation considers that the states $\mathbf{s}_t$ follow ρ, the actions are selected with policy π' that can deviate from π (off-policy learning), and the reward samples are noisy.

We denote the actor of agent n as $\pi^n(o^n \mid \theta_\pi^n)$, where θ_π^n are the parameters of the policy. The actors are updated by applying the chain rule to the performance objective defined in eq. (1) with respect to the actor parameters [36]:

$$\begin{aligned} &\nabla_{\theta_\pi^n} J^n(\pi^n \mid \pi^{-n}) \approx \\ &\mathbb{E}_{\mathbf{s}_t\sim\rho}\left[\nabla_{a^n} R^n(\mathbf{s}_t, \mathbf{a}_t \mid \theta_R^n)\mid_{a^n=\pi^n(o_t^n\mid\theta_\pi^n)} \nabla_{\theta_\pi^n}\pi^n(o_t^n \mid \theta_\pi^n)\right]. \end{aligned} \tag{6}$$

Note that the critic needs information about all the agents as $\mathbf{s}_t$ and that $\mathbf{a}_t$ includes the observations and actions of the N_t active agents at decision period t. This limits the scalability of this approach as the critic's complexity increases exponentially with the number of learning agents. Hence, this approach is typically limited to a small number of agents. In our case, we face an additional challenge as the number of active agents changes over time. This limits the applicability of standard function approximators such as feed-forward neural networks that have a fixed input dimensionality.

In order to tackle these challenges, we adopt ideas from Mean Field Theory [37], [21]. To this end, we assume that the interactions within the population of active agents are approximated by the interaction between a single agent and the average effect of the overall population. Hence, the interactions between each agent and a virtual agent approximating the rest of the agents are mutually reinforced. In this way, the optimal policy of an agent is updated based on the behavior of the overall population, while the population of agents is updated based on the individual policies.

To this end, we define the mean field critic as $\bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n)$, where $\bar{o}_t^{-n}$ is the mean field approximation of the observations of all the agents except n; and, similarly, $\bar{a}_t^{-n}$ is the mean field approximation of the actions of all the agents except n. Specifically, we define $\bar{\mathbf{o}}_t^{-n} = \Phi(\mathcal{T}_t^{-n}, D^o)$, where $\mathcal{T}_t^{-n} = \bigcup_{n'\neq n\in\mathcal{N}_t} \mathcal{T}_t^{n'}$; and $\bar{\mathbf{a}}_t^{-n} = \Phi(\mathbf{a}_t^{-n}, D^a)$, where $\mathbf{a}_t^{-n} = (a_t^1 \dots a_t^{n-1}, a_t^{n+1} \dots a_t^{N_t})$. Moreover, we expand the input of the actor to include the mean field approximation of the observation of the rest of the agents, i.e., $\bar{\pi}^n(o_t^n, \bar{o}_t^{-n} \mid \theta_\pi^n)$, which show better empirical performance. Note that the dimensionality of the input of the mean field critic does not depend on the number of agents, which improves scalability and allows a variable number of learning agents over time. Now, we can redefine eq. (5) as:

$$\begin{aligned} &L(\theta_R^n) = \\ &\mathbb{E}_{\mathbf{s}_t\sim\rho, a_t\sim\pi', r_t}\left[\left(\bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n) - r_t(\mathbf{s}_t, \mathbf{a}_t)\right)^2\right] \end{aligned} \tag{7}$$

Algorithm 1 ECORAN-P

Inputs: Batch size Z, D^o, D^a
Initialize: Reply buffer $\mathcal{D}^n = \emptyset$, θ_R^n, and $\theta_\pi^n \ \ \forall n \in \mathcal{N}$
for $t = 1 \dots T$ **do**
  Observe the system state $\mathbf{s}_t = (o_t^1 \dots o_t^{N_t})$
  Compute mean field approx. $\bar{\mathbf{o}}_t^{-n} = \Phi(\mathcal{T}_t^{-n}, D^o)$;
  Compute the actions $a_t^n = \bar{\pi}^n(o_t^n, \bar{o}_t^{-n} \mid \theta_\pi^n)\ \forall n \in \mathcal{N}_t$
  Use $\mathbf{a}_t$ during t and observe the reward r_t
  Compute mean field approx. $\bar{\mathbf{a}}_t^{-n} = \Phi(\mathbf{a}_t^{-n}, D^a)\ \forall n \in \mathcal{N}_t$
  Store $\langle o_t^n, \bar{\mathbf{o}}_t^{-n}, a_t^n, \bar{\mathbf{a}}_t^{-n}, \mathbf{s}_t, r_t \rangle$ in $\mathcal{D}^n\ \forall n \in \mathcal{N}_t$
  for $n = 1 \dots N_t$ **do**
    Sample Z experiences $\langle o^n, \bar{\mathbf{o}}^{-n}, a^n, \bar{\mathbf{a}}^{-n}, \mathbf{s}_t, r \rangle$ from $\mathcal{D}^n$
    Update θ_R^n by minimizing the critic loss
    $L(\theta_R^n) = \frac{1}{Z} \sum_i \left(\bar{R}^n(o_i^n, a_i^n, \bar{\mathbf{o}}_i^{-n}, \bar{\mathbf{a}}_i^{-n} \mid \theta_R^n) - r_i \right)^2$
    Update θ_π^n by applying the sampled policy gradient
    $\nabla_{\theta_\pi^n} J(\pi^n) \approx \frac{1}{Z} \sum_i \nabla_{\theta_\pi^n} \bar{\pi}^n(o_i^n, \bar{\mathbf{o}}_i^{-n} \mid \theta_\pi^n) \cdot$
    $\nabla_{a^n} \bar{R}^n(o_i^n, a^n, \bar{\mathbf{o}}_i^{-n}, \bar{\mathbf{a}}^{-n} \mid \theta_R^n)$, where
    $a^n = \bar{\pi}^n(o_i^n, \bar{\mathbf{o}}_i^{-n} \mid \theta_\pi^n)$ and
    $\bar{\mathbf{a}}^{-n} = \Phi(\{\bar{\pi}^{n'}(o_i^{n'}, \bar{\mathbf{o}}_i^{-n'} \mid \theta_\pi^{n'}) \mid n' \neq n\}, D^a)$
  end for
end for

and (6) as:

$$\nabla_{\theta_\pi^n} J^n(\pi^n \mid \pi^{-n}) \approx \tag{8}$$
$$\mathbb{E}_{\mathbf{s}_t \sim \rho} \left[\nabla_{a^n} \bar{R}^n(o_t^n, a_t^n, \bar{\mathbf{o}}_t^{-n}, \bar{\mathbf{a}}_t^{-n} \mid \theta_R^n) \nabla_{\theta_\pi^n} \bar{\pi}^n(o_t^n, \bar{\mathbf{o}}_t^{-n} \mid \theta_\pi^n) \right],$$

where

$$a^n = \bar{\pi}^n(o_t^n, \bar{\mathbf{o}}_t^{-n} \mid \theta_\pi^n), \tag{9}$$
$$\bar{\mathbf{a}}_t^{-n} = \Phi(\{\bar{\pi}^{n'}(o_t^{n'}, \bar{\mathbf{o}}_t^{-n'} \mid \theta_\pi^{n'}) \mid n' \neq n \in \mathcal{N}_t\}, D^a). \tag{10}$$

The pseudo-code of our solution is shown in Algorithm 1.

VII. Experimental Evaluation

Our experimental platform consists of a general-purpose server with an Intel Xeon Gold 6240R CPU with 16 cores dedicated to signal processing tasks and an NVIDIA GPU V100. The O-Cloud's Accelerator Abstraction Layer (AAL) [38] is implemented using Intel DPDK BBDev. To process 5G signals, we use a publicly available software library from Intel (Intel FlexRAN [39]) and a proprietary driver for the GPU. The Near-RT RIC comprises an Intel i7-8750H CPU @ 2.20GHz and 8 Gb of RAM.

We have integrated ECORAN in our real O-RAN platform. In particular, ECORAN-P is hosted in the Near-RT RIC as a xApp and computes the bit threshold for each BS. On the other hand, ECORAN-P is implemented into each BS and enforces this bit threshold offloading policy at the TB level. Fig. 5 shows a scheme of our experimental platform.

To emulate real 5G processing workloads, we collected traces from real base stations in Madrid, Spain, during April 2022 using an open-source tool called Falcon [40]. Note that no personal information has been collected. Based on these traces, we emulate the workload generated by two types of BS, as shown in Fig. 6. The left plot depicts the empirical distribution of the size of each TB generated; the center plot illustrates the dynamics of the rate of TBs; and the right plot depicts the SNR distribution of both BSs. In this way, we emulate two distinct BSs, a BS with a single-yet-demanding

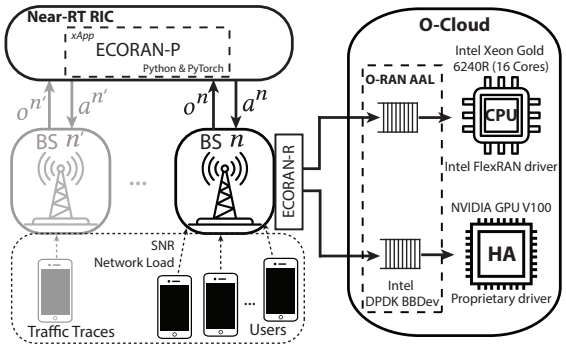

Fig. 5: Scheme of our experimental platform. Each BS has connected a set of mobile users that generate traffic load. All the BS exchange information with the near-RT RIC. Every t, the BSs send their observations o_t^n and receive their respective action a_t^n. For each TB, the BSs decided whether to use software processing or the HA.

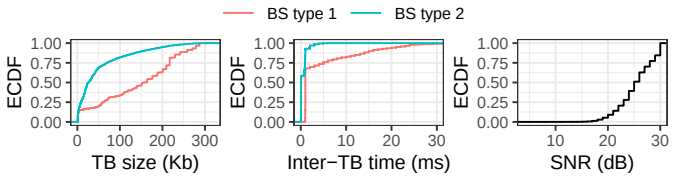

Fig. 6: Characterization of the incoming traffic to each of the BS types considered in the evaluation.

user (Type 1) and a BS with 10 lightly-loaded users (Type 2), with realistic network demand and mobility patterns.

In the rest of this section, we combine these two types of BSs to generate heterogeneous traffic loads for our experiments. In particular, in the settings with 6, 12, 36, 48, 60 active BS, there are 5, 10, 30, 40, 50 BSs of Type 1, respectively, and the rest of Type 2. Although we assess two types of BS, the instantaneous load of individual BSs is very bursty due to the trace-based mobility patterns and demands we emulate. Consequently, each BS requires individual offloading strategies. Moreover, this traffic generation method allows us to compare different solutions fairly.

In the following, we detailed the implementation of ECORAN. Both actor and critic have a similar neural network architecture with two 3D-convolutional layers with 8 and 1 cells, respectively, and a kernel of size 3. Then, connected to the convolutional layers, there are 2 fully-connected layers of 256 units per layer. The output of the actor and critic are activated with sigmoid and linear functions, respectively. Fig. 7 shows the architecture of one learning agent. Note that only the data represented with 3-dimensional matrices is fed into the 3D convolutional layers, while the rest of the data (vector representation) is fed into the critic's fully connected layers. The noise used for exploration follows an Ornstein-Ulhenbeck process with parameters $\theta_{\text{noise}} = 0.15$ and $\sigma_{\text{noise}} = 0.15$, generating temporally correlated values around 0 [41].

Since all the learning agents share the same goal, we consider the particular case in which the reward signal is common and the weights of the neural networks are shared across all the agents, which is a common practice in similar settings [21], [42], [43], [44]. We configure the algorithm with a batch size of 64 samples and $D = D^a = D^o = 5$. We empirically found

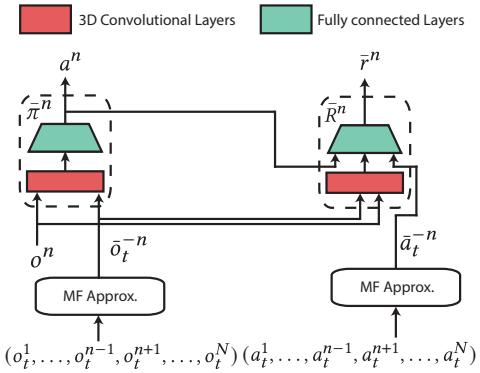

Fig. 7: Scheme of the architecture of learning agent n.

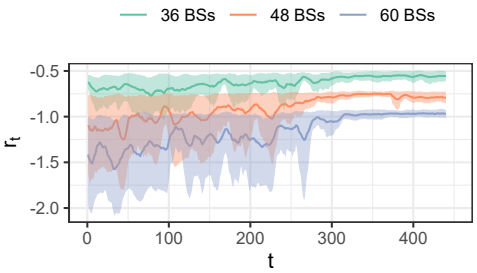

Fig. 8: Reward of the system during the training period for 36, 48, and 60 learning agents. The mean and 10$^{\text{th}}$, and 90$^{\text{th}}$ percentiles (colored shade) of 10 independent runs are shown.

that $\lambda = 2$ is large enough to avoid solutions that violate the reliability constraint.The minimum and maximum values of the TB size ($l_{\min} = 32$ bits, $l_{\max} = 286976$ bits) are given by the 3GPP standard organization [32].

A. Convergence Evaluation

To evaluate the convergence of ECORAN, we consider a scenario with 36, 48, and 60 BSs (i.e., learning agents). This is in line with current large-scale deployments implemented by mobile operators [9]. Fig. 8 shows the evolution of the reward during the training phase. While the complexity of typical multi-agent methods (e.g., [20], [45]) grows exponentially with the number of agents, we show in Fig. 8 that our solution does not show such a dependency. The reason is twofold. First, we share neural network weights across agents, and therefore the weight updates of each agent benefit all others. Second, for every t, we generate as many experience tuples (see line 11 of Algorithm 1) as the number of learning agents. A larger amount of data benefits the learning rate (off-policy learning). These two aspects compensate for the increase in complexity to coordinate a larger number of agents.

B. Comparison with other methods

We now compare ECORAN with a number of benchmarks:

- **Dedicated HAs.** This is the industry's standard approach, where each BS offloads every TB to a dedicated HA to guarantee reliability. We emulate as many HAs as BSs.
- **Always Offload.** BSs offload all their TBs to a shared HA.

All the remaining benchmarks rely on an O-Cloud with two shared processing resources: a HA and a CPU pool.

- **Minimum Waiting Time (MWT).** For each TB, the BS selects the computing resource that minimizes the total waiting time, i.e., $a_i = \text{argmin}_a \ \gamma(a, d_i) + q(a, d_i)$. This is a common approach in queuing theory aimed at minimizing latency. Note that, to evaluate this strategy, the processing times given by $\gamma(\cdot)$ should be known in advance, which is not possible in a real system. Hence, we use a dataset compiling processing times for all possible cases and then we simulate the system based on these pre-known times.
- **ECORAN$_{\text{-MF}}$.** This approach uses ECORAN *without* the mean field approximation, i.e., $\bar{\mathbf{o}}_t^{-n}$ and $\bar{\mathbf{a}}_t^{-n}$ are not available. Thus, each agent becomes an independent learner that only uses its own observations and actions for learning.
- **ECORAN$_{\text{CLDE}}$.** This is ECORAN with a *Centralized Learning Decentralized Execution (CLDE)* approach [20], [45] instead of the mean field approximation. This approach does not support a changing number of agents during learning or execution. We circumvent this challenge heuristically by overdimensioning the algorithm to the maximum number of agents in our training and evaluation, and filling with zeros the observations and actions of the agents that are not active.

We train ECORAN (and its variants) for a general scenario with a changing number of BSs (up to 60). Note that due to the integration of the mean field approximation in our solution, the number of learning agents can vary during both training and execution. After training, we evaluate the same instance of our algorithm in several scenarios with a different number of active BSs. In this way, we also assess the ability of our approach to *adapt* to any trivial deployment.

Fig. 9 shows the energy cost (left), the percentage of bits that are timely decoded (center), and the share of TBs that are offloaded to a HA (right). Moreover, Table I details the energy savings of each approach compared to using dedicated HAs (the industry standard), and the target reliability gap.

"Always Offload" is one of the strategies with the highest energy consumption. However, because the HA is shared across all the BSs and the CPU resources are not exploited, this strategy is not able to meet the reliability target during high-load scenarios, which is a hard requirement in our system. In the case of "MWT", the computing resource with the lowest total waiting time is selected. During low-load scenarios (i.e., 6 and 12 BSs) this strategy frequently selects the HA (86% and 90%, respectively, on average), which consumes substantial energy. Moreover, such a strategy leads to saturating the HA resource, which is fatal for high-load scenarios as the CPU is often forced to process large TBs, which renders poor reliability, up to 12% below the target value. Conversely, we can observe that, like when using dedicated HAs, only ECORAN and ECORAN$_{\text{-MF}}$ meet the target reliability in all of the scenarios. However, compared to using dedicated HAs, ECORAN achieves so with up to 41% energy savings.

Let us discuss why ECORAN$_{\text{CLDE}}$ and ECORAN$_{\text{-MF}}$ obtain suboptimal solutions. The former minimizes energy consumption but violates the reliability target in high-load scenarios, and the latter meets the constraint but consumes more energy with lower traffic loads. In ECORAN$_{\text{-MF}}$, both actor and critic

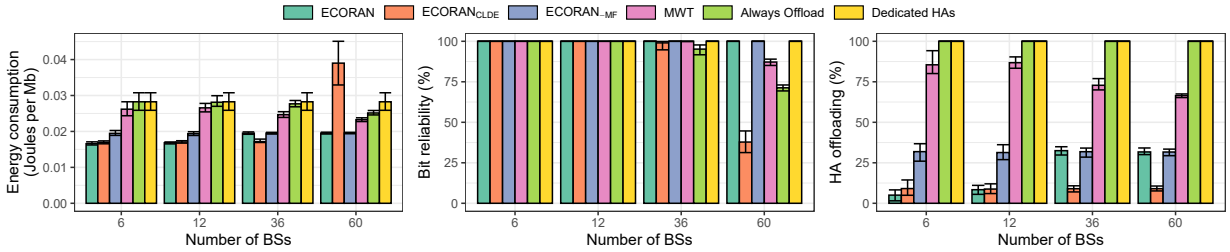

Fig. 9: Performance comparison. Energy consumption in Joules per Mb (left), bit reliability or percentage of correctly decoded bits (center), and HA offloading or percentage of TBs processed into the HA (right). Each bar shows the median, 10^{th}, and 90^{th} percentiles.

TABLE I: Performance comparison. Energy savings per bit computed w.r.t dedicated HA (industry standard). The reliability gap shows the percentage of extra reliability needed to meet industry-grade reliability, i.e., 99.999%

	Energy Savings (%)				Reliability gap (%)			
# BSs / Approach	6	12	36	60	6	12	36	60
ECORAN	**41.23**	**40.51**	30.96	**31.00**	**0**	**0**	**0**	**0**
ECORAN$_{\text{CLDE}}$	40.03	39.69	**38.75**	-38.94	**0**	**0**	2.01	62.02
ECORAN$_{\text{-MF}}$	31.03	31.48	31.02	30.92	**0**	**0**	**0**	**0**
MWT	6.97	5.91	12.75	17.74	**0**	**0**	0.08	12.96
Always Offload	0	0	1.95	10.99	**0**	**0**	5.23	28.72

ignore the information about other agents because the mean field approximations are not available. Thus, the agents cannot distinguish whether the number of active BSs (learning agents) is low (small workloads) or high (large workloads). Therefore, they are unable to learn the relationship between the reward signal (which is common to all agents) and the actual system state. Conversely, the critic in ECORAN$_{\text{CLDE}}$ has a global view of the system at the cost of extra complexity (larger neural network size). Although it can potentially learn the relationship between the global state and the reward during centralized learning, the actor only sees local observations, limiting adaptability during decentralized execution. In contrast, the mean field strategy adopted in ECORAN allows the algorithm to have a global view of the system without compromising complexity and scalability as both the global state and the global action are characterized by vectors of fixed dimensionality independent of the number of active agents.

C. System Validation

We now evaluate the practicality of our solution. We have two requirements for the xApp hosting ECORAN-P in the near-RT RIC: (i) the execution time needs to be faster than the time granularity of 100 ms of the system update; and (ii) produce a negligible energy footprint. Note that a heavy model can be either too slow to operate in real-time or can offset the energy savings if deployed in energy-hungry hardware.

Table II (left) shows the time and energy burden of ECORAN-P executed in the near-RT RIC described in Sec. VII. We consider the worst-case scenario with 60 agents for execution and a model update (actor and critic) for training. We observe an execution time shorter than 1 ms with a negligible energy burden. Conversely, although the training task is more demanding in terms of time and energy, it is only performed temporarily and can even be executed in the background on another machine if needed (off-policy training).

TABLE II: Average time and energy burden of ECORAN-P executed in the near-RT RIC with an Intel i7-8750H CPU (left). Percentage of the energy consumed by ECORAN-P over the total energy (O-Cloud plus near-RT RIC) for different traffic settings in the O-Cloud (right).

	Time (ms)	Energy (J)
Execution	0.572	0.008
Training	46.267	2.312

Traffic Load	% of Energy
Low	3.203 %
High	0.390 %

Table II (right) shows the percentage of energy consumed by ECORAN-P over the total energy consumed by the near-RT RIC and the O-Cloud together. For that, we consider a worst-case scenario in which, apart from the action computation, ECORAN-P is trained during 30 s (convergence time in Fig. 8) every hour. Moreover, we consider low (6 BSs) and high (60 BSs) traffic loads, as the energy consumption of the O-Cloud is highly dependent on the load. We measure an energy footprint of 3.2% in the worst case and below 1% when the O-Cloud has a medium or high load. This validates the practicality of our solution, which operates in a real O-RAN system meeting its time constraints and providing real energy savings.

VIII. Conclusions

In this work, we addressed the energy consumption problem present in new generation virtualized networks. To address this problem, we first conducted a series of experiments using our O-RAN platform. Then, based on the insights provided by the gathered data, we propose an efficient strategy that relies on opportunistic offloading in HAs. As the offloading strategy needs to be configured individually for each BS sharing the O-Cloud, we face a collaborative multi-agent problem. We propose ECORAN, a multi-agent contextual bandit algorithm that uses ideas from mean field theory to improve its scalability. Our experiments showed a fast convergence even with a large number of active learning agents, and up to 40% in energy savings while meeting the reliability target with respect to the current standard adopted by the telecom industry.

Acknowledgements

This work has been supported by the European Commission through Grant No. SNS-JU-101097083 (BeGREEN), 101139270 (ORIGAMI), and 101017109 (DAEMON), and partially by NextGeneration EU through UNICO I+D with Grants 2022/0004810 (SORUS) and CERCA Programme.

REFERENCES

[1] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the Virtualized RAN Ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[2] Heavy Reading, "Heavy Reading's Accelerating Open RAN Platforms Operator Survey," *White Paper*, June 2021.

[3] RCR Wireless News, "From greenfield to brownfield: Open RAN in 2022 (With large scale carrier commitments in place, what's next for the Open RAN ecosystem?)," *Editorial Report*, October 2021.

[4] GSMA Association, "5G energy efficiencies: green is the new black." white Paper, 2020.

[5] China Mobile Limited, "2021 Sustainability Report." white Paper, 2021.

[6] GSMA - Future Networks, "Energy efficiency: An overview," 2019.

[7] X. Foukas and B. Radunovic, "Concordia: Teaching the 5g vran to share compute," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, p. 580–596, 2021.

[8] G. Garcia-Aviles *et al.*, "Nuberu: Reliable ran virtualization in shared platforms," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, p. 749–761, 2021.

[9] N. Docomo, "Base-station equipment with the aim of introducing 3.5-ghz band td-lte," *NTT Docomo Technical Journal*, 2016.

[10] G. Auer *et al.*, "How much energy is needed to run a wireless network?," *IEEE wireless communications*, vol. 18, no. 5, pp. 40–49, 2011.

[11] J. Lorincz, T. Garma, and G. Petrovic, "Measurements and modelling of base station power consumption under real traffic loads," *Sensors*, vol. 12, no. 4, pp. 4281–4310, 2012.

[12] O. Arnold *et al.*, "Power consumption modeling of different base station types in heterogeneous cellular networks," in *2010 Future Network & Mobile Summit*, pp. 1–8, IEEE, 2010.

[13] J. G. Andrews *et al.*, "What will 5g be?," *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[14] J. A. Ayala-Romero *et al.*, "Experimental evaluation of power consumption in virtualized base stations," in *IEEE ICC '21*, 2021.

[15] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Orchestrating Energy-Efficient vRANs: Bayesian Learning and Experimental Results," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2910–2924, 2023.

[16] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrAIn: A deep learning approach tailoring computing and radio resources in virtualized RANs," in *The 25th Annual International Conference on Mobile Computing and Networking*, pp. 1–16, 2019.

[17] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "EdgeBOL: A Bayesian Learning Approach for the Joint Orchestration of vRANs and Mobile Edge AI," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 2978–2993, 2023.

[18] Intel, "Virtual RAN with Hardware Acceleration," *White Paper*, 2020.

[19] Dell, "Dell Open RAN Accelerator Card," *Solution brief*, 2022.

[20] R. Lowe *et al.*, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[21] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *International conference on machine learning*, pp. 5571–5580, PMLR, 2018.

[22] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *The world wide web conference*, pp. 983–994, 2019.

[23] D. Chen *et al.*, "Mean field deep reinforcement learning for fair and efficient uav control," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 813–828, 2020.

[24] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1774–1783, 2018.

[25] N. Zhao *et al.*, "Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 11, pp. 5141–5152, 2019.

[26] F. B. Mismar, J. Choi, and B. L. Evans, "A framework for automated cellular network tuning with reinforcement learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7152–7167, 2019.

[27] Z. Zhang *et al.*, "DQ scheduler: Deep reinforcement learning based controller synchronization in distributed SDN," in *IEEE ICC 2019*, pp. 1–7, 2019.

[28] X. Wang, X. Guo, J. Chuai, Z. Chen, and X. Liu, "Kernel-based multi-task contextual bandits in cellular network configuration," in *2019 IEEE International Conference on Big Data (Big Data)*, pp. 1517–1526, IEEE, 2019.

[29] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 1396–1404, IEEE, 2019.

[30] J. A. Ayala-Romero *et al.*, "Online learning for energy saving and interference coordination in hetnets," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1374–1388, 2019.

[31] Y. Blankenship, D. Hui, and M. Andersson, *Channel Coding in NR*, pp. 303–332. Cham: Springer International Publishing, 2021.

[32] 3rd Generation Partnership Project (3GPP), "3GPP TS 38.214; Technical Specification Group Radio Access Network; NR; Physical layer procedures for data (Release 16) ." Technical Specification, Oct. 2021.

[33] M. T. Arefin and F. R. Reza, *AWGN & Rayleigh Fading Channel: A Throughput Analysis for Reliable Data Transmission*. 2011.

[34] M. Feng, S. Mao, and T. Jiang, "Base station on-off switching in 5g wireless networks: Approaches and challenges," *IEEE Wireless Communications*, vol. 24, no. 4, pp. 46–54, 2017.

[35] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[36] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, PMLR, 2014.

[37] C. Domb, *Phase transitions and critical phenomena*. Elsevier, 2000.

[38] O-RAN ALLIANCE, "O-RAN Acceleration Abstraction Layer General Aspects and Principles. O-RAN.WG6.AAL-GAnP-v01.00," 2021.

[39] Intel, "FlexRAN LTE and 5G NR FEC Software Development Kit Modules," May 2019.

[40] R. Falkenberg and C. Wietfeld, "Falcon: An accurate real-time monitor for client-based mobile network data analytics," in *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, IEEE, 2019.

[41] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[42] J. A. Ayala-Romero, P. Mernyei, B. Shi, and D. Mazón, "Kraf: A flexible advertising framework using knowledge graph-enriched multi-agent reinforcement learning," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 47–56, 2022.

[43] Z. Guan, H. Wu, Q. Cao, H. Liu, W. Zhao, S. Li, C. Xu, G. Qiu, J. Xu, and B. Zheng, "Multi-agent cooperative bidding games for multi-objective optimization in e-commercial sponsored search," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2899–2909, 2021.

[44] J. Li, K. Kuang, B. Wang, F. Liu, L. Chen, F. Wu, and J. Xiao, "Shapley counterfactual credits for multi-agent reinforcement learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 934–942, 2021.

[45] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *International conference on machine learning*, pp. 2961–2970, PMLR, 2019.