# Energy-Aware Scheduling of Virtualized Base Stations in O-RAN with Online Learning

Kalntis, Michail; Iosifidis, Georgios

**DOI**
10.1109/GLOBECOM48099.2022.10001330

**Publication date**
2022

**Document Version**
Final published version

**Published in**
Proceedings of the GLOBECOM 2022 - 2022 IEEE Global Communications Conference

# Energy-aware Scheduling of Virtualized Base Stations in O-RAN with Online Learning

Michail Kalntis, George Iosifidis
Delft University of Technology, The Netherlands
Email: {m.kalntis, g.iosifidis}@tudelft.nl

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001330

***Abstract*—The design of Open Radio Access Network (O-RAN) compliant systems for configuring the virtualized Base Stations (vBSs) is of paramount importance for network operators. This task is challenging since optimizing the vBS scheduling procedure requires knowledge of parameters, which are erratic and demanding to obtain in advance. In this paper, we propose an online learning algorithm for balancing the performance and energy consumption of a vBS. This algorithm provides performance guarantees under unforeseeable conditions, such as non-stationary traffic and network state, and is oblivious to the vBS operation profile. We study the problem in its most general form and we prove that the proposed technique achieves sub-linear regret (i.e., zero average optimality gap) even in a fast-changing environment. By using real-world data and various trace-driven evaluations, our findings indicate savings of up to 74.3% in the power consumption of a vBS in comparison with state-of-the-art benchmarks.**

***Index Terms*—O-RAN, Online Learning, Scheduling, Network Optimization, Green Mobile Networks, Virtualization**

## I. Introduction

### A. Background & Motivation

The importance of virtualizing the base stations is best manifested by the current flurry of industrial activities aiming to develop and standardize O-RAN architectures [1]. The O-RAN Alliance is a global initiative that is devoted to revolutionizing Radio Access Networks (RAN). Its goal is to decentralize a field that has hitherto been dominated by a small number of companies and decrease the entry barrier for more potential firms. In this sophisticated system, RANs are constructed on virtualized network modules, resulting in virtualized Radio Access Networks. The focal point of these components are the virtualized Base Stations, which can be henceforward hosted on various devices, such as commodity servers or tiny embedded devices, and offer the possibility of significant operational/capital expenditure (OPEX/CAPEX) reductions. Promising examples include the open-source srsLTE [2] and OpenAirInterface (OAI) [3].

Indeed, there is a wide consensus that the vBS' programmability can bring crucial performance gains and add the much-needed versatility to the otherwise-rigid, RAN systems. Alas, these benefits come at a cost. These softwarized base stations are found to have less predictable performance and more volatile energy consumption [4], [5], [6], an effect that is amplified when instantiating them in general-purpose computing infrastructure. Hence, it is imperative to understand how to operate, or *schedule*, these vBSs in order to unblock their wide adoption without raising the energy costs of mobile networks to unsustainable levels.



First works aiming at this direction focus, and rightfully so, on learning vBS *meta-policies* (details below). These rules are decided at non-real-time scale and then imposed on the real-time schedulers of each vBS. O-RAN proposals have provisions for such two-level scheduling [7], [8]; and include recommendations for employing learning tools to increase, e.g., the long-term throughput. Nevertheless, to bridge the gap between theoretical proposals and practical results, it is necessary to learn effective policies without relying on strong (and often unrealistic) assumptions such as knowing the full vBS operation profile, the expected data traffic, or future channel conditions. Otherwise, it is likely to be trapped in highly-suboptimal vBS operation points; a finding that we quantify here in terms of excess (i.e., unnecessary) energy costs that can add up to $74\%$ (details in Sec. IV).

*The goal of this work* is to take the next step in this crucial problem by proposing and evaluating a robust algorithm that identifies effective meta-policies (or, simply *policies* hereafter) and is oblivious to information about the underlying vBS operation, their hosting platforms, network conditions and data traffic (or, user needs). The core idea is to model the vBS scheduling as a *bandit learning* problem [9] and design an algorithm that has provably-optimal performance under extensive conditions. The optimality criterion we employ is a combined objective of effective throughput (i.e., modulated by the users' traffic) and energy consumption, where the latter can be prioritized via a tunable weight parameter. Moreover, unlike prior works, our algorithm is lightweight and has minimal overheads, hence can be easily implemented in practice.

### B. Related Work

The idea of optimizing resource management in softwarized networks is not new, and prior works have focused mainly on *(i)* models that relate control knobs to performance functions, *(ii)* model-free approaches that rely on training data, and *(iii)* Reinforcement Learning (RL) techniques. Model-based examples include [4] and [10], which maximize the served traffic subject to vBS computing capacity. However, vBS operation is heavily affected by the hosting platform and network conditions [5], which renders such models impractical. Model-free approaches employ, e.g., Neural Networks, to approximate the performance functions of interest [11], and have been used for network slicing [12], edge computing [13], etc. Yet, their efficacy is conditioned on the availability of training data.

Another prominent approach focuses on runtime observations and is known as Reinforcement Learning. It is used, for example, in interference management [14] and the deployment of Software-Defined Networking (SDN) controllers [15]. RL solutions, however, suffer from the curse of dimensionality and do not offer performance guarantees.

Following an akin approach, contextual bandit algorithms have been employed to decide video streaming rates [16] or BS handover thresholds [17]; assign Central Processing Unit (CPU) time to virtualized BSs [18]; and control millimeter Wave (mmWave) networks [19]. These works require *context*-related information (e.g., about network conditions and traffic), which shapes the performance functions, to be known before the system is configured. More recently, versatile Bayesian learning techniques have been proposed for configuring vBSs [6] —which is closer to the current work. Despite their promise, these solutions require knowing the context, as well as all system perturbations to be stationary over time. Nonetheless, these assumptions are restrictive, especially for heterogeneous and small-cell networks.

To overcome these obstacles, we follow a fundamentally different path and design a vBS control scheme that builds upon the seminal *Exp3* algorithm [20]. Unlike all prior works, our approach: *(i)* offers robust performance guarantees; *(ii)* handles any type of network and load variations (even adversarial); *(iii)* is oblivious to the (time-varying and unknown) vBS performance functions; and *(iv)* exhibits low implementation complexity in terms of memory and computation requirements. This latter feature is in stark contrast with RL techniques (sizeable memory space required to store all space-actions combinations) and Bayesian approaches [21] (heavy-duty matrix inversions). The proposed policy belongs to the class of *adversarial* bandit learning, cf. [9], which has been successfully used in network routing [22] and power control in Internet of Things (IoT) networks [23].

### C. Contributions

We design a learning algorithm that decides thresholds for key vBS operation knobs, namely for the vBS transmission power, the eligible Modulation and Coding Scheme (MCS), and the duty cycle (or airtime). The policy is updated at a near-real-time scale and is subsequently fed to the real-time schedulers that fine-tune the vBS parameters, see Fig. 1. This type of meta-learning, i.e., deciding policies instead of fixing the vBS values directly, is central in the O-RAN architecture and has been recently proposed and studied experimentally, e.g., see [6], [18] and references therein. The proposed algorithm relies on bandit feedback and makes no assumptions about how these knobs affect the vBS performance, nor assumes knowledge of the users' traffic during each scheduling period. This renders it practical for different types of vBS, hosting platforms, and network/traffic conditions. The main contributions of this paper are summarized below:

- We study the vBS scheduling problem in its most general form, i.e., in non-stationary adversarial conditions and without knowledge of traffic and vBS operation functions.
- We design an algorithm that achieves sublinear regret w.r.t. the (unknown) best vBS configuration and has minimal computation and memory overhead. This is the first work applying *adversarial* bandit learning to vBS control.
- We use real-world traffic traces and testbed measurements to demonstrate the weaknesses of prior works [6], as well as the efficacy of the proposed learning algorithm.
- We release the source code[1] of our implementation online, under a permissive free software license, along with detailed documentation.

**Paper Organization**. The rest of this paper is organized as follows. Sec. II discusses the model and formally states the problem. Sec. III introduces the bandit learning algorithm and Sec. IV presents its data/trace-driven evaluation. Sec. V concludes our study.

## II. System Model and Problem Statement

**O-RAN Background**. Our model follows the O-RAN proposals [1], [7], [8], which have provisions for learning-based resource management. "Opening" the RAN is a significant initiative that aims to expand the vRAN ecosystem. It enables multiple vendors to design components of the network architecture, which was previously monopolized by a confined number of large industries that provided end-to-end solutions. We consider a virtualized Base Station comprising a Baseband Unit (BBU) hosted by an off-the-shelf platform and being attached to a Radio Unit (RU). BBU corresponds to a Long-Term Evolution (LTE) eNodeB (eNB) for a 4G network and to a New Radio (NR) gNodeB (gNB) for a 5G network. For the latter, gNB is disaggregated into three focal components: (*i*) the RU, (*ii*) the DU, and (*iii*) the CU.[2] The architecture of the implemented system can be seen in Fig. 1.

*Our goal is to design performance/energy-optimizing configuration policies that adapt to network conditions and user needs*. We assume that an rApp (Policy Decider - PD), see Fig. 1, is instantiated at the Non-RT RIC and implements an algorithm that learns to select efficient *radio policies*. These are essentially *adaptive threshold rules* which guide the underlying real-time schedulers towards the desirable vBS operation. The policy is communicated via the R1 interface to the Non-RT framework, and from there, it is provided to the Near-RT RIC via the A1 interface. In the Near-RT RIC, an xApp (Policy Enforcer - PE) forwards the radio policy to the E2 nodes[3] through the E2 interface. The optimal policy depends on the network conditions and users' load, both of which may vary arbitrarily across time and are typically unknown when the policy is decided. At the end of each decision period $t$, the Near-RT RIC's Data Monitor computes a *reward* by aggregating the adopted performance and energy cost metrics received via the E2 and feeds them to the PD through the O1 interface, before the next decision round.

[1] https://github.com/MikeKalnt/BSvBS

[2] In the O-RAN ecosystem, the terms O-CU, O-DU, O-RU, and O-eNB are used to denote the CU, DU, RU, and eNB, respectively.

[3] E2 nodes refer to RAN nodes, such as an O-CU, an O-DU, or an O-eNB.

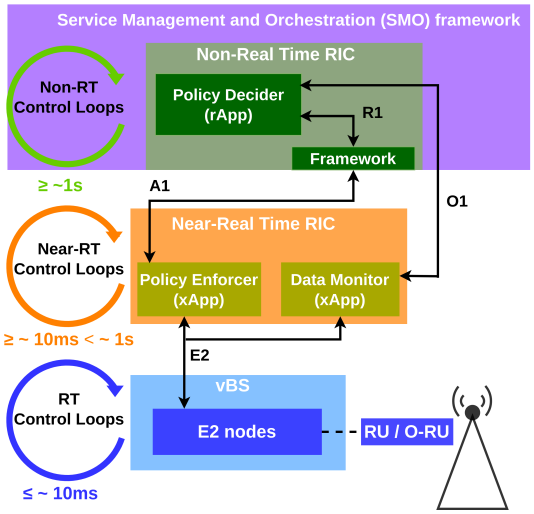



Fig. 1. **O-RAN-compliant architecture & workflow**. The key building block is the Non-Real-Time (Non-RT) RAN Intelligent Controller (RIC), hosted by the Service Management and Orchestration (SMO) framework; and the Near-Real-Time (Near-RT) RAN Intelligent Controller (RIC) [8]. The system has three control loops: (*i*) Non-RT in the Non-RT RIC, which involves large-timescale operations with execution time $>$ 1sec, (*ii*) Near-RT in the Near-RT RIC ($>$ 10msec), and (*iii*) RT (Real-Time) control loops in the E2 nodes ($<$ 10msec).

**vBS Controls**. We consider time-slotted system operation in alignment with O-RAN specs, where each slot represents a period (range of a few seconds) over which a certain policy is being applied. We optimize the system operation over a time horizon of $t = 1, \ldots, T$ slots, where $T$ can take arbitrarily large values and is decided in advance.[4] Without loss of generality (WLOG), we assume unitary slot length.

Our policy includes thresholds for specific scheduling controls that are key to vBS performance, in line with recent measurement-based studies [6], [5], [18]. In detail, for the downlink (DL) operation, we define the set of maximum allowed vBS transmission power control (TPC) $\mathcal{P}_d = \{p_i^d,\ \forall i \in [H]\}$, the set of highest eligible MCS $\mathcal{M}_d = \{m_i^d,\ \forall i \in [I]\}$ and the set of maximum vBS transmission airtime, or duty cycle $\mathcal{A}_d = \{a_i^d,\ \forall i \in [J]\}$, where $H$, $I$, and $J$ denote the number of transmission power, MCS, and airtime levels in DL, respectively. Hence, in period $t$, we determine the DL control:

$$x_t^d \in \mathcal{P}_d \times \mathcal{M}_d \times \mathcal{A}_d.$$

For the uplink (UL) operation, we introduce the set $\mathcal{M}_u = \{m_i^u,\ \forall i \in [K]\}$ and $\mathcal{A}_u = \{a_i^u,\ \forall i \in [L]\}$, where $K$ and $L$ express the number of MCS and airtime levels in UL, respectively.[5] Thus, the UL control in period $t$ is:

$$x_t^u \in \mathcal{M}_u \times \mathcal{A}_u.$$

[4]The assumption of fixing $T$ can be dropped by employing the doubling trick or time-adaptive learning parameters; details in next section.

[5]A UL TPC policy is not defined since the users' transmission power has less impact on the vBS power than the MCS and UL airtime.

Following 3GPP specs, we assume these controls take values from a finite set that includes all possible combinations:[6]

$$\mathcal{X} = \mathcal{P}_d \times \mathcal{M}_d \times \mathcal{A}_d \times \mathcal{M}_u \times \mathcal{A}_u.$$

Thus, the *radio policy* in period $t$ is specified as:

$$x_t = (x_t^d, x_t^u) \in \mathcal{X}.$$

**Rewards & Costs**. The first goal of the learner is to maximize the *effective* DL and UL throughput, which depends on the aggregate of the transmitted data and the backlog in each direction. In particular, in line with prior works (see [6] and references therein), we use the following *utility function*:

$$U_t(x_t) = \log\left(1 + \frac{R_t^d(x_t^d)}{d_t^d}\right) + \log\left(1 + \frac{R_t^u(x_t^u)}{d_t^u}\right), \quad (1)$$

when $d_t^d > 0$ and $d_t^u > 0$, with $U_t(x_t) = 0$ otherwise. $R_t^d(\cdot)$ and $R_t^u(\cdot)$ denote the DL and UL transmitted data during period $t$; and $d_t^d$ and $d_t^u$ are the respective backlogs (i.e., the user needs/requests during $t$). The logarithmic transformation balances the system utility across each stream, but we note that other mappings (e.g., linear) are eligible.

The second goal of the policy is to minimize the vBS energy cost. To that end, we introduce the *power cost* function $P_t(x_t)$, which depends on the configuration $x_t$ in an unknown and possibly time-varying fashion. We kindly refer the reader to our experimental study [5] regarding the challenges in modeling the vBS power cost. We focus principally on *(i)* the power consumption of the CPU at a vBS, which has the lion's share of the total power consumed at the BBU [5]; and *(ii)* the total vBS power consumption (including the RU). We consider these distinct cases to capture the scenarios (arising in practice) where the DU (hosting the BBU) and RU have, or do not have, a common power source. Therefore, we model $P_t(\cdot)$ as a black-box with values observed in runtime.

Putting the above together, the performance criterion for the PD is the *reward function* $\tilde{f}_t : \mathcal{X} \to \mathbb{R}$ defined as:

$$\tilde{f}_t(x_t) = U_t(x_t) - \delta P_t(x_t), \quad (2)$$

where the parameter $\delta > 0$ is set by the PD to tune the relative priority of the utilities and energy costs. It also serves as a metric transformation, allowing a meaningful scalarization of the function components. Further to that, we introduce, for technical reasons, the *scaled* reward function $f_t : \mathcal{X} \to [0, 1]$ since our learning algorithm operates on that interval. An easy-to-implement mapping that ensures this normalization is:

$$f_t(x_t) = \big(\tilde{f}_t(x_t) - \tilde{f}_{min}\big) / \big(\tilde{f}_{max} - \tilde{f}_{min}\big). \quad (3)$$

The scaling parameters $\tilde{f}_{min}$ and $\tilde{f}_{max}$ can be directly determined beforehand based on the value of $\delta$ (depending on the importance given to each component of the function), the minimum/maximum value of the power cost function (might be the minimum and maximum value of the monetary cost associated with the vBS operation), the minimum/maximum

[6]For instance, the MCS values are predetermined, and similarly, one can quantize the power and airtime values; see, e.g. [2] for the *srsLTE* vBS.

transmission power of the vBS, airtime, MCS and user loads. The bounded reward assumption comes WLOG.

**Environment & System Volatility**. It is crucial to note that both reward components vary with time. There are several factors contributing to this effect. First, the user traffic that shapes $U_t$ changes, sometimes drastically —e.g., in small cell networks where user churn is high. Second, the network conditions might as well vary (in slow, fast, or mixed timescales), and this affects the achieved data transmissions (hence $U_t$ changes even for fixed $x_t$) but also impacts the energy cost $P_t$ (low Signal-to-Noise Ratio (SNR) induces more BBU processing costs [5]). Third, the operation cost of the vBS hosting platform is subject to the variations of external computing loads (e.g., when co-hosting other services or other vBS/DUs), changes in the monetary cost (or availability) of the energy price, and so on. Importantly, all the above factors are *unknown* at the beginning of each scheduling period $t$. Indeed, it is challenging to predict the user loads, energy availability, channel conditions, etc., over a few seconds. This, in turn, means that often in practice, when we decide $x_t$ in each slot, we do not have access to the function $f_t$.

**Learning Objective**. The goal of the PD is to find a sequence of configurations $\{x_t\}_{t=1}^T$ that aggregate rewards so as to approach, asymptotically, the cumulative reward achieved by the single best (ideal) configuration. Formally, we employ the metric of *static expected regret*:

$$\mathcal{R}_T = \max_{x\in\mathcal{X}} \left\{ \sum_{t=1}^{T} f_t(x) \right\} - \mathbb{E}\left[ \sum_{t=1}^{T} f_t(x_t) \right], \qquad (4)$$

where the first term describes the best configuration that can be only selected with hindsight, i.e., with a priori knowledge of all future reward functions until $T$; and the second term measures the achieved cumulative reward by our policy. Note that the expectation is induced by any possible randomization in the selection of $\{x_t\}$ that is introduced by the PD.

Eventually, our objective is to devise a rule that decides the configurations in such a way that the average regret,[7] for any possible realization of rewards $\{f_t\}_{t=1}^T$, diminishes asymptotically to zero, i.e., $\lim_{T\to\infty} \mathcal{R}_T/T = 0$. Furthermore, we wish to ensure this condition without knowing the sequence of rewards, not even having access to $f_t$ at the time $x_t$ is being decided. This makes our policy applicable to a range of practical scenarios, such as in highly volatile wireless environments, locations with high user churn, or small-cells where user demands are non-stationary [24].

## III. Bandit Learning Algorithm

Our PD builds on the seminal *Exp3* algorithm [20] that imposes no assumptions on the sequence of rewards $\{f_t\}_{t=1}^T$. This, in turn, means that the obtained performance bounds are guaranteed to hold independently of how the network and environmental factors vary and/or affect the rewards and costs. Hence, unlike prior works such as [6] and [18], the achieved performance is robust and valid for non-stationary conditions. Besides, as we demonstrate in Sec. IV, the actual performance is often substantially higher than these worst-case bounds. Another prominent feature of our algorithm is that it works with *bandit* feedback, i.e., it is adequate to observe the outcome $f_t(x_t)$ of the employed configuration $x_t$ —instead of the entire $f_t(\cdot)$, which remains unknown.[8]

In detail, the underlying idea is that we learn, on the fly, the correct probability distribution $y_t$ (the sequence of distributions) from which we can draw the configuration $x_t$ for each period $t$:

$$x_t \sim \mathbb{P}(x_t = x') = y_t(x'), \forall x' \in \mathcal{X}.$$

The distributions $\{y_t\}_t$ belong to the probability simplex:

$$\mathcal{Y} = \left\{ y \in [0,1]^{|\mathcal{X}|} \;\middle|\; \sum_{x\in\mathcal{X}} y(x) = 1 \right\},$$

and are calculated in each round using the following carefully-crafted explore/exploit rule:

$$y_t(x) = \frac{\gamma}{|\mathcal{X}|} + (1-\gamma)\frac{w_t(x)}{\sum_x w_t(x)}, \;\; \forall\, x \in \mathcal{X}. \qquad (5)$$

Parameter $\gamma \in [0,1]$ determines the extent to which the PD samples a configuration randomly (exploration), or prioritizes those configurations found to perform well (exploitation). The latter happens with the help of the weight vector $w_t = \big(w_t(x): x = 1, \ldots, |\mathcal{X}|\big)$ that tracks the success of each tested policy. In particular, we update the weights at the end of each period, using the exponential update:

$$w_{t+1}(x) = w_t(x) \exp\left( \frac{\gamma \Phi_t(x)}{|\mathcal{X}|} \right), \;\; \forall\, x \in \mathcal{X}, \qquad (6)$$

where $\Phi_t(x)$ is an unbiased function estimator defined as:

$$\Phi_t(x) = \begin{cases} f_t(x_t)/y_t(x_t), & \text{if } x = x_t, \\ 0, & \text{otherwise.} \end{cases} \qquad (7)$$

We recall that $x_t$ is the selected configuration in slot $t$. If the estimator $\Phi_t(x)$ is used to estimate the actual reward $f_t(x_t)$, it is straightforward to see that:

$$\mathbb{E}[\Phi_t(x) | x_1, x_2, \ldots, x_{t-1}] = f_t(x_t),$$

where $x_1, \ldots, x_{t-1}$ are the configurations chosen up to $t-1$. By weighting each observed value with its selection probability, we ensure that the PD will eventually explore configurations with a small probability.

The steps of the proposed learning scheme are summarized in Algorithm 1, which takes as input the time horizon $T$ over which we optimize the vBS operation and the number of eligible configurations $|\mathcal{X}|$; this information suffices to optimize the value of the exploration parameter $\gamma$. The performance of the algorithm is summarized in the following lemma, which holds for any possible sequence of functions $\{f_t\}_{t=1}^T$:

[7] Due to the exploration phase of the algorithm proposed, it might happen that $\mathcal{R}_{t+1} > \mathcal{R}_t$ because the algorithm in slot $t+1$ explored a configuration that was under-performing compared to the configuration chosen in slot $t$. However, the average regret could still diminish to zero as time evolves. More information is available in Sec. III.

[8] Clearly, when applying $x_t$ we observe $f_t(x_t)$ but do not learn how the system would have performed for any other configuration $x \in \mathcal{X}$, with $x \neq x_t$.

**Algorithm 1:** Bandit Scheduling for vBS (BSvBS)

**Input**: Horizon $T$; Configurations $|\mathcal{X}|$;

**Initialize**: $\gamma = \min\left\{1, \sqrt{\frac{|\mathcal{X}|\ln|\mathcal{X}|}{(e-1)T}}\right\}$;

$$w_1(x) \leftarrow 1, \ \forall x \in \mathcal{X}.$$

**for** $t = 1, 2, \ldots, T$ **do**
- Update the distribution using (5).
- Sample next configuration: $x_t \sim y_t$.
- Receive & scale reward $f_t(x_t)$.
- Calculate weighted feedback using (7).
- Update the weights using (6).

**end**

**Lemma 1.** *Algorithm 1 for a fixed horizon $T$ ensures expected regret defined in* (4)*:*

$$\mathcal{R}_T \leq 2\sqrt{(e-1)}\sqrt{T|\mathcal{X}|\ln|\mathcal{X}|} \tag{8}$$

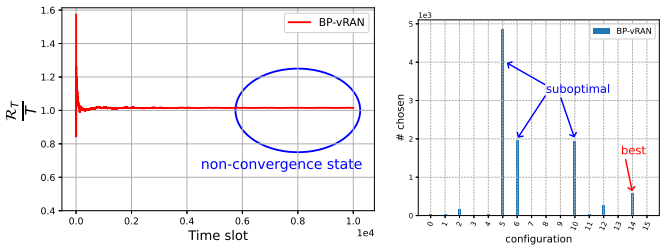
*Proof.* The proof follows by tailoring the main result of [20]. We provide a brief but sufficient explanation. In particular, for selecting $\gamma$, we need to determine an upper bound $g$ on the cumulative reward of the best configuration until $T$. Given that: *(i)* the horizon $T$ is known in advance; and *(ii)* the rewards $f_t(x_t)$ for each chosen configuration $x_t$ at time $t$ cannot be greater than 1 (due to the proposed normalization described in Sec. II), the value of $g$ can be set equal to $T$, i.e., $g = T$. Also, the number of *bandit arms* in our case corresponds to the eligible configurations; hence it is equal to $|\mathcal{X}|$. □

We notice that the expected regret is indeed sublinear $\mathcal{R}_T = o(T)$, which ensures that its time average diminishes to zero. Hence, Algorithm 1 is guaranteed to achieve the same performance as the (unknown) single best configuration, without imposing any conditions on the system operation, network conditions, or user demands. This robust behavior fills the gap of recent related works [18], [6]. Moreover, we highlight that the regret depends on the number of possible meta-policies up to a square root factor. And while their number is expected to be smaller than the number of policies applied to the RT O-RAN level, this finding still points to an interesting direction for further reducing this dependency.

## IV. Performance Evaluation

**Experimental Setup & Scenarios**. We evaluate Algorithm 1 in a variety of scenarios using our recent dataset [6], which includes measurements of the power consumption and performance of vBS policies. The experiments have been conducted using a *srsRadio* vBS [2], and we have used its default schedulers for the underlying real-time decisions (which comply with our meta-policies).[9]

[9]The dataset contains 32 797 measurements for different policies, fixed for approximately one minute. The experiments are carried out on a small factor general-purpose PC (Intel NUC BOXNUC8I7BEH with CPU i7-8559U@2.70 GHz), which deploys the BBU and is configured with a bandwidth of 10 MHz. This means that it supplies a maximum capacity of approximately 32 Mbps and 23 Mbps for the downlink and uplink operation, respectively. See [6] for details.




(a) (b)

Fig. 2. BP-vRAN regret in Scenario B for $|\mathcal{X}| = 16$. **(a)**: Evolution of $R_T/T$ in non-stationary conditions. **(b)**: Numbers of times each eligible policy was chosen in a window of $T = 10k$ slots.

The random perturbations in this setup emanate due to time-varying UL and DL demands, $\{d_t^u, d_t^d\}_t$, and time-varying CQIs (Channel Quality Indicators), $\{c_t^u, c_t^d\}_t$. The latter are integer numbers sent from the User Equipment (UE) to the network to designate how good or bad the channel quality is. The dataset contains $|\mathcal{X}| = 1080$ configurations (policies), but we use a subset of them because calculating the best configuration in hindsight is computationally challenging when $|\mathcal{X}|$ is large.[10] For the power cost function, we set $P_t(x_t) = V_t$, where $V_t$ is the total power consumed by the vBS. We also set $\delta = 1$ to prioritize the minimization of the power consumption.

For the following analysis, we assume that the traffic loads discerned in our system are sampled, either from $d_t^d \sim \mathcal{U}(29, 32)$, $d_t^u \sim \mathcal{U}(20, 23)$ (high DL and UL demands, respectively), or from $d_t^d, d_t^u \sim \mathcal{U}(0.01, 1)$ (low DL and UL demands, respectively). Similarly, the channel qualities are drawn, either from $c_t^d, c_t^u \sim \mathcal{U}(13, 15)$ (good channel qualities[11] in DL and UL, respectively), or from $c_t^d, c_t^u \sim \mathcal{U}(1, 3)$ (poor channel qualities[12] in DL and UL, respectively). According to these distributions, we discern two scenarios:

- Scenario A: the demands and CQIs are consistently drawn uniformly at random from the *high* distribution (*stationary*).
- Scenario B: the demands and CQIs are drawn randomly from the *high* distribution in slots $\{2t-1\}_{t=1}^{\lceil T/2 \rceil}$ and from the *low* distribution in slots $\{2t\}_{t=1}^{\lfloor T/2 \rfloor}$.

The first scenario aligns with the experiments in recent studies, e.g., [6], [5], [18]. The second one implements a *ping-pong* strategy that corresponds to the most challenging-to-learn **adversarial** scenario in regret analysis, cf. [9]. Clearly, an algorithm that performs well under this case is expected (in fact, guaranteed) to perform well in all other scenarios.

**Gap in Prior work**. We first show that state-of-the-art works under-perform in commonly-encountered non-stationary conditions. We focus on BP-vRAN [6], which relies on the seminal GP-UCB algorithm, cf. [21]. To demonstrate that even a simple case hampers its operation, we focus on a subset of configurations: $|\mathcal{M}_d| = |\mathcal{M}_u| = |\mathcal{A}_u| = |\mathcal{A}_d| = 2$ and $|\mathcal{P}_d| = 1$, thus $|\mathcal{X}| = 16$.

[10]We stress that this benchmark configuration is needed for the plots of $R_T$, but **it is not required** when one uses the algorithm in practice. Hence, this limitation is related only to presenting the regret here.

[11]CQI 13 and 15 correspond to SNR of 25 dB and 29 dB, respectively.

[12]CQI 1 and 3 correspond to SNR of 1.95 dB and 6 dB, respectively.

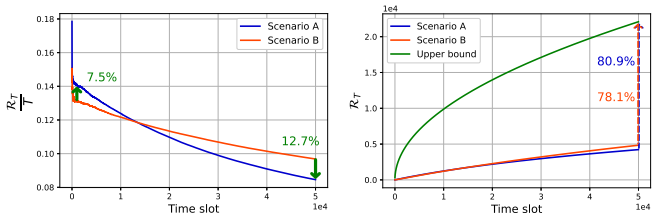
BP-vRAN models the user demands and CQIs as *context*, which are observed before a policy is decided. Given that the context directly impacts the selection of the controls, we will show how changes in the network conditions and demand will affect the algorithm's success. We present an example where the context differs between its observation and application to the system. This case might apply quite often in practice, given that the slots of reference are of several seconds. For the plots in this section, it is crucial to note that the reward function $f_t(x_t)$ is unbounded.[13]

We perform the experiments in Scenario B. Thus, even though we detect low demands and CQIs in the first slot, when the context is applied to the system, these values have increased significantly because, for example, many users with minimum noise interference arrived. In the next time slot, we observe high demands and CQIs, but the actual context has decreased when applied to the system because, for example, few users are present and SNR is reduced. According to the distributions mentioned above, the pattern remains the same in the following slots, i.e., context altering between high and low values in the observation and application to the system.

In Fig. 2(a), we underline that the expected average regret does not decrease, even after $T = 10k$ slots. This happens because the algorithm takes decisions in each slot $t$ by assuming knowledge of $f_t$, which might take arbitrarily low or high values, depending on the network conditions. However, due to the system's volatility, the policy for each $t$ should be selected solely based on past values $\{f_\tau(x_\tau)\}_{\tau=1}^{t-1}$. Fig. 2(b) demonstrates that BP-vRAN insists on selecting configuration $\#5$ (approximately for $50\,\%$ of the slots) and picks the optimal configuration $\#14$ only for $590$ out of the $10k$ slots. This manifests its inability to explore the configuration space due to the non-stationary demands and CQIs.

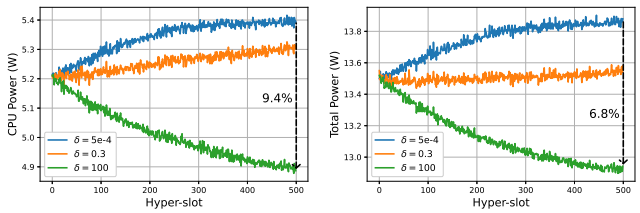
**Evaluation of Algorithm 1**. We consider both Scenarios A and B in our experiments, and we set $|\mathcal{X}| = 256$ for the reasons mentioned above. Fig. 3 depicts the expected regret for 20 independent runs in a window of $T = 50k$ slots. More precisely, Fig. 3(a) plots the decay of $R_T$ for BSvBS in the stationary and adversarial cases. During the first $12k$ slots, the incurred regret for Scenario A is higher than the case of Scenario B, i.e., we perceive a $7.5\,\%$ difference in slot $1k$. As time evolves and confidence in the performance of configurations is built, it is reasonable to observe the regret of the stationary case (Scenario A) to be lower than the more volatile system of Scenario B. That is, in slot $50k$, the regret for the adversarial case is $12.7\,\%$ greater than the stationary's Scenario in slot $50k$. Furthermore, we see in Fig. 3(b) that the experienced regret is by far lower than the upper's bound; that is $80.9\,\%$ and $78.1\,\%$ lower for Scenario A and B, respectively.

In Fig. 4, we show the effect of $\delta$ in the consumed power. In detail, we plot the evolution of the power consumption in Scenario B and we distinguish two cases: *(i)* the total power consumption; and *(ii)* the BBU/CPU power consumption.[14] We run the algorithm for $T = 100k$ slots, and we define a *hyper-slot* of 200-slots-length to facilitate the presentation of results. Indeed, as $\delta$ increases, priority is given to the minimization of the power instead of the maximization of the utility (see eq. 2). Thus, for the hyper-slot 500, we manage to save $6.8\,\%$ and $9.4\,\%$ in the total and CPU power consumption, respectively, by using $\delta = 100$ instead of $\delta = 5e-4$. The operators of the system can use our proposal for conserving energy.




(a) Evolution over time. (b) Upper bound.

Fig. 3. BSvBS regret in Scenarios A and B for $|\mathcal{X}| = 256$.




(a) CPU power. (b) Total power.

Fig. 4. BSvBS power consumption in Scenario B for $|\mathcal{X}| = 256$.

Table I presents the power consumed by BP-vRAN and BSvBS in Scenario B and compares the gains each of them achieves w.r.t. the ideal-minimum-energy of the benchmark. We set $\delta = 100$ to prioritize the power costs strongly over the accomplished utility. By running BSvBS for $T = 50k$ slots, we get $35.5\,\%$ and $38.8\,\%$ savings in the CPU and total power, respectively, compared to BPvRAN. In other words, BSvBS approaches closer to the minimum possible energy cost than its competitor.[15] Interestingly, we also see an increase in these attained gains by observing the performance of BSvBS for more time slots. That is, by doubling the number of slots, the savings increase by more than $1.5\times$ up to $57.9\,\%$ and $68.6\,\%$ in the CPU and total consumption; and become $62.1\,\%$ and $74.3\,\%$, respectively, for $T = 200k$ slots. On the one hand, we note that BSvBS performs better as time passes and reaches closer to the consumption of the best configuration, even in the adversarial scenario. On the other hand, the indication that the power consumption of BP-vRAN almost doubles as the slots are doubled means that the algorithm is stuck exploiting under-performing configurations.

[13]For that reason, we avoid showing the expected regret of BP-vRAN and BSvBS in the same plots.

[14]The testbed uses a single energy source, so the total power consumption aggregates the power of the entire platform and the radio component.

[15]It is worth noting that vBS consumes considerable power even under the best configuration (Min column) or even when no user is active; thus, the direct comparison of the power would not demonstrate the actual gains.

TABLE I
POWER COST (SCENARIO B)

| Slots | CPU (kW) | | | **Saving** (%) | Total (kW) | | | **Saving** (%) |
|---|---|---|---|---|---|---|---|---|
| | BP-vRAN | BSvBS | Min | | BP-vRAN | BSvBS | Min | |
| $200k$ | 1052.6 | 992.1 | 955.1 | **62.1** | 2735.6 | 2609.8 | 2566.2 | **74.3** |
| $100k$ | 534.6 | 501.1 | 476.7 | **57.9** | 1375.2 | 1313.2 | 1284.7 | **68.6** |
| $50k$ | 262.4 | 252.9 | 235.6 | **35.5** | 677.3 | 660.9 | 635.0 | **38.8** |

## V. CONCLUSIONS AND FUTURE WORK

The virtualization of base stations and the design of O-RAN systems lies at the forefront of research in mobile networks. A milestone in this roadmap is finding scheduling policies that maximize the performance of vBSs while restraining their energy consumption. These policies should be practical, i.e., have minimal overheads, and applicable in realistic scenarios, meaning they should not require strong assumptions about the (often volatile) network conditions and/or user needs. The proposed learning scheme is O-RAN-compatible, has robust performance guarantees, offers a knob for prioritizing energy cost reduction, and has lightweight implementation while outperforming other computation-demanding policies (based on different learning approaches). Indeed, our extensive data-driven experiments showcase gains w.r.t. these state-of-the-art competitors that range from $35.5\%$ up to $74.3\%$ in terms of energy savings. To encourage future study in this field, we have made the source code used in this work publicly available. The significance of these results can be understood by considering the number of base stations already deployed, which is only expected to increase in the near future. Finally, our analysis identifies exciting directions for future work, such as improving the learning bounds by reducing further the dependency on the policy dimension.

## VI. ACKNOWLEDGMENTS

This work was supported by the European Commission through Grant No. 101017109 (DAEMON).

## 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] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An Open-Source Platform for LTE Evolution and Experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, ser. WiNTECH '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 25–32. [Online]. Available: https://doi.org/10.1145/2980159.2980163

[3] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A Flexible Platform for 5G Research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, oct 2014. [Online]. Available: https://doi.org/10.1145/2677046.2677053

[4] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally Aware Sum-Rate Optimal Scheduling for Centralized Radio Access Networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[5] J. A. Ayala-Romero, I. Khalid, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Experimental Evaluation of Power Consumption in Virtualized Base Stations," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[6] J. A. Ayala-Romero, A. Garcia-Saavedra, X. Costa-Perez, and G. Iosifidis, "Bayesian Online Learning for Energy-Aware Resource Orchestration in Virtualized RANs," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[7] O-RAN Alliance, "O-RAN Architecture-Description 6.0," Technical Specification, March 2022.

[8] ——, "O-RAN Cloud Architecture and Deployment Scenarios for O-RAN vRAN 2.02 (O-RAN.WG6.CAD-v02.02)," Technical Spec., 2021.

[9] S. Bubeck and N. Cesa-Bianchi, "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems," 2012. [Online]. Available: https://arxiv.org/abs/1204.5721

[10] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, "CARES: Computation-aware Scheduling in Virtualized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. PP, pp. 1–1, 10 2018.

[11] C. Zhang, P. Patras, and H. Haddadi, "Deep Learning in Mobile and Wireless Networking: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[12] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Pérez, "DeepCog: Optimizing Resource Provisioning in Network Slicing With AI-Based Capacity Forecasting," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 2, pp. 361–376, 2020.

[13] H. Guo, J. Liu, and J. Lv, "Toward Intelligent Task Offloading at the Edge," *IEEE Network*, vol. 34, no. 2, pp. 128–134, 2020.

[14] J. Alcaraz, J. Ayala Romero, J. Vales-Alonso, and F. Losilla-López, "Online Reinforcement Learning for Adaptive Interference Coordination," *Transactions on Emerging Telecommunications Technologies*, vol. 31, 10 2020.

[15] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, and L. Wu, "DQ Scheduler: Deep Reinforcement Learning Based Controller Synchronization in Distributed SDN," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[16] B. Alt, T. Ballard, R. Steinmetz, H. Koeppl, and A. Rizk, "CBA: Contextual Quality Adaptation for Adaptive Bitrate Video Streaming," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1000–1008.

[17] 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*, 2019, pp. 1396–1404.

[18] 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*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3300061.3345431

[19] M. Qureshi and C. Tekin, "Fast Learning for Dynamic Resource Allocation in AI-Enabled Radio Networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. PP, pp. 1–1, 11 2019.

[20] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The Nonstochastic Multiarmed Bandit Problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[21] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[22] B. Awerbuch and R. Kleinberg, "Online Linear Optimization and Adaptive Routing," *J. Comput. Syst. Sci.*, vol. 74, no. 1, p. 97–114, feb 2008. [Online]. Available: https://doi.org/10.1016/j.jcss.2007.04.016

[23] A. Marcastel, E. V. Belmega, P. Mertikopoulos, and I. Fijalkow, "Online Power Optimization in Feedback-Limited, Dynamic and Unpredictable IoT Networks," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 2987–3000, 2019.

[24] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, 2016.