Towards Energy Efficiency in RAN Network Slicing

Hnin Pann Phyu, Diala Naboulsi, Razvan Stanica, Gwenael Poitau

Towards Energy Efficiency in RAN Network Slicing

Hnin Pann Phyu
Département de Génie Logiciel et TI
École de Technologie Supérieure (ÉTS)
Montreal, Canada
hnin.pann-phyu.1@ens.etsmtl.ca

Diala Naboulsi
Département de Génie Logiciel et TI
École de Technologie Supérieure (ÉTS)
Montreal, Canada
diala.naboulsi@etsmtl.ca

Razvan Stanica
Univ Lyon, INSA Lyon, Inria, CITI
Villeurbanne, France
razvan.stanica@insa-lyon.fr

Gwenael Poitau
Dell Technologies
Montreal, Canada
gwenael.poitau@dell.com

***Abstract*—Network slicing is one of the major catalysts to turn future telecommunication networks into versatile service platforms. Along with its benefits, network slicing is introducing new challenges in the development of sustainable network operations. In fact, guaranteeing slices requirements comes at the cost of additional energy consumption, in comparison to non-sliced networks. Yet, one of the main goals of operators is to offer the diverse 5G and beyond services, while ensuring energy efficiency. To this end, we study the problem of slice activation/deactivation, with the objective of minimizing energy consumption and maximizing the users quality of service (QoS). To solve the problem, we rely on two Multi-Armed Bandit (MAB) agents to derive decisions at individual base stations. Our evaluations are conducted using a real-world traffic dataset collected over an operational network in a medium size French city. Numerical results reveal that our proposed solutions provide approximately 11-14% energy efficiency improvement compared to a configuration where all the slice instances are active, while maintaining the same level of QoS. Moreover, our work explicitly shows the impact of prioritizing the energy over QoS, and vice versa.**

***Index Terms*—5G, Network Slicing, Energy Efficiency, QoS**

I. Introduction

The telecommunication industry accounts for approximately 2% of total global carbon emissions [1]. By 2030, 8% of the projected global electricity demand will come from the information and communications technology sector as a whole, even in the best case scenario [2]. Energy consumption will continue increasing in beyond 5G and 6G networks, where computationally intensive services will be largely deployed. Although 5G equipment is more energy efficient than 4G [3], with the data traffic volume increasing tremendously along with 5G services, overall energy consumption will increase too. In fact, the energy consumption of a 5G base station is three times higher than that of a 4G base station, when both are considered at a full load [4].

This work was supported by the National Natural Sciences and Engineering Research Council of Canada (NSERC) through research grant RGPIN-2020-06050 and by the CHIST-ERA ECOMOME project, through the Fonds de Recherche du Québec – Nature et Technologies (FRQNT). This work was partially funded by the ANR COCO5G (ANR-22-CE25-0016) project.

5G is envisioned to serve a wide variety of services, with heterogeneous traffic, through network slicing [5]. This is done by forming, on one physical network, multiple virtual networks on a per-service basis, i.e., slices. That said, slices requirements need to be met, including performance isolation. Guaranteeing these requirements and the additional virtualization layer come with some overhead, which produces higher energy consumption with respect to non-sliced network deployments [6]. One of the key objectives in the field is to offer this service differentiation, while reducing the associated CO_2 emissions. Indeed, energy efficiency in networks is no longer an option but a necessity. When delving into this topic, we observe that, today, the highest amount of energy is consumed in the radio access network (RAN), approximately 70% of the overall network energy utilization [7].

To deal with this, several research works consider base station sleep schemes to further optimize the energy consumption in 5G networks [8], [9]. While such techniques show effective results, they are more challenging to be applied directly in the case of multi-services network slicing environments. That is mainly because slice instances can exhibit quite different temporal traffic patterns. Completely shutting down or putting the entire base station into sleep mode could notoriously impact the quality of service (QoS) of users in specific slice instances. This motivates us to introduce a new approach, in which slice instances are dynamically activated and deactivated, according to their traffic patterns, thereby enhancing the overall base station energy efficiency. However, deactivating some slice instances to minimize the energy consumption can potentially degrade the QoS of users. On the other hand, activating all slices all the time, so as to maximize QoS, significantly increases energy consumption. Accordingly, the energy minimization objective shall be coupled with a QoS maximization objective [10].

To manage the trade-off between the two objectives, operators may consider using an EcoSlice, which is a slice instance with bare minimum resources and network functions. By that, it incurs much lower energy consumption than typical slice instances. The EcoSlice is up and running $24/7$ to provide

a bare-minimum service. In some conditions, e.g., low traffic demand, operators may switch the users of other slices to this specific EcoSlice, without a significant QoS impact.

In this regard, we study the problem of slice activation/deactivation, with the objective of minimizing the energy consumption while satisfying the user QoS. The contributions of our work are twofold. First of all, we propose two different approaches for solving the problem, namely a Deep Contextual MAB (DCMAB) algorithm and a Thompson Sampling Contextual (Thompson-C) algorithm. These approaches allow to derive solutions dynamically over time, while considering traffic patterns of individual slice instances deployed at a base station. Moreover, our proposed agents enable operators to navigate users to and from an EcoSlice, if their requested slice instance is activated/deactivated. Second, we evaluate the performance of the proposed approaches and their computational cost using a real-world traffic dataset.

The rest of the paper is organized as follows. Section II discusses the related work of energy efficiency in network slicing. Then, the network model and problem statement are laid out in Section III. In Section IV, we present the detailed design of our proposed solutions. We articulate the results in Section V and conclude the paper in Section VI.

II. Related Work

With the aim of enabling energy efficiency, several research works consider optimizing the allocation of network slice resources (i.e., radio, CPU, transmission bandwidth and power) in the different domains (i.e., RAN, Edge Computing (EC), Core Network (CN) and end-to-end network). At the RAN slicing level, [11] combines deep learning (DL) and reinforcement learning (RL) on a distributed framework to efficiently allocate radio and transmission power resources over base stations. They use stacked and bidirectional Long-Short Term Memory (SBiLSTM) to predict the per slice resources demand on a large time scale and rely on asynchronous advantage actor-critic (A3C) to allocate resources to users on a small time scale. Their proposed framework achieves higher energy efficiency than baselines using static power allocation.

In [12], the authors optimize the energy consumption and computation cost in a network slicing based Cloud-RAN (C-RAN) setting, using a twin-delayed double-Q soft Actor-Critic (TDSAC) approach. Their agent performs the up/down scaling of computing and beamforming power resources. Their work outperforms other baseline RL models in terms of overall network energy and computing cost. Similarly, [6] designs a slice energy consumption model based on the C-RAN architecture. An optimisation problem is solved per-slice, with the objective of minimizing the overall network energy cost, jointly considering communication and computation resources. This approach improves energy efficiency over a baseline focusing only on radio resources.

Focusing on CN slicing, the authors in [13] formulate a security-aware network slicing optimization problem to enhance the energy efficiency of CN nodes. They limit themselves to static resource allocation. Their proposed solution provides more power savings than a greedy approach.

Considering an obvious trade-off, it is sensible to couple QoS maximization and energy consumption minimization. Therefore, focusing on end-to-end network slicing, the authors in [14] aim to maximize the energy efficiency while respecting service level agreement (SLA) constraints. To this end, they rely on statistical federated learning (stFL). Their federated local agents coordinate and predict per slice network metrics, without transferring datasets to a central unit, and largely outperform other federated learning and centralised solutions.

While prior works attempt to achieve the energy efficiency as well as ensuring QoS, we believe there is still room to further optimize the energy efficiency by switching off some of the underutilized slice instances, under some conditions. As of our knowledge, there is no contemporary work studying this problem. In this light, we introduce the RAN slice activation/deactivation problem with the aim of minimizing energy consumption and maximizing QoS. To this end, we rely on the fully decentralized state-aware MAB approaches to enable decisions for slice instances at each base station, while considering the impact on energy and QoS factors.

III. System Model and Problem Statement

We study the problem of slice instance activation and deactivation at individual base stations with the aim of minimizing energy consumption and maximizing QoS. We thus explicitly lay out the system model, energy consumption and user QoS model, needed as part of our defined problem. We then formalise the main objective of our problem. We formulate the latter as a Markov decision process (MDP), and re-design it after that as a contextual Multi-Armed Bandit (MAB) problem.

A. System Model

We consider a time-slotted system. Accordingly, we define τ as the slice activation/deactivation interval (SADI), where slices are active or inactive continuously over the period of that interval. Accordingly, activation/deactivation decisions are made at the end of every τ, for the upcoming $\tau + 1$. The period of the SADI is defined based on the operator policies. Besides, we define T as a time interval of interest, such that $\tau \in T$. Figure 1 illustrates the time frame consideration in our proposed model. In this example figure, we consider four different types of slices: three of them, denoted as Enhanced Mobile Broadband (eMBB), Ultra-Reliable Low Latency Communication (URLLC) and Massive Machine-Type Communication (mMTC), provide services with different QoS levels and they can be activated/deactivated as needed, as well as the EcoSlice which is always up and running.

We define $\mathcal{I}_b$ as a set of slice instances attached to base station $b \in \mathcal{B}$. We define $\mathcal{U}_b^\tau$ as the set of users served by base station b at τ. Accordingly, $U_{i,b}^\tau$ is the set of users that can be served by slice instance i of base station b at τ. Thus, $\mathcal{U}_b^\tau = \bigcup_{i \in \mathcal{I}_b} U_{i,b}^\tau$. Each slice instance $i \in \mathcal{I}_b$ is characterized by a specific QoS class identifier (QCI) [15] and its energy consumption $E_{i,b}^\tau$.

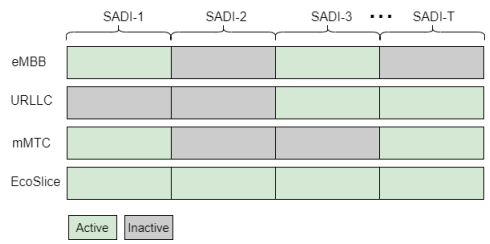


Fig. 1: Slice Activation-Deactivation Interval (SADI) illustration.

Without loss of generality, we consider user delay as an indicator of the QoS, but any other metric could be easily integrated instead. In the optimal slice instance activation scheme, underutilized slice instances are switched off to save energy when certain conditions are met. Consequently, the user-perceived delay is prone to deteriorate. In this light, we deliberately study the impact of optimal slice instance activation on both energy and delay. Hence, we define δ_i as a predefined achievable delay for a slice instance i. We also consider an EcoSlice instance $i_e \in \mathcal{I}_b$, to which users are switched when their requested slice instances are inactive. The EcoSlice instance i_e also has a predefined achievable delay δ_{i_e} and its energy consumption over τ is denoted as $E^{\tau}_{i_e,b}$.

At every τ, each user $u \in \mathcal{U}^{\tau}_b$ makes a specific request, characterized by a delay requirement d^{τ}_u and a traffic flow demand l^{τ}_u. Each base station b has a set of possible configurations $\mathcal{K}_b$. Each configuration $k \in \mathcal{K}_b$ implies activation/deactivation decisions for some slices. In detail, k contains $\{c^{\tau}_i | i \in \mathcal{I}_b\}$. Here, $c^{\tau}_i = 1$ if slice instance i is active at b during τ, and 0 otherwise. Needless to say, $c^{\tau}_{i_e} = 1$ for any τ.

B. Energy Consumption Model

We define the function $f(.)$ to refer to the overall energy consumption of a base station. This energy model can be further fine-tuned based on specific operator resource management policies and activated RAN energy saving features. It is composed of the energy consumption resulting from its individual deployed slice instances $E^{\tau}_{i,b}$ and the static energy consumption of the base station P^{static}_b (i.e cooling and circuit power):

$$f_b(c^{\tau}_i, \mathcal{I}_b) = \sum_{i \in \mathcal{I}_b} c^{\tau}_i \cdot E^{\tau}_{i,b} + P^{static}_b \tag{1}$$

with

$$E^{\tau}_{i,b} = \rho^{\tau}_{i,b} \psi_i P^{dynamic}_b + \psi_i P^{fixed}_b \tag{2}$$

As indicated in the above equation, the energy consumption of slice instances for a base station b encompasses a load-dependent power consumption component, $P^{dynamic}_b$, and a load-independent power consumption component, P^{fixed}_b. Specifically, as the name implies $P^{dynamic}_b$ depends on the traffic load of the base station. It is worth stressing that a slice instance consumes some power even at zero load traffic, in order to run its corresponding network functions. Thereupon, P^{fixed}_b is independent of the traffic load, but related to the energy consumption of associated network functions of specific slice instances.

We note here that not all slice instances are designed equally [16]. Their required network functions and signalling traffic are different [17]. For instance, an URLLC service potentially consumes higher energy, because it requires specific network functions to offer high reliability and very low latency [18]. In short, the more stringent the latency requirements of the service, the higher its consumed energy [19]. Regardless of their requirements for energy, both mMTC and eMBB have flexibility in terms of latency. It is fair to say that, even under the same amount of traffic load, the energy consumption of each service is different. Meanwhile, the EcoSlice is deployed with relatively low energy consumption. It is sensible to conclude that different slice types consume different amounts of energy not only because of their traffic portion but also because of their service attributes [20].

In this vein, ψ_i denotes the power consumption impact factor of slice instance i on both $P^{dynamic}_b$ and P^{fixed}_b. We let the operators define the value of ψ_i based on their slice instances configurations and analytics. Having said that, based on the prior explanation, it is pragmatic to assume that URLLC has larger ψ value than eMBB and mMTC. Needless to say, ψ value of the EcoSlice is the lowest. Besides, $\rho^{\tau}_{i,b}$ is the traffic load portion of associated slice instance i of base station b during τ. For this, we can simply obtain $\rho^{\tau}_{i,b}$ by dividing the traffic load over slice $l^{\tau}_{i,b}$ by the total base station traffic load, as below:

$$\rho^{\tau}_{i,b} = \frac{l^{\tau}_{i,b}}{\sum_{i \in I_b} l^{\tau}_{i,b}} \tag{3}$$

C. User QoS Model

As explained, our objective is coupled with ensuring the user QoS. In this regard, we define the user satisfaction factor η^{τ}_u for each user u at τ associated to slice instance i as follows:

$$\eta^{\tau}_u = \begin{cases} 1 & \text{if } \delta_i \leq d^{\tau}_u \\ 0 & \text{otherwise} \end{cases}, u \in U^{\tau}_{i,b}, i \in \mathcal{I}_b, \tau \in T \tag{4}$$

where d^{τ}_u is the delay requirement of user u at time τ. Consequently, the average per slice QoS at base station b is defined as:

$$\eta^{\tau}_{i,b} = \frac{\sum_{u \in U^{\tau}_{i,b}} \eta^{\tau}_u}{|U^{\tau}_{i,b}|} \tag{5}$$

We then compute the average QoS of the base station b considering all the associated slice instances during τ:

$$\eta^{\tau}_b = \frac{\sum_{i \in \mathcal{I}_b} \eta^{\tau}_{i,b}}{|\mathcal{I}_b|} \tag{6}$$

D. Objective Function

Given the system model and utility models mentioned in the preceding sections, the objective of our slice activation/deactivation problem can be expressed as:

$$max \sum_{\tau=1}^{T} \left[\frac{1}{f_b(c^{\tau}_i, \mathcal{I}_b)} + \eta^{\tau}_b \right] \tag{7}$$

As one can see in Equation 7, the objective function is influenced by the time-varying user demand and active slice instances. Thus, we believe that RL-based approaches are best-suited for this problem, as they enable complex decision-making without requiring an explicit modeling of the network environment [21]. In what follows, we formulate the problem as MDP and contextual MAB.

E. Markov Decision Process (MDP)

An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R})$. $\mathcal{S}$ denotes the set of states in the system. Representing states as feature vectors helps the RL agent converge to a near-optimal reward value, but one can also simplify a problem with a less complex state representation (which might converge to a similar reward, with less computation). In this light, we explore two definitions for a state s in this problem: *i)* energy consumption and QoS of base station in the previous SADI: $s = \left\{f_b(c_i^{\tau-1}, \mathcal{I}_b), \eta_b^{\tau-1}\right\}$ and *ii)* simply the SADI identifier: $s = \{\tau\}$. The reason for the latter definition is that, since the user demand depends on time and it shows a significant periodicity, a simple state representation based only on the SADI might already contain enough information for the RL agent. Each action $a \in \mathcal{A}$ is a configuration k, as defined in Section III-A, and $\mathcal{A}$ is the set of possible configurations: $\mathcal{A} = \mathcal{K}_b$.

In the MDP model, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ captures the stochastic transition probability function to transition to s' from state s based on action a, with $\sum_{s' \in \mathcal{S}} P(s'|s, a) = 1$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. P is unknown to an RL agent. However, it occurs that if an agent knows the current state and the reward obtained in each iteration, it can still converge to the optimal solutions through RL approaches [22]. Accordingly, the formulation of reward function is critical for the RL agent to be able to learn the optimal policy and it usually boils down to the main objectives of the problem. We therefore define reward $r \in \mathcal{R}$ as:

$$r(s, a, s') = \frac{1}{f_b(c_i^{\tau}, \mathcal{I}_b)} + \beta \cdot \eta_b^{\tau} \tag{8}$$

As shown in Equation 8, our reward function is aligned with our objective function. Besides, we define β as a QoS impacting factor on the reward. In short, the larger the β value is, the more the QoS is emphasized with respect to the energy consumption of the base station. If $\beta = 1$, the objective is to find the trade-off between energy and QoS.

Due to the nature of our defined problem, we observe two key points in our MDP representation: *i)* the transition probability of MDP can be simplified, such as $P(s'|s, a) \equiv P(s'|a)$, where states are independent of each other, and *ii)* unlike typical MDP [12], our reward function depends only on current state and action, but not on the successor state. With this, the reward function could be simplified as $r(s, a, s') \equiv r(s, a)$. Based on these observations, we present an equivalent formulation as a state-aware MAB in the following.

F. Multi-armed Bandit (MAB)

In this section, we formulate our slice instance activation/deactivation problem as a state-aware MAB. Formally, state-aware MAB is tupled with $(\mathcal{S}, \mathcal{A}, \mathcal{R})$. Same as in our MDP model, we use the two different state definitions for $s \in \mathcal{S}$: energy consumption and QoS observed over a SADI, or simply the SADI identifier. Similarly, the set of actions $a \in \mathcal{A}$ is the set of available configurations.

We then define the associated reward set $\mathcal{R}$. Since we consider the state-aware MAB (i.e., involving multiple states), our reward distribution is non-stationary, and changes based on the state s (also called context in the following). With this, the reward set can be defined as $\mathcal{R} = \{r(s, a)|\ a \in \mathcal{A}, s \in \mathcal{S}\}$. In this regard, we rely on the same reward calculation as Equation 8. Needless to say, the objective here is to maximize the expected reward $E\left[\sum r(s, a)\right]$.

To evaluate our reward function, one standard approach is to compete with the best-action benchmark. On the other hand, we compute the regret resulting from not selecting the optimal action at each iteration. That said, one would define the cumulative regret incurred by an agent over a total of J time steps as [23]:

$$Regret(J) = \sum_{j=1}^{J} (r_j^*(s, a) - r_j(s, a_j)) \tag{9}$$

where $r_j^*(s, a)$ is the best-action benchmark at round j and can be obtained via $r_j^*(s, a) = \underset{a}{max}\ r_j(s, a)$, and a_j is the action selected by the agent in round j. It is worth mentioning that the regret function is non-negative, as it compares the optimal reward to the actual reward obtained by the agent.

IV. Proposed Solution

In this section, we lay out our two fully decentralized approaches, based on DCMAB and Thompson-C agents.

A. DCMAB Agent

As explained, we use two different context/state definitions and thus we have two types of DCMAB agents: *i)* DCMAB-EQ where the state is the overall energy and QoS over the base station: $s = \left\{f_b(c_i^{\tau-1}, \mathcal{I}_b), \eta_b^{\tau-1}\right\}$, and *ii)* DCMAB-SADI when the state is the SADI: $s = \{\tau\}$. Due to space limitation, we outline them together in Algorithm 1. However, we make sure the main differences (occuring at Line 3 and Line 11 of Algorithm 1) are clearly outlined.

The inputs of the DCMAB agent consist of the probability of selecting a random action ϵ, the learning rate α for the deep neural network (DNN) model, and maximum time steps J to train the DCMAB agent. The output is a trained model, which can predict a reward distribution $\widehat{R}(\tilde{w})$ for the available actions of the associated states. With this, the algorithm begins by the initialization of the weights $\tilde{w}$ with arbitrary values and the variable j referring to an iteration and starting with zero (Line 1). At each step, the agent observes a context s: for DCMAB-EQ , s is the overall energy consumption and QoS factor and for DCMAB-SADI, s is a SADI identifier (Line 3). Then, it predicts the reward distribution for all the actions (Line 4) for a given context. After that, an action is selected by considering the exploration and exploitation

Algorithm 1: DCMAB-EQ and DCMAB-SADI

Input: Probability of selecting a random action ϵ, learning rate α, maximum time steps J
Output: $\widehat{R}(\tilde{w})$

1 Initialize $\tilde{w}$ randomly and $j = 0$
2 **repeat**
3 Observe context/state: s - based on state definition $s = \left\{ f_b(c_i^{\tau-1}, \mathcal{I}_b), \eta_b^{\tau-1} \right\}$ *for DCMAB-EQ* ***or*** $s = \{\tau\}$ *for DCMAB-SADI*
4 Predict Reward Distribution for each action: $\left[\widehat{R}_{\tilde{w}}(a|s)\right]_{a \in A}$
5 **if** *generate random probability:* $rand() < \epsilon$ **then**
6 Select a random action a
7 **else**
8 Select $a = \underset{a \in A}{argmax} \left[\widehat{R}_{\tilde{w}}(a|s)\right]$
9 **end if**
10 Evaluate reward $r(a|s)$
11 Update new state s'
12 Calculate the loss: $\mathcal{L}(\tilde{w}) \triangleq (r(a|s) - \left[\widehat{R}_{\tilde{w}}(a|s)\right]_a)^2$
13 Update the weights: $\tilde{w} \leftarrow \tilde{w} - \alpha \bigtriangledown \mathcal{L}(\tilde{w})$
14 $j \leftarrow j + 1$
15 **until** $j > J$;

tradeoffs (Line 5 - Line 9). Precisely, the random action is selected with probability ϵ, and otherwise the action giving the maximum reward is selected. Next, the chosen action is applied to the defined network environment, which, after the potential reconfiguration, returns an actual reward (calculated using Equation 8) (Line 10). Accordingly, the new state s' is obtained based on the action a for DCMAB-EQ, or new state s' is simply the next SADI (which is independent of the previous action a) for DCMAB-SADI (Line 11). Then, the loss between the predicted reward and the actual reward is computed for the purpose of model training (Line 12). We rely on a gradient descent method to update the weights matrix of the DNN with a learning rate α (Line 13). Then, we go for another iteration (Line 14) and the above process is repeated for a maximum number of time steps J (Line 15).

B. Thompson-C Agent

Unlike DCMAB, the Thompson-C agent adopts a statistical approach with the goal of achieving a proper estimation of the posterior distribution of expected reward for each action. The Thompson-C agent (Algorithm 2) operates as follows. We note that we only consider the SADI identifier as context/state for the Thompson-C agent. Accordingly, the inputs of the algorithm are the context data for all the actions: $\mathbb{C} = (d_k)_{|T| \times |\mathcal{K}_b|}$, where d_k is a context vector for configuration k, the number of dimensions of the context vector $z = |T|$, the tunable parameters φ and M (that can be tuned by the operator as needed) and the maximum number of time steps J to run the Thompson-C. The output is a posterior distribution

Algorithm 2: Thompson-C

Input: Context data $\mathbb{C} = (d_k)_{|T| \times |\mathcal{K}_b|}$, number of dimensions of context/state vector $z = |T|$, φ, M, maximum time steps J, $\sigma = M\sqrt{9zln(\frac{J}{\varphi})}$
Output: $\mathcal{N}\left(\hat{\mu}, \sigma^2 D^{-1}\right)$

1 Initialize $D = \mathbb{I}_z$, $\hat{\mu} = (0)_z$, $j = 0$
2 **repeat**
3 Sample $\tilde{\mu}$ from Gaussian distribution $\mathcal{N}\left(\hat{\mu}, \sigma^2 D^{-1}\right)$
4 Select action $a = \underset{k}{argmax}\ d_k^{\mathbb{T}} \tilde{\mu}$
5 Observe reward r_a
6 $D = D + d_a d_a^{\mathbb{T}}$
7 $\hat{\mu} = D^{-1}(d_a r_a)$
8 $j \leftarrow j + 1$
9 **until** $j > J$;

$\mathcal{N}\left(\hat{\mu}, \sigma^2 D^{-1}\right)$ of having the optimal parameter $\hat{\mu}$. For better understanding, $\hat{\mu}$ can be regarded as a weighted vector for a z-dimensional context/state. The parameter σ can be obtained via $\sigma = M\sqrt{9zln(\frac{J}{\varphi})}$.

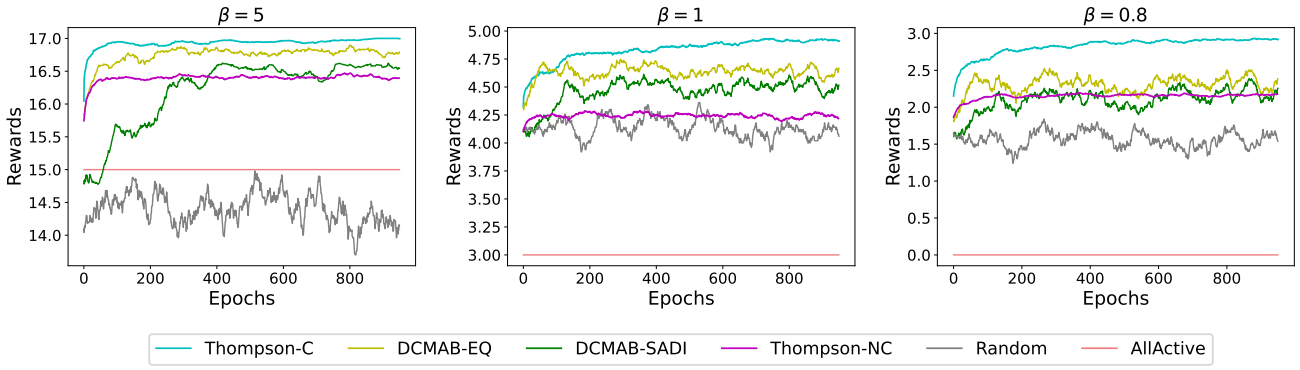
The algorithm begins by setting the parameter D as the identity matrix with a z-dimensional vector. $\hat{\mu}$ is initialized with zeros as a z-dimensional vector and $j = 0$ (Line 1). In each time step j (Line 2), the Thompson-C agent samples a parameter $\tilde{\mu}$, from the posterior distribution $\mathcal{N}\left(\hat{\mu}, \sigma^2 D^{-1}\right)$ (Line 3). It then selects an action that yields the best sample (Line 4) and observes an associated reward (Line 5). Finally, the parameters D and $\hat{\mu}$ are updated (Line 6 and Line 7). Then, we go for another iteration (Line 8). The above process is repeated for a maximum number of time steps J (Line 9).

V. Evaluation

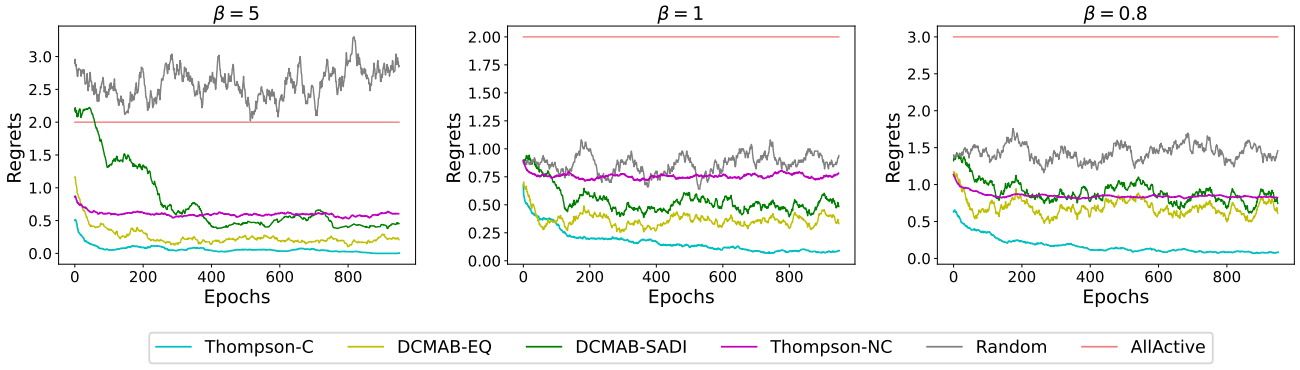
In this section, we evaluate the performance of our proposed solutions. We start with a description of the dataset and simulation environment. Afterwards, we explain the benchmark approaches and implementations that we use. Finally, we discuss the overall results.

A. Dataset and Simulation Setup

We evaluate the proposed solutions using a real-world dataset collected from the Orange 4G network, in Poitiers, France. The dataset includes mobile data traffic demand of different mobile applications at a base station level. We assume slice instances are deployed on an application-basis, i.e., one application maps to one slice instance. More precisely, Facebook, YouTube, and Google have been considered as three different types of slice instances attached to each base station. It is worth mentioning that those three different applications exhibit very different traffic demands, which is appropriate for the simulation of network slicing [26]. On the user side, we assume stochastic delay requirements of users for each application, as indicated in Table I.



(a) Reward.



(b) Regret.

Fig. 2: Reward and regret obtained for different β values.

TABLE I: LIST OF PARAMETERS

Parameter	Value
P_b^{static} [24]	18 Watts
P_b^{fixed} [25]	139 Watts
$P_b^{dynamic}$ [25]	742 Watts
ψ_i [Facebook, YouTube, Google, EcoSlice]	[1.2, 1.6, 1.4, 1]
δ_i [Facebook, YouTube, Google, EcoSlice]	[10, 1, 15, 11] ms
Number of users per slice	[11-30]
Users delay requirement d_u^τ	Facebook: [11-20]ms YouTube: [6-17]ms Google: [16-25]ms
Loss function	MSE
Learning rate α	0.001
β	[5,1,0.8]
Maximum episodes J	1000
φ	0.5
M	0.01

The granularity of the dataset is 10 minutes, for 10 days in May 2019. With this, for our simulation purposes, we consider T is 10 days and SADI $\tau = 10$ $minutes$. Thus T includes 1440 τ. We analyse 10 base stations from the Poitiers city center, where different slice instances are associated. We then apply our proposed solutions using an action set where each action implies the activation/deactivation of one of the slices: [Facebook, YouTube, Google, EcoSlice].

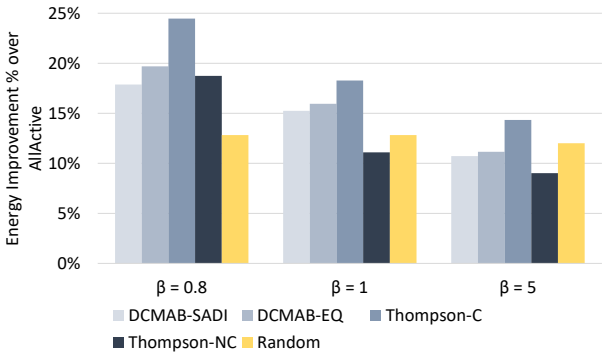


Fig. 3: Energy improvement over AllActive

B. Benchmarks and Implementation Setup

We compare the performance of our proposed DCMAB and Thompson-C solutions with three counterparts: Thompson Sampling Non-Contextual (Thompson-NC), AllActive and Random. Unlike Thompson-C, no state/context information is considered in Thompson-NC [27]. For AllActive, as the name implies it, all the slice instances are active. And a random action is selected at each iteration for the Random approach.

For the implementation, we rely on the Pytorch framework for the DCMAB agents. All the models (i.e. DCMAB, Thompson-C, Thompson-NC, AllActive and Random) are

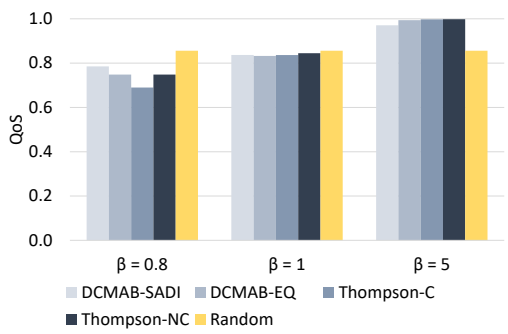


Fig. 4: QoS based on different β values

implemented using a Python environment and trained on the high-performance Linux server provided by Digital Research Alliance of Canada. The DNN of a DCMAB agent is composed of three fully-connected layers of 100 neurons, followed by a RELU activation function for each. The detailed parameters are summarized in Table I.

C. Results

1) Overall Agents Performances: First of all, to fully comprehend the performance of our agents, we study the trends of reward, regret, QoS and energy for $\beta = [5, 1, 0.8]$. We note that agents focus on QoS when $\beta = 5$, search for a trade-off when $\beta = 1$, and stress on energy when $\beta = 0.8$. Therewith, we compare the reward trends of our proposed solutions and their peers in Figure 2. The curves are smoothed by averaging within a rolling window of 50 iterations.

In Figure 2a, the DCMAB-EQ and Thompson-C agents exhibit the superior reward, even significantly better than AllActive, followed by DCMAB-SADI and Thompson-NC. Random approaches failed our proposed solutions in every scenario. Noticeably, AllActive shows inferior performance to that of the other agents in general. We also perceive the same behavior for the regret trends for all the agents in Figure 2b.

2) Roles of Agents in Energy and QoS: We then explicitly verify if our proposed solutions are feasible for energy optimization in RAN slicing by comparing them with the baselines. In this regard, as depicted in Figure 3, compared with the AllActive strategy (currently the standard approach), the energy improvement of Thompson-C is approximately 24%, 18% and 14% respectively, for β equals 0.8, 1 and 5. As expected, the energy gain deteriorates when β value increases. The prior phenomenon holds for all the agents (except Random). Also, we saw a great deal of energy gains over AllActive for DCMAB-EQ, DCMAB-SADI, Thompson-NC and even Random as well. Overall, we notice the Thompson-C agent is quite superior to others in terms of energy gain. Conversely, Thompson-NC has slightly lower energy gains than the DCMAB agents for all β values.

We are all aware that optimizing energy consumption means compromising QoS to an extent. In this light, we visualize the QoS of all the agents in Figure 4. We note that Thompson-C exhibits a comparable performance with its fellows in terms of QoS, but a slightly lower performance when $\beta = 0.8$. This is linked to Thompson-C showing highest energy gain at $\beta = 0.8$. It is observed in Figure 4 that our proposed solutions satisfy almost 100% QoS at $\beta = 5$ (slightly lower for DCMAB-SADI). Therewith, it is at $\beta = 5$ where our agents make themselves stand out, as they deliver the same QoS as AllActive, while providing significant improvement in energy efficiency. Regardless of showing acceptable performance, the Thompson-NC agent shows lower performance than our proposed solutions, comforting our modelling choices. We can not help but stress that: *i)* state-aware agents outperform the one in which context/state is not considered, and *ii)* a DNN approach is constantly surpassed by a statistical approach.

3) Impact of EcoSlice: To grasp the benefits EcoSlice, we examine in Figure 5 the performance of the Thompson-C agent with and witout an EcoSlice, in terms of reward, regret, QoS and energy utilization. We select the Thompson-C algorithm here as it shows the best results in most of the scenarios compared to its fellows. As we can observe, Thompson-C demonstrates better performance under the different metrics when compared to Thompson-C (w/o EcoSlice). Therefore, an EcoSlice significantly enhances the overall energy efficiency of the network by allowing operators to switch off the underutilized slice instances and yet ensure QoS. Without the assistance of an EcoSlice, one can not reach the level of energy efficiency that we have accomplished.

4) Computing Time Comparisons: The detailed computing time comparisons conducted on the Digital Research Alliance of Canada servers are shown in Figure 6. Paying the price for its performance, Thompson-C takes much longer computing time than all the other agents. Despite Thompson-C being considered to be a better agent than DCMAB-EQ, it is not a good option for the real-time decision-making process, at least not with a SADI of 10 minutes, as we consider. On the other side, the DCMAB agents, who also outperformed the baselines in terms of energy efficiency, are on par with Thompson-NC in terms of computing power. There is no single answer here, and Thompson-C can be a favourable solution for a system without computing time constraints. However, all the solutions we proposed in this work offer avenues for operators to optimize the energy efficiency by slice activation/deactivation while controlling the impact on QoS. Operators can also opt between different design choices by varying β, based on their targets and limitations.

5) MSE of DCMAB: Last but not least, we also compare the MSE of the two DNN-based solutions, DCMAB-EQ and DCMAB-SADI, in Figure 7. As we can see, DCMAB-EQ has a stable training process with lower MSE than DCMAB-SADI. This explains the superior performance of DCMAB-EQ in the prior results, as DCMAB-EQ predicts better reward distribution of associated actions. In Figure 7, MSE results are shown for $\beta = 5$ only, but we noticed similar behaviour for other tested values.

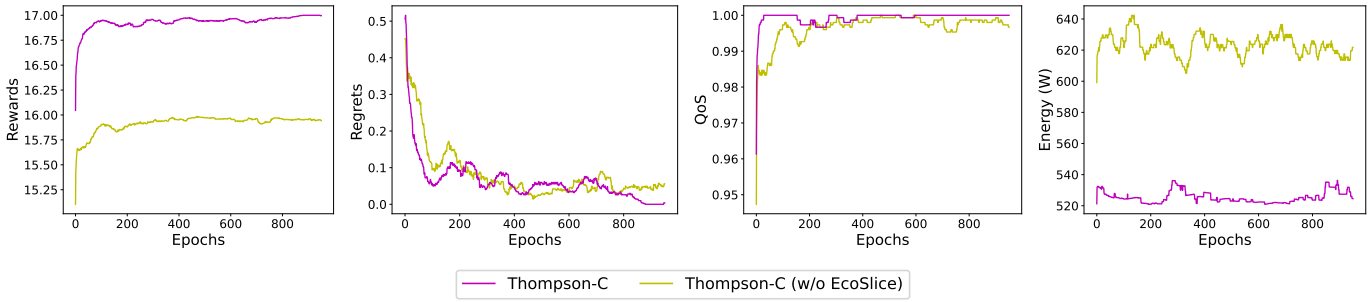


Fig. 5: Comparisons of Thompson-C Vs Thompson-C (w/o EcoSlice)

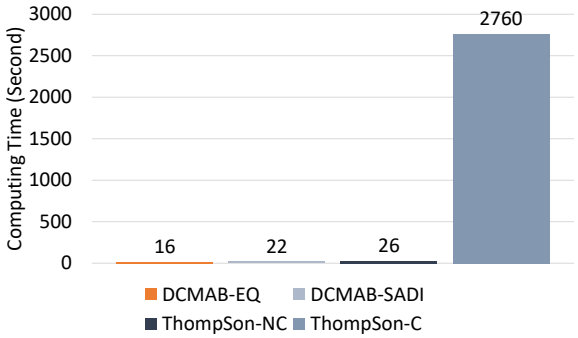


Fig. 6: Computing time comparisons

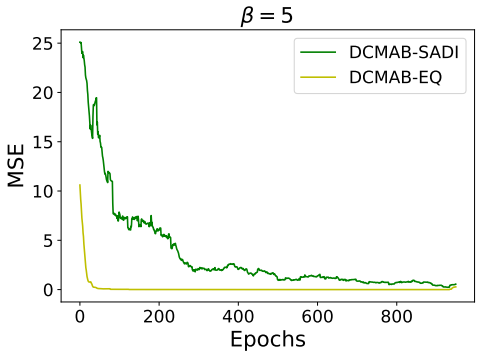


Fig. 7: MSE comparisons of DCMAB-EQ Vs DCMAB-SADI

VI. Conclusion

In this paper, we focus on the slice activation/deactivation problem, to further enhance the energy efficiency in RAN slicing. To this end, we advocate the state-aware MAB approaches (i.e., DCMAB and Thompson-C), where an agent attempts to activate the optimal slice instances while providing guaranteed QoS. More than anything else, we investigate the important aspect of the compromise between energy consumption and user QoS. The results are derived based on a real-world datasets and demonstrate that the MAB approach in general, and DCMAB and Thompson-C in particular, are appropriate for the slice activation/deactivation problem. They significantly alleviate the energy consumption at the base station level while ensuring a satisfactory QoS level.

References

[1] B. Cubukcuoglu, "The Importance of Environmental Sustainability in Telecom Service Providers' Strategy," *Risk, Reliability and Sustainable Remediation in the Field of Civil and Environmental Engineering*, pp. 249–254, 2022.

[2] N. Jones, "The Information Factories," *Nature*, no. 561, pp. 163–167, 2018.

[3] NGMN, "Green Future Networks - Network Energy Efficiency," NGMN, Tech. Rep., 2021.

[4] Y. Fan, B. Wang, J. Wei, M. Tan, and H. Ran, "A Hierarchical Distributed Operational Framework for Renewables-Assisted 5G Base Station Clusters and Smart Grid Interaction," *Frontiers in Energy Research*, vol. 10, pp. 1–4, June 2022.

[5] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[6] M. Masoudi, O. T. Demir, J. Zander, and C. Cavdar, "Energy-Optimal End-to-End Network Slicing in Cloud-Based Architecture," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 574–592, April 2022.

[7] N. Piovesan, D. López-Pérez, A. D. Domenico, X. Geng, H. Bao, and M. Debbah, "Machine Learning and Analytical Power Consumption Models for 5G Base Stations," *IEEE Communications Magazine*, vol. 60, no. 10, October 2022.

[8] F. Han, Z. Safar, and K. J. Liu, "Energy-Efficient Base-Station Cooperative Operation with Guaranteed QoS," *IEEE Transactions on Communications*, vol. 61, no. 8, pp. 3505–3517, 2013.

[9] M. Feng, S. Mao, and T. Jiang, "Base Station On-Off Switching in 5G Wireless Networks: Approaches and Challenges," *Wireless Communications*, vol. 24, no. 4, p. 46–54, January 2017.

[10] A. Chatzipapas, S. Alouf, and V. Mancuso, "On the Minimization of Power Consumption in Base Stations using On/Off Power Amplifiers," *Proc. IEEE Online Conference on Green Communications (GreenCom)*, pp. 18–23, 2011.

[11] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Energy-Efficient Deep Reinforcement Learning Assisted Resource Allocation for 5G-RAN Slicing," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 1, pp. 856–871, 2022.

[12] F. Rezazadeh, H. Chergui, L. Christofi, and C. Verikoukis, "Actor-Critic-Based Learning for Zero-touch Joint Resource and Energy Control in Network Slicing," *Proc. IEEE International Conference on Communications (ICC)*, 2021.

[13] O. Akin, U. C. Gulmez, O. Sazak, O. U. Yagmur, and P. Angin, "GreenSlice: An Energy-Efficient Secure Network Slicing Framework," *Journal of Internet Services and Information Security*, vol. 12, no. 1, pp. 57–71, 2022.

[14] H. Chergui, L. Blanco, L. A. Garrido, K. Ramantas, S. Kuklinski, A. Ksentini, and C. Verikoukis, "Zero-Touch AI-Driven Distributed Management for Energy-Efficient 6G Massive Network Slicing," *IEEE Network*, vol. 35, no. 6, pp. 43–49, 2021.

[15] G. Sun, Z. T. Gebrekidan, G. O. Boateng, D. Ayepah-Mensah, and W. Jiang, "Dynamic Reservation and Deep Reinforcement Learning Based Autonomous Resource Slicing for Virtualized Radio Access Networks," *IEEE Access*, vol. 7, pp. 45 758–45 772, 2019.

[16] C. Marquez, A. Banchs, M. Gramaglia, C. Ziemlicki, M. Fiore, and Z. Smoreda, "Not All Apps Are Created Equal: Analysis of Spatiotemporal Heterogeneity in Nationwide Mobile Service Usage," *Proc. 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, pp. 180–186, 2017.

[17] W. Guan, X. Wen, L. Wang, Z. Lu, and Y. Shen, "A Service-Oriented Deployment Policy of End-to-End Network Slicing based on Complex Network Theory," *IEEE Access*, vol. 6, pp. 19 691–19 701, 2018.

[18] A. K. Bairagi, M. S. Munir, M. Alsenwi, N. H. Tran, S. S. Alshamrani, M. Masud, Z. Han, and C. S. Hong, "Coexistence Mechanism between eMBB and uRLLC in 5G Wireless Networks," *IEEE Transactions on Communications*, vol. 69, no. 3, pp. 1736–1749, 2021.

[19] M. Bennis, M. Debbah, and H. V. Poor, "Ultrareliable and Low-Latency Wireless Communication: Tail, Risk, and Scale," *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834–1853, 2018.

[20] C. Marquez, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Resource Sharing Efficiency in Network Slicing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 909–923, 2019.

[21] H. P. Phyu, D. Naboulsi, and R. Stanica, "Machine learning in network slicing—a survey," *IEEE Access*, vol. 11, pp. 39 123–39 153, 2023.

[22] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009, pp. 215–216.

[23] A. Slivkins, *Introduction to Multi-Armed Bandits*. Foundations and Trends in Machine Learning, 2019, pp. 5–6.

[24] M. M. Aftab Hossain, C. Cavdar, E. Bjornson, and R. Jantti, "Energy Saving Game for Massive MIMO: Coping with Daily Load Variation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2301–2313, 2018.

[25] B. Debaillie, C. Desset, and F. Louagie, "A Flexible and Future-Proof Power Model for Cellular Base Stations," in *IEEE 81st Vehicular Technology Conference (VTC Spring)*, 2015, pp. 1–7.

[26] H. P. Phyu, D. Naboulsi, and R. Stanica, "Mobile Traffic Forecasting for Network Slices: A Federated-Learning Approach," in *IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2022, pp. 745–751.

[27] S. Agrawal and N. Goyal, "Near-Optimal Regret Bounds for Thompson Sampling," *Journal of the ACM*, vol. 64, no. 5, 2017.