Reinforcement Learning for Delay-Constrained Energy-Aware Small Cells with Multi-Sleeping Control

Ali El Amine*, Paolo Dini†, and Loutfi Nuaymi‡
*Orange Labs, Belfort, France
†CTTC/CERCA, Castelldefels, Barcelona, Spain
‡IMT Atlantique, IRISA, UMR CNRS 6074, F-35700 Rennes, France
e-mail: ali.elamine@orange.com, loutfi.nuaymi@imt-atlantique.fr, and paolo.dini@cttc.es

***Abstract*—In 5G networks, specific requirements are defined on the periodicity of Synchronization Signaling (SS) bursts. This imposes a constraint on the maximum period a Base Station (BS) can be deactivated. On the other hand, BS densification is expected in 5G architecture. This will cause a drastic increase in the network energy consumption followed by a complex interference management. In this paper, we study the Energy-Delay-Tradeoff (EDT) problem in a Heterogeneous Network (HetNet) where small cells can switch to different sleep mode levels to save energy while maintaining a good Quality of Service (QoS). We propose a distributed Q-learning algorithm controller for small cells that adapts the cell activity while taking into account the co-channel interference between the cells. Our numerical results show that multi-level sleep scheme outperforms binary sleep scheme with an energy saving up to 80% in the case when the users are delay tolerant, and while respecting the periodicity of the SS bursts in 5G.**

***Index Terms*—Energy saving, delay, multi-level sleep mode, 5G, Q-learning**

I. Introduction

In order to support future traffic requirements (high-speed mobile data services and better coverage), next generation cellular networks are expected to deploy denser mobile access networks with different Base Station (BS) sizes (large and small cells). Although this improves the capacity and coverage of these networks, it also brings new challenges such as pushing the limits of energy consumption and problems related to interference management. In order to meet these challenges, there is a growing interest to steer research efforts towards energy-efficient solutions, with particular emphasis on BS Sleep Mode (SM) techniques.

BS sleep scheme is considered among the best methods to save energy, since it does not require changes to current network architecture, and it is easy to implement [1]. We distinguish between two types of sleep schemes: binary SM level (ON and OFF) and multi-level SM. In binary SM, the BS shuts down completely when the traffic is low to save energy. The literature is rich and many surveys exist on binary SM [1]–[4]. On the other hand, multi-level SM splits the sleep states into several levels with different characteristics on the activation/deactivation period and the power consumption [5]–[7]. The latter adds flexibility to cope with the traffic requirements to further enhance the system performance by reducing the energy consumption of future networks while minimizing the impact on the Quality of Service (QoS). In this regard, there exist several studies that consider the problem of Energy-Delay Tradeoff (EDT) using multi-level SM scheme [8]–[10]. By applying tools from machine learning, and in particular reinforcement learning, these works study the multi-objective function of the EDT problem for different SM policies depending on the service use case, allowing the operator to tune the network parameters towards more energy savings or less service delay.

Coupled with the problem of energy consumption in dense Heterogenious Networks (HetNets) is the co-channel interference, where the unplanned deployment of small cells makes the interference management more complex. In [11], the authors designed a centralized Fuzzy Q-Learning (FQL) algorithm controller for small cells to minimize the energy consumption while respecting the users' QoS. The proposed solution manages the interference between the small cells in order to limit its effect and to maximize the energy efficiency of the network.

In this paper, we study the problem of EDT in a HetNet where small cells can switch to different SM levels in order to save energy. We propose a distributed Q-learning algorithm controller for small cells that adapts their activity based on the level of interference, the traffic load and the size of the buffered data in the cell. Different from the works in [8]–[10], in this paper we take into account the co-channel interference between the cells to minimize its effect to enhance the network's performance. Different from [11], we propose a distributed Q-learning algorithm where each cell makes autonomous decisions according to the Decentralized SON (D-SON) paradigm. Furthermore, we extend the model of [11] to multi-SM scheme. Our numerical analysis shows that multi-level SM scheme outperforms binary sleep scheme in terms of energy savings and added delay. We also focus on the convergence analysis, and the impact of the convergence time on the tuning parameter.

The remainder of the paper is organized as follows. Section II details the system model understudy with the 5G multi-level SM scheme. In Section III, we present our distributed Q-learning Small Base Station (SBS) controller algorithm for multi-level SM scheme. We discuss some performance results in Section IV. In Section V, we draw our conclusions and discuss future research directions.

II. System model

We consider a two-tier heterogeneous multi-cell network composed of a Macro Base Station (MBS) and SBSs. Together, these BSs serve a total of K users. The SBSs are densely deployed in the coverage area of the macrocell and operate in a dedicated carrier to create a local hotspot that will increase the system capacity in the designated area, where needed. In order to enhance the system energy efficiency, these SBSs can dynamically switch to different SM levels. This will not only save energy and limit the CO_2 footprint, but it will also enhance the system capacity by limiting the interference between neighboring cells. When in SM, the SBS only listens to incoming traffic. A distributed local controller is in charge of managing the activity of these cells (see Fig. 1). We note that for binary SM levels (ON and OFF), finding the optimal set of active SBSs is a combinatorial problem with high computational complexity. In this work, we further add another layer of complexity by considering multi-level SMs. In order to solve this problem, the local controller manages a given SBS by means of interacting with the environment in order to map what to do (action) when faced with a current situation (state). This method is known as Q-Learning. In particular, each SBS at each time slot, measures the expected average interference from neighboring cells and the SBS state in the previous iteration. The local controller uses this information along with the SBS buffer and traffic load states to control its activity, as we detail later in this section.

A. Multi-Level Sleep Modes

In [5], GreenTouch identified four distinct SM levels by grouping sub-components with similar transition latency when being activated or deactivated. The presented model enables to quantify the power consumption of the BS in each of the four SMs. These are:

- SM 1: It considers the shortest time unit of one OFDM symbol (i.e., $71\mu s$) comprising both deactivation and reactivation times. In this mode only the power amplifier and some processing components are deactivated.

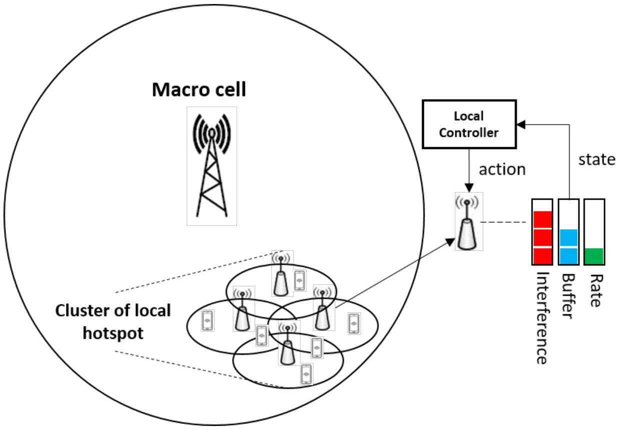

Fig. 1: Network architecture with a distributed controller.

- SM 2: It corresponds to the case of sub-frame or Transmission Time Interval (TTI) (i.e., 1 ms). In this SM, more components enter the sleep state.
- SM 3: It corresponds to the frame unit of 10 ms. Most of the components are deactivated in this mode.
- SM 4: This is the deepest sleep level. Its unit corresponds to the whole radio frame of 1s. It is the standby mode where the BS is out of operation but retains wake-up functionality.

Higher energy savings can be achieved when switching BSs to a deeper SM, since more components will be deactivated. However, this will be associated with longer transition latency which may impact the QoS of the system. In Table I, we present the SM levels characteristics.

Along with SM and users' dynamics, the BS has to wake up periodically to send signalling bursts. In contrast to Long Term Evolution (LTE) systems where each antenna must transmit every 0.2 ms a unique Cell Reference Signals (CRS) for channel quality estimates and mobility measurements among other Synchronization Signaling (SS), no CRS is required for 5G [12]. Instead, SS and Physical Broadcast CHannel (PBCH) are transmitted in SS/PBCH block periodically. It has been agreed in Third Generation Partnership Project (3GPP) [12] that this periodicity can be set to any value among [5, 10, 20, 40, 80, 160 ms]. With these values, SM 4 cannot be used. Hence, we limit our work to the first three SM levels.

TABLE I: BS Sleep Modes Characteristics [5].

Sleep level	**Deactivation duration**	**Minimum sleep duration**	**Activation duration**
SM 1	35.5 μs	71 μs	35.5 μs
SM 2	0.5 ms	1 ms	0.5 ms
SM 3	5 ms	10 ms	5 ms
SM 4	0.5 s	1 s	0.5 s

B. Network Operation

In our model, we consider connected users continuously requesting packets during a fixed period of time. We use the term block of data to describe the overall packets requested by users during a time slot t (ms) and contained in one block. In this work, we do not focus on the inter-packet dynamics. We rather consider a continuous flow of data with an average size (kbits) that is first buffered in the SBS buffer before being transmitted to the designated users.

At each time slot t, a block of data (composed of several packets) is buffered and stacked in the designated cell buffer. The size of this block n depends on the number of users being served in the cell and users' requirement based on the service. Furthermore, this block of data is characterized by a Time-To-Live (TTL) l (ms), which is the latency constraint above which the remaining data are lost. Here, we consider that the block of data can be partially served in case the serving time was not enough to deliver it all, due to high interference and/or inactivity of the cell (i.e., in sleep mode). Furthermore, the cell buffer operates in FIFO (First Input First Output) fashion. One can relate such service to a current or future 5G delay-tolerant application where the order of the packets does not matter.

If a user requests a packet from a SBS in a SM state, e.g., SM 1, SM 2, or SM 3, it will be buffered until the SBS activates in a future time step. Thereafter, the SBS manages the available radio resources to serve the users during the transmission time slot. The state of the SBS has an impact on the latency added to the system. The deeper the SM level is, the more time the user will have to wait until the SBS reactivates.

C. Problem formulation

Following the above system description, we can model the problem as a discrete-state Markov Decision Process (MDP), where a SBS m in a state $s_m(t)$ takes an action $a_m(t)$ and transitions to another state $s_m(t+1)$. We denote by $\mathcal{S} = \{R, B, I_l\}$ the state space of the SBS, where R, B and I_l are the sets related to the cell throughput, buffer state, and the estimated spectral efficiency loss (bit/s/Hz) due to the interference from nearby active SBSs, per SBS, respectively. In this work, we assume that the users request a Near Real Time Service (NRTS) traffic with an average packet size [kbits]. The time needed to complete this service depends on two factors: The buffering time until the SBS re-activates, and the capacity of the serving SBS that is highly affected by the co-channel interference from neighboring cells.

At each time slot, the local controller decides an action $a_m(t) \in \mathcal{A} = \{Active, SM1, SM2, SM3\}$ for a SBS after observing its state. In order to evaluate the cost associated with each state-action pair, we use a function that takes into account the power consumption of the SBS, and the users' QoS:

$$c(s,a) = (1-w) \cdot p(s,a) + w \cdot d(s,a) \tag{1}$$

where $p(s,a)$ is the power consumption of the SBS, w is a weighting factor that prioritizes between the power consumption and the QoS that is defined by $d(s,a)$ that indicates the data lost due to exceeding the delay constraint either from the operational state of the cell (i.e., in sleep) or the high interference level limiting the cell throughput.

In order to solve the above MDP, we first need to derive the transition probabilities and cost from one state to another. Then, a backward dynamic programming needs to be performed to evaluate the optimal value function, in order to derive the optimal policy. However, this requires a complete knowledge of the system information. To alleviate the non-causal system information, we rely on QL to find the optimal policy by only exploring and interacting with the current state of the system.

III. Distributed Q-Learning

A. Preliminaries on Reinforcement Learning

Distributed Q-learning is an online optimization technique that aims at controlling multi-agent systems, i.e., a system featuring M BSs which take decisions (select the appropriate SM level) in an uncoordinated fashion. Each BS has to learn independently a policy (SM 1, SM 2 or SM 3) through real-time interactions with the environment. Q-learning finds the optimal policy in the sense that it maximizes the expected value of the total reward (Q-value) over all successive episodes. The agents (i.e., BSs) have a partial view of the overall system, and their actions may differ since the users are unevenly distributed over the network. In particular, the decision of a BS to choose a SM level is affected by how many users it has to serve, and on how delay-tolerant these users are.

In Q-learning, each agent takes an action a_m^t from an action set $\mathcal{A}$, then moves to a new state s_m^{t+1} while receiving a reward r_m^t (in this work, the reward r is the cost function in 1). This reward is then used to update the Q-value locally, $Q(s_m^t, a_m^t)$, indicating the level of convenience of selecting action a_m^t when in state s_m^t. The Q-value is updated following the update rule:

$$\begin{aligned} Q(s_m^t, a_m^t) \leftarrow {} & Q(s_m^t, a_m^t) + \\ & \alpha[r_m^t + \gamma \max_{a \in A} Q(s_m^{t+1}, a_m^{t+1}) - Q(s_m^t, a_m^t)] \end{aligned} \tag{2}$$

where α is the learning rate that represents the speed of convergence, and $\gamma \in [0,1]$ is the discount factor that determines the current value of the future state costs.

During the learning phase, each agent selects the corresponding action based on the ϵ-greedy policy, i.e., it selects with probability $1-\epsilon$ the action associated with the maximum Q-value, and with probability ϵ selects a random action (here, $y \in [0, 1]$ is a random variable):

$$a_m^t = \begin{cases} \underset{a\in\mathcal{A}}{\operatorname{argmax}}\, Q(s_m^t, a_m^t), & \text{if } y > \epsilon \\ rand(\mathcal{A}), & \text{otherwise} \end{cases} \quad m = 1, \dots, M. \tag{3}$$

By implementing the ϵ-greedy policy, the BS would have explored all possible actions and avoided local minima. For more details on Reinforcement Learning (RL) and Q-learning the reader is referred to, e.g., [13].

B. Q-Learning Algorithm

At time slot t, the local state of the small cell m is $s^m(t) = \{r^m(t), b^m(t), i_l^m(t)\}$. The states have been quantized each into 3 levels. We chose these levels of quantization to balance between performance and complexity. In order to explore different combinations of states, we make the users move inside their serving cell at each time slot. We define an episode as a fixed simulation run where the users in motion request data continuously with a given minimum rate requirement. During this episode, the users move in a predefined and deterministic pattern, and the algorithm explores different states to find the optimal policy that minimizes the cost function in (1). After the training phase is complete, the system goes online and operates in real-time. The Q-Learning algorithm is described in Algorithm 1.

Algorithm 1 : Q-Learning Algorithm

```
procedure Training ($Q_T^m(s,a)$)
Initialize the positions of the users, their trajectory and $q^m(s,a) = 0, \forall m \in \mathcal{M}, \forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}$.
Set the weight $w$, and the average user velocity.
while Learning do
    for $m \in \mathcal{M}$ do
        Visit state $s^m$.
        Select an action $a^m$ using $\epsilon$-greedy rule in (3).
        Calculate the cost $c^m$.
        Observe next state $s'^m$.
        Update the Q-value $q^m(s,a)$ from (2).
    end for
end while
end procedure

procedure Online
for $m \in \mathcal{M}$ do
    $Q^m(s,a) = Q_T^m(s,a)$
    Run Q-Learning.
end for
end procedure
```

IV. Numerical Analysis

In this section, we evaluate the proposed Q-learning SBS controller through numerical results. We simulate using MATLAB a scenario of one macro BS covering a cluster of 4 SBSs. NRTV traffic is simulated for users located randomly in the cluster. We assume an average user rate and latency constraints equal to 1 Mbps and 100 ms, respectively. We summarize the main simulation parameters in Table II.

TABLE II: Parameters' values and assumptions.

Parameters	Values
Number of SBSs	4
SBS radius	40 m
Bandwidth	10 MHz
Maximum SBS transmitted power	30 dBm
Min. dist. SBS-SBS	20 m
Min. dist. SBS-user	5 m
Number of users K	30
User required rate	1 Mb/s
SBS carrier frequency	3.5 GHz
Small cell path loss	Umi with LOS [14]
Shadowing distribution	Log-normal
Simulation runtime	10 seconds

A. Convergence analysis

We analyze the convergence of the proposed Q-learning algorithm for different values of w during the training phase in figures 2. During this phase, the system is fed with a training episode that describes the dynamics of the network: the users are moving in the designated area, they are requesting service packets, and the small cells are reacting (i.e., taking action to switch to different modes) based on their partial view of the environment that defines their local states (cell throughput, cell buffer size and estimated spectral efficiency loss). By repeating this episode a number of epoch of times, each cell should have learned how to map its observable states to actions. The design of the training episode is important so that the agent gathers enough experience from the environment, which is critical in stabilizing the Q functions, thus forging the best policy.

In Fig. 2, we observe that the system converges after few iterations (around 10 iterations for most values of w). When $w = 0$ and $w = 1$, the equation in (1) simplifies to minimizing energy consumption and service delay only, respectively. In this special case, the agent (i.e., small cell) is able to find the optimal policy relatively fast compared to other values of w. When $w \in]0, 1[$, the agent needs more time to capture the tradeoff between energy and delay, thus requiring more epochs to finally converge. We note that for $w = 1$, the cost function in 1 reduces to minimizing the data lost $d(s, a)$. As a result, the Q-value converges to zero.

Since the convergence time of the training phase is fast as previously shown in Fig. 2, it is possible to skip the independent training phase and merge it with the online one. In this case, the system starts online with no previous knowledge about the environment and learns to

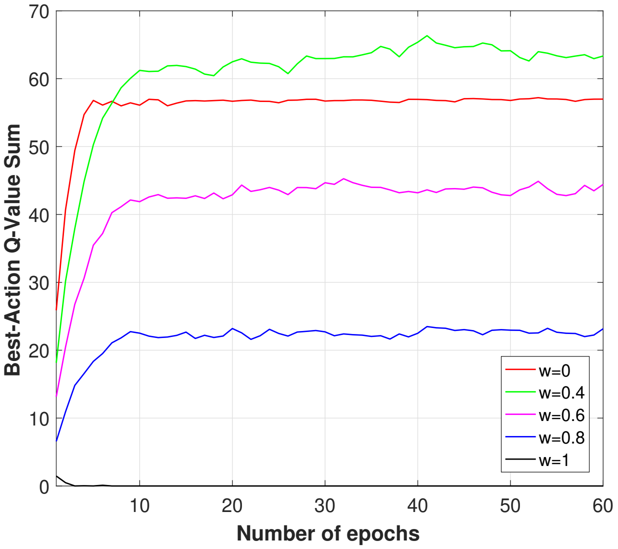

Fig. 2: Evolution of the convergence of the Q-Learning offline algorithm.

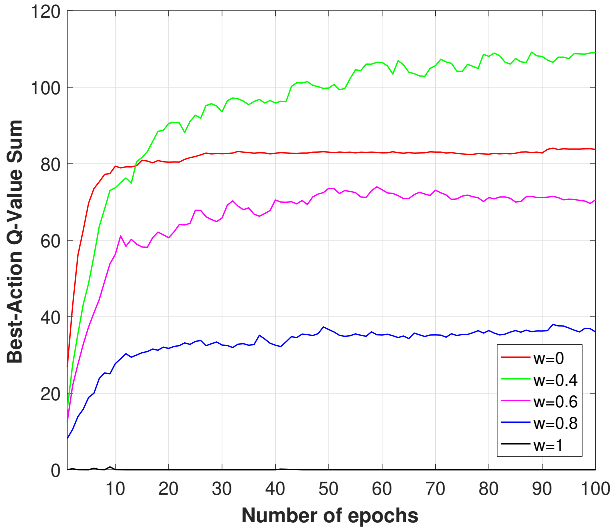

Fig. 3: Evolution of the convergence of the Q-Learning online algorithm.

adapt on the spot. The main advantage of this approach is that it does not require the prerequisite training step. However, it extends the time until the system converges. This can be observed in Fig. 3 where we notice an increase in the number of iterations required for the Q-functions to stabilize. For instance, we observe that for the case of $w = 0.4$, we do not reach convergence for the defined number of epochs. However, it is interesting for the other cases of w. The major reason behind this increase is that with the online mode, the episode is changing in each iteration in contrast to the training phase where the training episode is fixed. This dynamic nature of the iteration makes it harder for the agent to learn as opposed to static episode.

B. *Policy analysis*

In Fig. 4, we report the different policies of the SBSs for different values of the tuning parameter w. We observe that the system tends to save more energy when w approaches zero by switching to the deepest SM most of the times, while a gradual shift to active mode is observed when increasing w. However, it is interesting to highlight the behavior for $w = 1$. For this case, the cells are required to be active for a minimum period of time to minimize the delay (around 37% of the time). During the remaining time and after satisfying the agent's goal, the cell switches randomly to any of its states. This can be observed by the uniformly distributed state modes for SM 1, SM 2 and SM 3, while the active mode has a higher probability in order to satisfy the delay constraint of 100 ms.

In Fig. 5, we illustrate the average percentage cell energy consumption normalized to the case where the cell is always on (i.e., without sleep mode) and the average small cell delay as a function of w with packet delay constraint of 100 ms. We compare our proposed Q-learning algorithm (QL) with another QL algorithm that considers two states only (active and SM 3). This algorithm is referred to as binary SM-QL algorithm. We observe that our multi-SM-QL algorithm outperforms the binary QL algorithms for the different values of w. This is due to the added flexibility with multi SM levels on saving energy. By tuning the value of w, the operator can shift the performance of his network towards either energy saving or QoS.

In order to analyze the performance of the policies, we report in Fig. 6 the average data lost due to the sleep scheme policy. Even though the average delay of the requested data falls under the delay constraint limit of 100 ms (as shown in Fig. 5), around half of these requested data are dropped for small values of w. This is because the Q-learning algorithm optimizes the data dropped that is indirectly related to the served delay. Hence, it is important to link served data delay with their drop rate.

V. Conclusions

Network densification with small cells is a key enabler in future 5G networks to support high data rate demands. However, it is critical to manage the activity of these cells in order to limit the network energy consumption. This paper has investigated an approach based on reinforcement learning to manage the activity of small cells to reduce the energy consumption while maintaining a good QoS. In this regard, we proposed a Q-learning algorithm that controls the states of these cells over different sleep mode levels that are compliant with 5G requirements with respect to the synchronization signals periodicity. The proposed algorithm allowed the operator

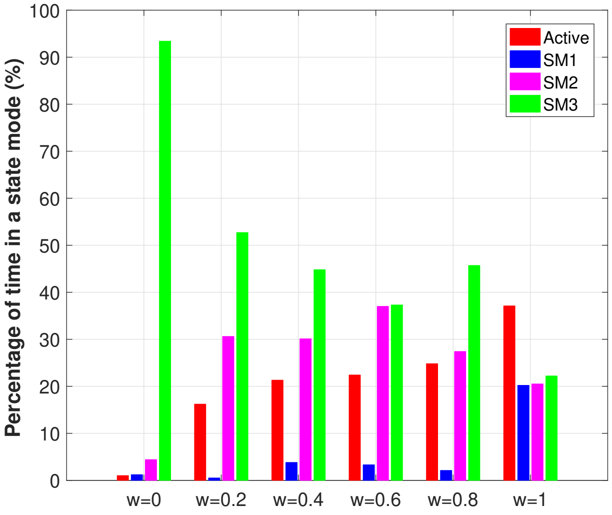

Fig. 4: Different SM policies as a function of w.

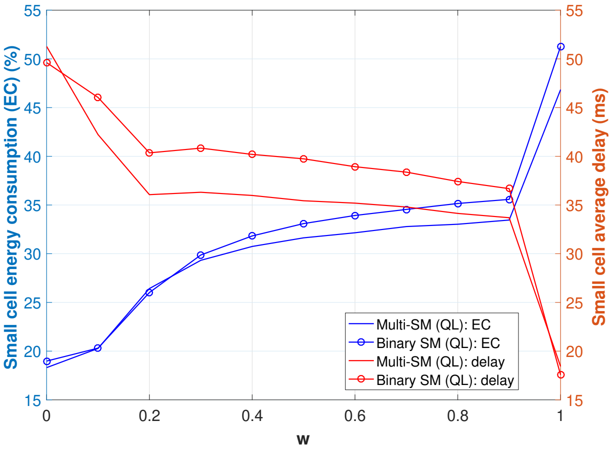

Fig. 5: Performance assessment and comparison as a function of w.

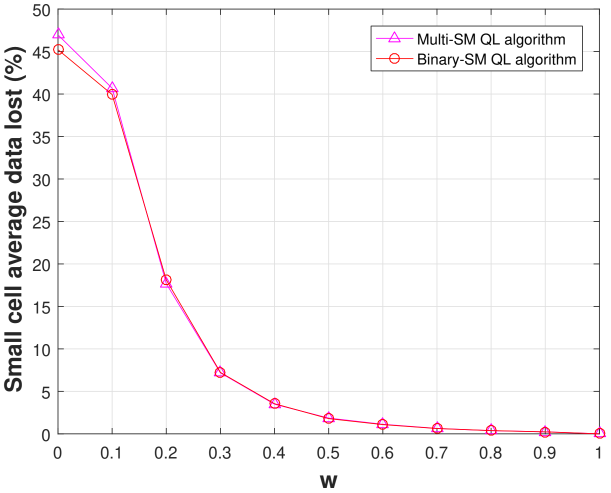

Fig. 6: Data block drop rate assessment.

to freely manage the tradeoff between energy saving and delay. Our simulations showed that given a delay constraint, the network is capable of saving at least 50% energy. Future work will focus on extending the model to include users offloading between cells including the macro cell.

Acknowledgments

This work has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 675891 (SCAVENGE).

References

[1] J. Wu, Y. Zhang, M. Zukerman, and E. K. Yung. Energy-efficient base-stations sleep-mode techniques in green cellular networks: A survey. *IEEE Communications Surveys Tutorials*, 2015.

[2] Ł. Budzisz, F. Ganji, G. Rizzo, M. Ajmone Marsan, M. Meo, Y. Zhang, G. Koutitas, L. Tassiulas, S. Lambert, B. Lannoo, M. Pickavet, A. Conte, I. Haratcherev, and A. Wolisz. Dynamic resource provisioning for energy efficiency in wireless access networks: A survey and an outlook. *IEEE Communications Surveys Tutorials*, 2014.

[3] F. Han, S. Zhao, L. Zhang, and J. Wu. Survey of strategies for switching off base stations in heterogeneous networks for greener 5G systems. *IEEE Access*, 2016.

[4] M. Feng, S. Mao, and T. Jiang. Base station on-off switching in 5G wireless networks: Approaches and challenges. *IEEE Wireless Communications*, Aug 2017.

[5] B. Debaillie, C. Desset, and F. Louagie. A flexible and future-proof power model for cellular base stations. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, May 2015.

[6] C. Liu, B. Natarajan, and H. Xia. Small cell base station sleep strategies for energy efficiency. *IEEE Transactions on Vehicular Technology*, Mar. 2016.

[7] P. Lähdekorpi, M. Hronec, P. Jolma, and J. Moilanen. Energy efficiency of 5G mobile networks with base station sleep modes. In *2017 IEEE Conference on Standards for Communications and Networking (CSCN)*, Sep. 2017.

[8] Fatma Ezzahra Salem et al. Reinforcement learning approach for advanced sleep modes management in 5G networks. In *2018 IEEE Vehicular Technology Conference (VTC-Fall)*, Jul. 2018.

[9] A. El-Amine, M. Iturralde, H. A. H. Hassan, and L. Nuaymi. A distributed Q-Learning approach for adaptive sleep modes in 5G networks. In *2019 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2019)*, Apr. 2019.

[10] A. El-Amine, H. A. H. Hassan, M. Iturralde, and L. Nuaymi. Location-Aware sleep strategy for Energy-Delay tradeoffs in 5G with reinforcement learning. In *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Istanbul, Turkey, 2019. IEEE.

[11] A. De Domenico and D. Kténas. Reinforcement learning for interference-aware cell dtx in heterogeneous networks. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, April 2018.

[12] 3GPP. On requirements and design of SS Burst Set and SS Block Index Indication. In *TS 38.300 Release 15*, 2017.

[13] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[14] 3GPP. Small cell enhancements for E-UTRA and E-UTRAN-Physical layer aspects (release 12). In *TR 36.872*, Dec. 2013.