Joint cell/antenna switch-off for energy savings at the Base Station

Pierre-Eliot Jourdan

August 31, 2023

Acknowledgements

This report summarizes my end-of-studies internship at the offices of Nokia Bell Labs in Massy, France. This project was a great opportunity for me to discover the world of research in the telecommunications industry and get a professional experience in the domain of AI in which I specialized during my studies at ENSEEIHT engineering school.

During this 6-months internship, I had the chance to work in a very welcoming team called Modeling & Optimization and lead by Matthew Andrews . I especially want to thank my supervisors Lorenzo Maggi and Ryo Koblitz with whom I spend a lot of time working and building the Reinforcement Learning model for energy saving. Their continuous involvement in the project was a booster for me to produce a solution to a major problem in which we had no idea what outcome to expect in the beginning.

I am grateful to Nokia for giving me the chance to be part of their research and innovation labs and I know the technical and social skills I have learnt during this internship will be very useful for my future professional career.

I finally want to thank my professors at ENSEEIHT thanks to whom I acquired the knowledge necessary for this professional project.

Table of contents

Abstract

Nowadays, for ecological, economic or geopolitical reasons, energy saving has become a ubiquitous goal in various areas. Telecommunications is one of them. Transmission stations today offer a much better quality of service than several years ago, but this improvement goes hand in hand with an increasing power consumption of these stations, particularly due to the growing number of users. To this can be added the difficulties in supplying electricity. This context explains why we witness a mindset shift away from maximum performance to sustainable performance in the telecom industry.

In the face of these new challenges, technologies are being developed to meet them, such as the upcoming deployment of 6G or the use of artificial intelligence, which proves to be a very useful tool in resource management problems. This is indeed the subject of this internship carried out at Nokia Bell Labs laboratory, Nokia's innovation hub.

This project completes some previous works done by Lorenzo Maggi and Ryo Koblitz. It aims at building a more developed Reinforcement Learning model in which several assumptions made are lifted.

The main goal of the project is to come up with a solution that dynamically realizes the best trade-off possible between reducing the energy consumption (EC) of a base station whilst guaranteeing a good quality of service (QoS) to the users connected to it. This project is very important for Nokia as it aims at becoming a Nokia Radio Access Network (RAN) feature in the near future.

This work culminated in an invention disclosure report, which is under review and may form the basis of a patent application, paving the way for adoption into product.

1 Introduction

As climate change and global warming are becoming frightening realities today, many solutions emerge in the industry in order to tackle these major issues. The main challenge is to conciliate the growing needs in digital technologies and the respect of our environment while living in a tense geopolitical context in which energy is now considered a precious resource.

Energy saving is now more important than ever if we are to build a sustainable world and society compatible with the ongoing development of digital technologies and the rise of compute energy requirements.

For a few decades the development of technology has enabled a growing number users to get access to the Internet all over the world. If it can be seen as a progress in many aspects, this has also increased substantially the energy consumption of telecommunication infrastructures providing this access. This analysis highlights the global importance of telecommunications in a more and more interconnected world. Many companies like **Nokia** believe telecommunications are a key climate change mitigation enabler and now highly focus their research on how to develop products that have good energy saving features.

Nokia Bell Labs is a world-renowned industrial research and scientific development company, owned by Nokia Corporation since 2016. It has been at the forefront of technological innovation for over 90 years, and has made significant contributions to the field of telecommunications and beyond. In the past, the Bell Labs laboratories have conducted research on topics like information theory, phones, satellites etc... which have lead to the emergence of 4 main areas of research : **Networks**, **Automation** (including robotics), **Semiconductors** & **Devices** and **AI** & **Software systems**.

Figure 2: Photo of Nokia Bell Labs historical site in Murray Hill (US)

As an intern in the offices of Bell Labs located in Massy, France, I was part of a team of 16 people called **Modeling** & **Optimization** and lead by Matthew

Andrews. The members of the team can work together although they work on other Bell Labs sites spread across the globe like in Murray Hill (US) or Cambridge (UK).

This department sits inside the AI Research Lab. The group works on several aspects related to robotics, industrial monitoring, knowledge extraction and wireless optimization. My internship is part of the group's research on energy efficiency for current and next wireless generations (from 4G to 6G).

The team works on problems that tend to come to them from Nokia Business Units, mainly **Mobile Networks**. These Business Units often solicit the department on problems they are either experiencing right now, in which case they need actionable solutions in the short term, or problems they will face in the future.

Today an important part of the research in telecommunications at Nokia Bell Labs is focused on the development of 6G that will provide a higher efficiency than 4G/5G standards. However the emergence of this technology in our everyday life is many years away from now and in spite of expected efficiency gains, energy consumption in absolute terms is likely to rise due to the expected exponential growth in user traffic. The advent of new technologies in our everyday life, such as flying connected vehicles for example, will be responsible for this traffic increase. As well, new hardware will result in new embodied carbon.

Therefore, it is necessary to start saving energy today, without needing new hardware, which can also be used in future networks.

The general class of problems this internship belongs to is **Radio Resource Management** (RRM). RRM is the system-level management of radio resources, and other radio transmission characteristics in wireless communication systems (cellular / wireless networks...).

RRM thinks of **system-level** optimization rather than low-level (physical layer) problems. Many system-level approaches exist for reducing site level energy consumption (see [1]) like building energy-efficiency infrastructures, using renewable energy sources or balancing increasing traffic with intelligent networks, which is the topic of this internship. All these approaches will pave the way for green communication and mobile networks in the future.

As said in the abstract, this project pursues previous state-of-the-art work conducted at Nokia Bell Labs explained in [2] and [3]. In this paper, the goal was to use **Bayesian optimization** methods to save energy at a base station by shutting down frequency carriers over time. The mission was also to avoid traffic congestion in the network and meet with the QoS requirements for the users. The researchers came up with a strategy to answer this problem and simulations have shown it could reduce the power consumption at a base station by 11%.

This legacy method of carrier shutdown mentioned before can be applied to a lower levels like antennas. We call this technique **MIMO-muting**. MIMO (Multiple Input and Multiple Output) is a method to increase the capacity of radio link with multiple transmission and multipath propagation (thanks to

antennas).

In order to develop this work, we will solve the similar problem using the MIMO-muting technique. Also, a few hypothesis made on the previous model are lifted in this project. In the state-of-the-art model, **time dynamics** are not taken into account i.e. it is assumed that when a carrier is switched ON/OFF, there is no latency and the BTS goes directly in the new configuration. As well, there is a **predefined order** for frequency carrier switch ON/OFF. These two assumptions have been lifted in this project to deal with a more realistic problem.

NB : Most of the technical terms used in this introduction will be defined in the next section.

• This internship is divided in two parts. **The first part** was dealt with during the first 4 months of the internship. It aims at solving our energy saving problem on a simple model (MDP) on which a part of the assumptions made on the state-of-the-art model were lifted. This part will give us an overview of the strategies that work well in practice.

In the second part of the internship, we tackle the real problem (with no assumptions at all) and use Reinforcement Learning methods to learn the strategies found in the previous part on a base station model closer to the reality. **Unfortunately this part was only covered during the 2 last months of the internship, so the simulations on the model haven't been carried out yet.**

2 Scenario

A telecommunications network is a huge chain of infrastructure (cables, routers, antennas...), spanning thousands of kilometers spread across the world, able to connect any user on the planet to another one, no matter the distance between them. As mentioned in the Introduction, in this project we work on a **Base Transceiver Station** (BTS).

A BTS is a telecom infrastructure gathering a set of antennas. The role of each antenna is to send and receive electromagnetic waves to mobile devices in order to connect them to the rest of the network. BTS are the final links connecting a device to the chain while processing the demands from hundreds of users connected to them, all trying to send and receive data at the same time.

Figure 3: Image of a Base Transceiver Station (BTS)

When a signal (a radio or electromagnetic wave) is emitted from a BTS, it is sent over a channel that covers a range of frequencies, as explained in [4]. A **frequency carrier** is a BTS component composed of antennas able to receive and send data on a specific frequency band. Each frequency carrier in the BTS covers a different range of frequency. A **Radio Unit** is another BTS component which can be seen as a collection of frequency carriers.

In this internship, we want to implement an energy-saving solution on a BTS with a nested structure, similar to that of a Russian doll, as depicted in figure 4. The BTS is composed of multiple Radio Units (RU) which may contain several frequency carriers (called cells). Each of these cells contains a set of antennas (also named TX) transmitting a signal to users connected to the BTS. To change the configuration in the BTS, it is possible to activate or deactivate RUs, cells or antennas over time.

A radio unit (RU) can have 3 possible configurations depending on the state of its cells. If at least one of its cells is activated then the RU is in **'Active'** mode. On the contrary, if all its cells are deactivated, then there are 2 possible modes for the RU: **'Idle'** or **'Deep sleep'**. In 'Idle' mode, the RU is on standby, its activation time is short but it still consumes some energy. On the other hand, in 'Deep sleep' mode, the RU is completely turned off, its energy consumption is almost zero but the time to activate it is much longer.

Similarly, there are several possible configurations for a cell of the BTS depending on the number of antennas turned ON within it. Either all antennas are turned OFF and the cell is inactive. Either some or all antennas are turned ON, defining $n_{antenna}$ active modes for the cell. Hereafter we will denote these modes of a cell respectively '0', '1',... '$n_{antenna}$'. In general, $n_{antenna} = 2$ for every cell.

An example of BTS architecture is visualized on the figure below :

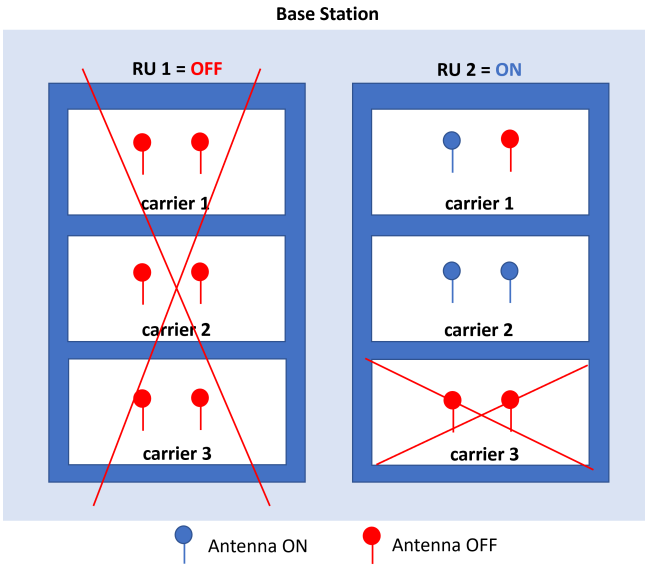


Figure 4: Example of BTS structure with 2 RUs (1 OFF on the left, 1 ON on the right) composed of 3 cells each

In this example, the BTS is composed of 2 RUs, each of them composed of 3 cells. The first RU is turned off, either in 'Idle' or 'Deep sleep' mode because all its cells are deactivated (mode '0'). The second RU is in 'Active' mode because 2 of its cells are active. Carrier 1 is in mode '1', carrier 2 is in mode '2'.

At each time interval, we evaluate the quality of service of the current configuration of the BTS by calculating a list of **Key Performance Indicators** called KPIs. The KPIs can evaluate various elements such as traffic volume,

latency, downlink / uplink throughput etc... The customer will fix limits of performance the KPIs will have to respect to provide a good QoS to the users.

For each new configuration of the BTS obtained at time t, we also compute the associated power consumption $c_t(n_t)$ (in Watts = Joules/s). It corresponds to the rate at which energy is consumed. We will often talk about **energy consumption** (EC) in this report. The energy consumption at time t depends on the configuration of the BTS i.e, on the number of RUs, cells or antennas turned ON at that time. It also depends on the traffic i.e. the number of users connected to the BTS at time t.

The objective is to dynamically optimize the BTS hardware configuration i.e, the number of active RU/cell/antennas over time, to achieve the wanted QoS-vs-BTS energy consumption trade-off.

However, this problem is complex for various reasons.
First of all, the problem is **combinatorial**, meaning the dimensionality can quickly explode when adding RUs or cells to the structure of the BTS. In most of the simulation in this report, the tests were run on a simple but realistic architecture of the BTS. This choice was made in order to understand and explain more easily the behavior of a BTS in different situation as well as to reduce the computational complexity of the simulations.

Secondly, many exogenous factors intervene in the model. In the project, we mainly focus on two of them : the **traffic** and the **channel condition**.

- The traffic represents the number of users connected to the BTS at time t, which of course varies over time. The traffic influences a variable called **load** (a value between 0 and 100) that defines the percentage of resource utilized by a user connected to the BTS at time t.
- To evaluate the quality of signal transmission to users, it is necessary to have information on the state of the transmission channel. We will denote it **CQI** (Channel Quality Index). The CQI (a value between 1 and 15) can vary depending on weather conditions (wind, rain...), the distance between the BTS and the user or the interference from other users. The higher the CQI value is, the better the channel conditions are. A bad CQI can deteriorate the quality of the signal emitted and lead to a bad quality of service for the user receiving data.

Finally, switching ON or OFF hardware components in the BTS takes finite time, introducing significant **time dynamics** to the system. That's why, if a BTS configuration transition is ongoing, it is necessary to wait for the transition to complete before switching ON/OFF new RUs, cells or antennas.

All the previous points show how challenging our problem is. We will see in the next parts how to formulate it mathematically and why **Reinforcement Learning** methods are useful to find a solution to this problem.

3 Problem formulation

As mentioned in the scenario, the two main factors to study in this project are the energy consumption (EC) and the quality of service (QoS). Hence the necessity to build good models for their estimation over time. We will now define these models and some auxiliary definitions that will be used in their implementation.

3.1 Load and Energy Consumption model

The power consumption of the BTS over time is evaluated according to the **load** associated with a configuration. The load refers to the percentage of resources used by a user connected to an active cell. It is related to the traffic, i.e. the number of connected users, without being proportional.

One of the main assumptions made in this project is that, at every time step t, there is a perfect **load balancing** over the active cells in the BTS. Since the load is defined for a given frequency carrier (cell), we will use the following formula :

$$\begin{cases} L_t^c = \min(L_0 \frac{n_{totalTX}}{n^c_{activeTX}}, 100) & \text{if } n^c_{activeTX} \neq 0 \\ L_t^c = 0 & \text{otherwise} \end{cases} \tag{1}$$

$n_{totalTX}$ denotes the total number of antennas in the BTS and $n^c_{activeTX}$ the number of active antennas in the cell c under study. L_0 is called **nominal load** and is used to set the ratio $\frac{n_{totalTX}}{n^c_{activeTX}}$ between 0 and 100. L_0 is a parameter that varies over time to modelize the fluctuations of the traffic.

As we can observe, the load of cell c is inversely proportional to the number $n^c_{activeTX}$ of active antennas in this cell. Indeed, the load in the whole BTS is 100% so increasing the number of active antennas in a cell will spread the load over the antennas, which will reduce the load value for this cell. On the other hand, if there is no active antennas in the cell then the load is 0.

To build a relevant energy consumption model, we exploited some existing data from Nokia. Studies in [5] have shown we can approximate the energy consumption of a cell by a linear function of its load. We have :

$$\begin{cases} EC_t^c = AL_t^c + B & \text{if cell } c \text{ is active} \\ EC_t^c = x & \text{otherwise} \end{cases} \tag{2}$$

The coefficients A, B and x are specific for an RU. x represents the cell power consumption value in Idle mode i.e. when all components are switched OFF. Thus, $x < B$ since the Idle configuration consumes much less power than an Active configuration.

The problem is that the empirical data provided by Nokia was not exhaustive for any RU (only exhaustive for RUs with 1 or 3 cells). Therefore, we chose

to linearly interpolate of the coefficients as a function of the number of cells in the RU.

We can illustrate the linear approximation of the power consumption as a function of the load, called here **Physical Resource Block (PRB) utilization**, on the following figure (in [2]):

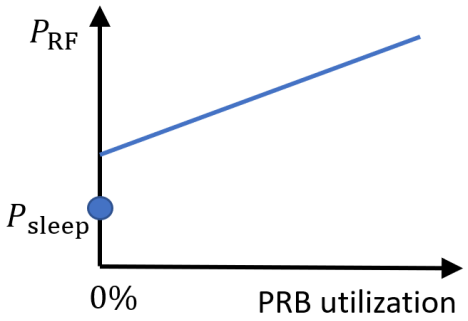


Figure 5: Linear approximation model of the power consumption of a cell

Here, P_{sleep} represents the x value in equation (2) (with $L_t^c = 0\%$).

From equation (2) we notice that, since the load for a given cell is lower when its number of active antennas is higher, cells with more active antennas will have a lower energy consumption than ones with fewer active antennas. Hence the necessity to adapt the coefficients A and B to take this into account. We chose to multiply these coefficients by the number of active antennas depending on the state of the cell.

We expressed the power consumption model for a given cell of the BTS. In order to get the power consumption of the BTS itself, we simply add the power consumption for each of its cells.

3.2 Budget and Quality of Service

As mentioned in the introduction, the Quality of Service is evaluated according to the respect of the KPI values. **We assign a KPI value for each cell** in the BTS. A variable OK_t allows us to check if at time t, enough of these KPIs exceed a certain limit y set by the customer.

The number of acceptable KPI values depend on the customer. In the project we will try two different formula (but many others could work and provide interesting results) :

$$\begin{cases} OK_t = \bigcap_{k=1}^{K} (KPI_k \geq y_k) \\ OK_t = \bigcup_{k=1}^{K} (KPI_k \geq y_k) \end{cases} \tag{3}$$

In the first function, we assume $OK_t = 1$ if and only if all the KPI values are beyond the limit y whereas in the second one, $OK_t = 1$ if at least one of the KPI values is greater than y.

In the problem, we are also interested to know how the Quality of Service behaved over the last few time steps. To take into account this dynamic, we define the **acceptable KPI frequency**, named $\overline{OK_t}$. $\overline{OK_t}$ is defined as the frequency the KPIs have been acceptable over the last T^{OK} time steps (it is a sliding window over time).

$$\overline{OK_t} = \frac{1}{T^{OK}} \sum_{i=0}^{T^{OK}-1} OK_{t-i} \tag{4}$$

We notice that $\overline{OK_t} = OK_t$ if $T^{OK} = 1$.

After that, we can define the minimum acceptable frequency of KPIs. We let the customer define the minimum acceptable value for $\overline{OK_t}$, denoted by x.

This notation allows us to introduce the **budget** b_t at time t defined as the difference between $\overline{OK_t}$ and its limit x. We have :

$$b_t = \overline{OK_t} - x \tag{5}$$

The budget will be considered in this project as a reference for the QoS : we will assume the Quality of Service is good enough if and only if $b_t \geq 0$.

4 MDP model

This is the first part of the internship dealing with the MDP model developed to solve the problem. In this part, we want to find out more about the strategies that give optimal results in several different situations.

Due to the numerous constraints mentioned in section 2 (consideration of time dynamics...) and the large dimensionality, the formulated problem is a complex optimization one. Therefore, some more efficient tools than classical optimization methods are needed to solve it.

A **Markov Decision Process** (MDP) is a mathematical framework for making sequential decisions under uncertainty i.e. where the system's responses have stochastic influences. MDPs are useful for studying optimization problems solved via dynamic programming and are used in many disciplines.

A MDP is defined by the 4-tuple (**states**, **actions**, **probability transitions**, **rewards**), as said in [6]. The states define a **fully observable** environment in which actions are taken to move from a state to another.

All the transitions between states are stochastic meaning that, in a MDP, a probability transition function is defined giving the probability associated with each action.

Also, for each transition, a reward is received depending on the quality of the action. Similarly, a reward function is defined, giving the reward received for each action taken. The reward function is central in the formulation of a MDP since it defines what we want to optimize. Indeed, the goal of the MDP is to maximize what we call the **long-term reward**. If we note r_t the reward obtained at time t, the long-term reward R can be defined two different ways as follows :

$$\begin{cases} R = \sum_{t=1}^{\infty} \gamma^t r_t & \text{(discounted long-term reward)} \\ R = \lim\limits_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} r_t & \text{(average long-term reward)} \end{cases} \tag{6}$$

$\gamma \in [0; 1]$ is called the **discount factor**. In this MDP, we chose to maximize the discounted long-term reward over time.

• **Value Iteration algorithm**

Solving an MDP helps us to converge towards a strategy called **policy**, giving for every state of the environment the action that will maximize the long-term reward. Several algorithms exist to converge towards this optimal policy. The one we will use in the tests is called **value iteration**.

The value iteration algorithm computes the optimal state value function V by iteratively updating the estimate. At each step, it looks ahead one step and goes over all possible actions to find the one that maximizes the value function.

Here is a description where, at every step, V_{new} is the new value function computed, R_t is the current reward vector (reward for every state), P_a is the transition probability matrix associated with action a and Q is simply a matrix to store the value function for every state and every action :

Value Iteration Algorithm

Initialization :
$V = V_0$, best_actions $= 0_{n_{actions}}$, $Q = 0_{n_{states}, n_{actions}}$, $\delta = 10$, $\epsilon = 0.01$
While $\delta \geq \epsilon$ **:**
    For a **in actions :**
        $Q[:, a] = R_t + \gamma P_a V$
    $V_{new} = \max_{actions} Q[:, 0 : n_{actions}]$
    best_actions $= \arg\max_{actions} Q[:, 0 : n_{actions}]$
    $\delta = abs(V - V_{new})$
    $V = V_{new}$
Return : V, best_actions

The algorithm is guaranteed to converge to the optimal state value function and the optimal policy, according to [7] and [6].

• **Bellman equation**

The value iteration algorithm is based on **Bellman's principle of optimality** (cf [8]). This principle states that an optimal policy has the property that whatever the initial state and action are, the remaining actions must constitute an optimal policy with regard to the state resulting from the first decision.

To demonstrate convergence towards a solution satisfying this principle of optimality, the supposedly optimal solution obtained needs to verify for every state s the following equation called **Bellman equation** :

$$V(s) = \max_a \sum_{s' \in S} R_t(s) + P(s|s', a)V(s') \tag{7}$$

• **Stationary distribution**

Steps after steps, an agent evolves in this known environment until it only visits the states for which it will receive maximum rewards. We can observe that by computing the **stationary distribution** that gives the probability or the proportion of time the agent spends in every state of the environment. If we note P^π the transition matrix associated with the optimal policy π, then the stationary vector S verifies :

$$S = P^\pi S$$

Therefore, a way to find the stationary distribution is to find the left eigenvector of the transition matrix P, associated with the eigenvalue 1.

For its efficiency in solving complex optimization problems with many constraints, we decided to build a MDP model for our problem.

4.1 MDP formulation

A few **assumptions** were made when formulating the MDP model to reduce its complexity. First, the traffic and CQI are known and supposed constant over time. Then, time dynamics are not taken into account, but a shortcut will be used and explained in this part to model a similar behavior. Finally we assume there is a perfect load balancing over the cells in the BTS (this assumption will not be lifted in this whole project).

The first thing to do when formulating the MDP is to define the states, actions, reward and transition probabilities between the different states.

4.1.1 States s_t

A state s_t will be defined, at time t, by the current hardware configuration n_t of the BTS and the number of times n_{OK_t} the variable OK_t is equal to 1 during the last T^{OK} iterations. In other words, $n_{OK_t} = \sum_{i=0}^{T^{OK}-1} OK_{t-i} = T^{OK}\overline{OK_t}$.

So we have : $s_t = (n_t, n_{OK_t})$

Note that the current configuration is actually an index associated with the configuration in question. For this, it is necessary to order and number the different BTS configurations. In the case we have only one RU containing 2 cells with 2 antennas in each we have the following order :

$$(0,0) = DS, (0,0) = idle, (1,0), (2,0), (0,1), ..., (2,2)$$

If we add to the BTS another RU with exactly the same structure as the first one, we get the following order for the configuration space :

$$(DS, DS), ..., ((2,2), DS), (DS, idle), ..., ((2,2), (2,2))$$

This standard notation of states will be used throughout the MDP problem to construct associated vectors and matrices.

4.1.2 Actions a_t

An action a_t, taken at time t, will be considered as the desired next configuration of the BTS (at time $t+1$, i.e., $a_{t+1} = n_{t+1}$). Thus, with this implementation choice, it is possible to switch from one BTS configuration to any other. Therefore, all transitions are possible.

4.1.3 Transition matrices P_{a_t}

As for the transition matrix P_{a_t} associated with a certain action a_t, two types of transition are to be analyzed : $n_{OK_t} \to n_{OK_{t+1}}$ et $n_t \to n_{t+1}$.

• We first study the transition $n_{OK_t} \to n_{OK_{t+1}}$. When moving from one sliding window to another, we have :

$$[OK_{t-T^{OK}}, ..., OK_t] \to [OK_{t-T^{OK}+1}, ..., OK_{t+1}]$$

Therefore, in this new window (at time $t+1$), we will either have one more 1, one less 1, or the same number of 1s as in the window at time t. Thus,

$$n_{OK_{t+1}} = \begin{cases} n_{OK_t} + 1 & \text{if } OK_{t-T^{OK}} = 0 \text{ and } OK_{t+1} = 1 \\ n_{OK_t} - 1 & \text{if } OK_{t-T^{OK}} = 1 \text{ and } OK_{t+1} = 0 \\ n_{OK_t} & \text{otherwise} \end{cases} \tag{8}$$

From the previous equation and the chosen numbering of configurations, we deduce that the transition matrix will be block tridiagonal as follows:

$$P_{a_t} = \begin{bmatrix} \Theta_0 & R_0 & 0 & \cdots & \cdots & 0 \\ L_1 & \Theta_1 & R_1 & \cdots & \cdots & \vdots \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & \cdots & \cdots & L_{T^{OK}-1} & \Theta_{T^{OK}-1} & R_{T^{OK}-1} \\ 0 & \cdots & \cdots & 0 & L_{T^{OK}} & \Theta_{T^{OK}} \end{bmatrix}$$

Each block R_i lists all the transitions $n_{OK_t} = i \rightarrow n_{OK_{t+1}} = i+1$. Each block L_i lists all the transitions $n_{OK_t} = i \rightarrow n_{OK_{t+1}} = i-1$. Each block Θ_i lists all the transitions $n_{OK_t} = i \rightarrow n_{OK_{t+1}} = i$. Every probability in the other blocks are equal to 0 by definition.

Let us define the probabilities associated with transitions of n_{OK_t}. According to equation (8), the probability that $n_{OK_{t+1}} = n_{OK_t} + 1$ is equal to the probability $Pr(OK_{t-T^{OK}} = 0)$ multiplied by $Pr(OK_{t+1} = 1|n_{t+1})$ (that we will explain later) since these two events are **independent**. Yet,

$$Pr(OK_{t-T^{OK}} = 0) = 1 - Pr(OK_{t-T^{OK}} = 1) = 1 - \frac{n_{OK_t}}{T^{OK}}$$

Similarly, the probability that $n_{OK_{t+1}} = n_{OK_t} - 1$ is equal to the probability $Pr(OK_{t-T^{OK}} = 1)$ multiplied by $Pr(OK_{t+1} = 0|n_{t+1}) = 1 - Pr(OK_{t+1} = 1|n_{t+1})$.
Finally, we deduce the probability of $n_{OK_{t+1}} = n_{OK_t}$:

$$Pr_{n_{OK_{t+1}}=n_{OK_t}} = 1 - (Pr_{n_{OK_{t+1}}=n_{OK_t}+1} + Pr_{n_{OK_{t+1}}=n_{OK_t}-1})$$

We now have all the probability for the transitions $n_{OK_t} \rightarrow n_{OK_{t+1}}$:

$$\begin{cases} Pr_{n_{OK_{t+1}}=n_{OK_t}+1} = (1 - \frac{n_{OK_t}}{T^{OK}}) p(n_{t+1}) \\ Pr_{n_{OK_{t+1}}=n_{OK_t}-1} = \frac{n_{OK_t}}{T^{OK}} (1 - p(n_{t+1})) \\ Pr_{n_{OK_{t+1}}=n_{OK_t}} = \frac{n_{OK_t}}{T^{OK}} p(n_{t+1}) + (1 - \frac{n_{OK_t}}{T^{OK}})(1 - p(n_{t+1})) \end{cases} \tag{9}$$

$p(n_{t+1}) = Pr(OK_{t+1} = 1|n_{t+1})$ and it corresponds to the **probability that enough KPIs are acceptable over all active cells** (second formula in (6)).

This probability is equal to the product of probabilities that each active cell verifies the KPI condition. The probability function that an active cell verifies the KPI condition was not implemented in this project but in a previous one by Lorenzo Maggi. It only takes as parameters the load of the cell as well as the CQI associated and return a probability. We used this function in the entire project.

• Now, let's study the transitions $n_t \to n_{t+1}$. Each block on the diagonal, and upper and lower diagonals of the transition matrix associated with an action a has the following shape:

$$\begin{bmatrix} (1-\lambda_1^a)P_{\#_j} & 0 & \lambda_1^a P_{\#_j} & 0 & \cdots & 0 \\ 0 & (1-\lambda_2^a)P_{\#_j} & \lambda_2^a P_{\#_j} & 0 & \cdots & \vdots \\ 0 & 0 & P_{\#_j} & 0 & \cdots & \vdots \\ \vdots & & \vdots & \ddots & & 0 \\ \vdots & \cdots & \lambda_{n-1}^a P_{\#_j} & \cdots & (1-\lambda_{n-1}^a)P_{\#_j} & 0 \\ 0 & \cdots & \lambda_n^a P_{\#_j} & 0 & \cdots & (1-\lambda_n^a)P_{\#_j} \end{bmatrix}$$

In each block of such matrix, there are obviously non-zero probabilities on the column corresponding to the index of the next configuration targeted by the action a. Yet, we also notice the presence of non-zero probabilities on the diagonal. This is the trick found to deal with time dynamics in our MDP model.

Time dynamics : According to the previous matrix, the probabilities on the diagonal means we assume there is also a probability that the system remains in its current configuration when taking an action a. The choice between the current configuration or the one targeted by the action is regulated by a set of parameters $\lambda = (\lambda_i)$. If we note (like in the previous matrix) $P_{\#_j} = Pr(OK_{t+1} = 1 | n_{t+1})$, then for a given λ_i the BTS will take the new configuration with probability $\lambda_i P_{\#_j}$ and will remain in its current configuration with probability $(1-\lambda_i)P_{\#_j}$.

λ is a shortcut to take into account the transition delay from one BTS configuration to another (antenna, cell or RU activation/deactivation time). λ depends on the current configuration and the next action. Indeed, if we are in a "Deep Sleep" state (no active RU), and we consider the action where we turn on all the antennas or the one where we turn on only one antenna, the transition time will be different. Thus, $\lambda = (\lambda_{i,j})_{i,j \in [1;n]}$ with n the number of possible configurations of the BTS.

More precisely, we define 7 different values of λ corresponding to turning OFF/ON an RU ($\lambda[0]/\lambda[1]$), a cell ($\lambda[2]/\lambda[3]$) or an antenna ($\lambda[4]/\lambda[5]$) or staying in the current configuration ($\lambda[6]$) (the latter is useless since it does not intervene in the transition matrix - see above).

We can then define a matrix for the $\lambda = (\lambda_{i,j})_{i,j \in [1;n]}$ parameters for a certain RU :

$$\begin{bmatrix} \lambda[6] & \lambda[1] & \lambda[1] & \cdots & \cdots & \cdots & \lambda[1] \\ \lambda[0] & \lambda[6] & \lambda[3] & \cdots & \cdots & \cdots & \lambda[3] \\ \lambda[0] & \lambda[2] & \lambda[6] & \lambda[5] & \min(\lambda[4],\lambda[5]) & \cdots & \min(\lambda[4],\lambda[5]) \\ \vdots & \vdots & \lambda[4] & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \min(\lambda[4],\lambda[5]) & \cdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \cdots & \ddots & \vdots \\ \lambda[0] & \lambda[2] & \min(\lambda[4],\lambda[5]) & \cdots & \cdots & \cdots & \lambda[6] \end{bmatrix}$$

Some actions can lead to a change of configurations in different RUs meaning it has to appear in the matrix of λ. With the chosen numbering of configurations, we can conclude that the λ matrices for each RU are then nested within each other. For example with 2 RUs, we get the following block matrix :

$$\begin{bmatrix} \min(\lambda_{mat}^{RU1},\lambda[6]) & \min(\lambda_{mat}^{RU1},\lambda[1]) & \cdots & \min(\lambda_{mat}^{RU1},\lambda[1]) \\ \min(\lambda_{mat}^{RU1},\lambda[0]) & \min(\lambda_{mat}^{RU1},\lambda[6]) & \cdots & \min(\lambda_{mat}^{RU1},\lambda[3]) \\ \vdots & \vdots & \ddots & \vdots \\ \min(\lambda_{mat}^{RU1},\lambda[0]) & \min(\lambda_{mat}^{RU1},\lambda[2]) & \cdots & \min(\lambda_{mat}^{RU1},\lambda[6]) \end{bmatrix}$$

4.1.4 Cost C_t

The reward associated with the studied MDP will be considered as a cost that must be minimized in the long run.

According to the BTS configuration, the budget can be negative, which should not happen since the QoS would be under its acceptable limit by definition. It is therefore necessary to define a penalty function to avoid choosing a configuration that could lead to a negative budget. We denote it $l_{KPI}(.)$ and define it as follows :

$$\begin{cases} l_{KPI}(b_t) = 0 & \text{if } b_t \geq 0 \\ l_{KPI}(b_t) > 0 & \text{otherwise} \end{cases} \tag{10}$$

Here is an example of KPI penalty on the following figure :

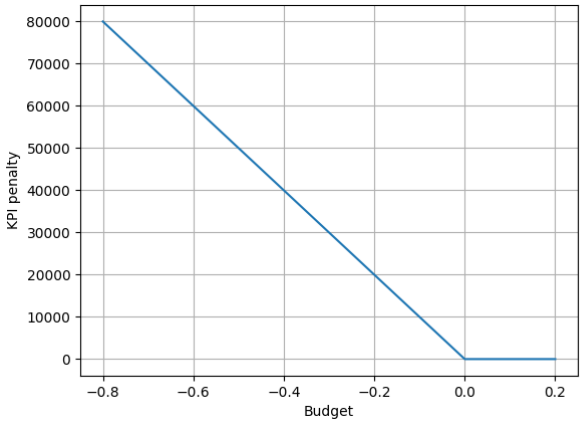


Figure 6: Example of KPI penalty function

In this example, the penalty is a linear function of the budget :

$$l_{KPI}(b_t) = KPI_{PENcoeff} \times b_t$$

where $KPI_{PENcoeff}$ controls how much we penalize configurations leading to negative budgets: the higher this parameter is, the higher will the penalty be. We will keep this choice of implementation for the penalty in the next parts of the project.

The cost at time t that we want to minimize will be seen as the sum of the energy consumption of the current BTS configuration n_t and the KPI penalty associated with the current budget b_t :

$$C_t = EC_t(n_t) + l_{KPI}(b_t) \tag{11}$$

The goal of the problem is therefore, as described in equation 12, to minimize the energy consumption $EC_t(n_t)$ of the BTS as well as the penalty $l_{KPI}(b_t)$ related to the budget :

$$\min_{n_t} \lim_{W \to +\infty} \frac{1}{W} \sum_{t=1}^{W} \mathbb{E}(EC_t(n_t) + l_{KPI}(b_t)) \tag{12}$$

An idea was to integrate to the cost of the MDP a penalty on configuration changes. Since switching ON or OFF hardware components takes time and power to materialize, we could have added a penalty so that we urge the BTS to remain in its current configuration as long as the budget is acceptable. However, this could have complexified the expression of the cost but also have forced the BTS to remain in configurations that consume too much energy, which could not have guaranteed a good trade-off. For these reasons, we decided not to introduce this penalty on configuration changes.

4.2 Results

Tests have been run on a simple BTS architecture in which we assume having 2 RUs with 1 cell in each. Each of these cells contain a set of 2 antennas that can be switched ON/OFF over time. The **default parameters** for the tests are defined as follows :

$T_{OK} = 30$, $KPI_{PENcoeff} = 10^5$, $L_0 = 50$, $x = 0.8$ (budget limit $n_{OK} \geq 24$), $CQI_1 = CQI_2 = 12$, $\gamma = 0.96$, $\lambda_{val1} = \lambda_{val2} = [0.001, 0.001, 0.25, 0.25, 0.5, 0.5, 1]$.

4.2.1 Convergence and Bellman equation

According to the introduction of section 4, the value iteration algorithm is guaranteed to converge towards the optimal values.

When running the algorithm with the default parameters defined above, we can display the value function for a given state or the mean value of the value function V. On the following plot, we notice a convergence of the value function over time for the state (idle,idle) :

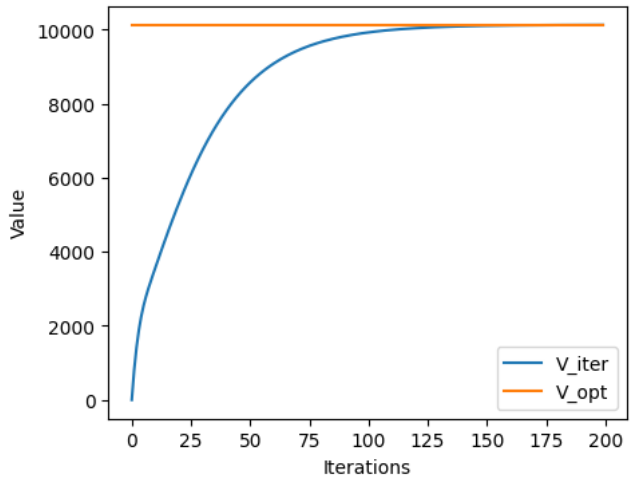


Figure 7: Value Iteration convergence for the state (idle,idle)

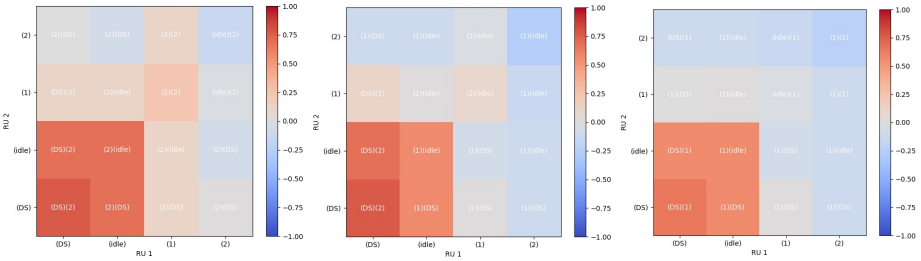
After verifications, we can confirm that the solution obtained also verified the Bellman equation, **for every state**. As a consequence, our MDP model gives us an optimal strategy for minimizing the long-term cost, according to Bellman's principle of optimality.

4.2.2 MDP policy

In this part, we will analyze the MDP policy for different parameters. The idea is to find out how the agent behaves in different situations and if its actions make sense in these new contexts.

- **Standard parameters**

It is now possible to display, for a fixed value of n_{OK}, a heat map representing the actions to be taken in each configuration of the BTS. This can be observed in the following figures:



(a) $n_{OK} = 0$ (b) $n_{OK} = 26$ (c) $n_{OK} = 30$

Figure 8: Heat map of actions for different n_{OK} values (with the default parameters) showing the difference of energy consumption between the current BTS configuration and the targeted one

The x-axis and y-axis lists the possible configurations for each of the two RUs. Thus, all the possible configurations are represented in these heat maps (16 different configurations so 16 boxes). In each box, we can see the action the agent wants to take for each initial configuration. The color in each box represents the variation in energy consumption between the initial configuration and the targeted one. This will help us find out if more or less hardware components are active in the next wanted BTS configuration.

On the first heat map, $n_{OK} = 0$ which corresponds to a bad QoS. We notice that the agent is willing to take actions that switch on all the antennas in at least one of the two RUs in order to improve the budget and QoS (especially in 'idle' and 'DS' configurations). These actions result in an increase of the energy consumption by the BTS which explains the red color for the heat map.

On the second heat map, $n_{OK} = 26 \geq 24$ i.e. the budget is over its acceptable limit. When having a look at the actions in the heat map, we notice the agent starts targeting configurations with only 1 active antenna in active cells meaning it tries to progressively reduce the energy consumption of the BTS. That's why the heat map is getting bluer.

On the last heat map, $n_{OK} = 30 >> 24$ so the budget is maximum. Here,

the QoS is even too good and there is a need by the agent to significantly decrease the BTS energy consumption. Here is why more blue boxes appear on the heat map since the agent takes more actions that lead to configurations with less active components. However, since we have a high KPI penalty, the 'idle' and 'DS' configurations are avoided by the agent which explains why these boxes remain red in the figure.

On the previous heat maps we get information on states that may never be visited by the system. The **stationary vector** gives us information on what states are visited during the run of the algorithm. We can have a look at the stationary vector associated with the previous tests :

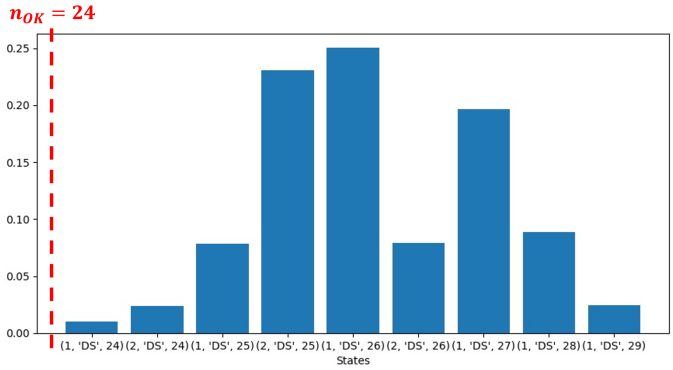


Figure 9: Bar chart of the maximum values in the stationary vector (with the default parameters)

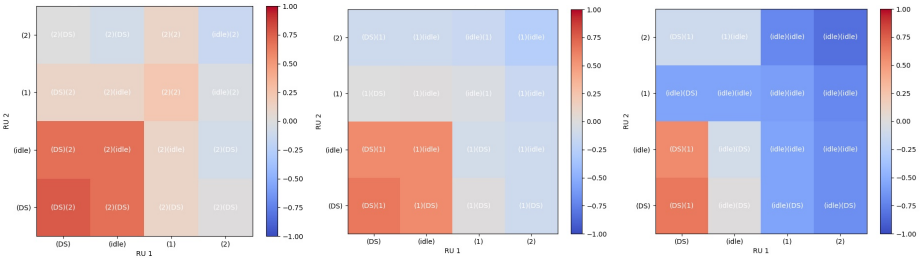
On the figure, we only plot the states in the stationary vector with the highest values. The states on the x-axis are defined as a tuple that contains the configurations of RU1 and RU2 as well as the value of n_{OK}.

We notice that among the most visited states, $n_{OK} \geq 24$ which satisfies the condition on the budget and the QoS.

- **Lower KPI penalty**

Let's now observe the effect of reducing the KPI penalty on the MDP policy. Let's choose $KPI_{PENcoeff} = 10^4$ and let's plot the same heat maps as previously to observe the actions the agent is willing to take in this context :

We observe the plot is different for $n_{OK} = 26$ and $n_{OK} = 30$. Once the budget has reached its limit, the agent is even more likely to switch OFF antennas in the BTS and target configurations with at most one active antenna in each RU. As for $n_{OK} = 30$, the heat map is almost completely blue and the agent now targets 'idle' and 'DS' configurations. The situation almost seems critical since it wants to shut down every hardware component in the BTS. The



(a) $n_{OK} = 0$ (b) $n_{OK} = 26$ (c) $n_{OK} = 30$

Figure 10: Heat map of actions for different n_{OK} values (with a lower KPI penalty) showing the difference of energy consumption between the current BTS configuration and the targeted one

reason is since we are less strict on the respect of the KPI, the weight of the energy consumption in the cost of the MDP problem is higher. This is the reason why the agent focuses more on the consumption of the BTS than the KPI values.

We can now observe the visited states by plotting the stationary vector:

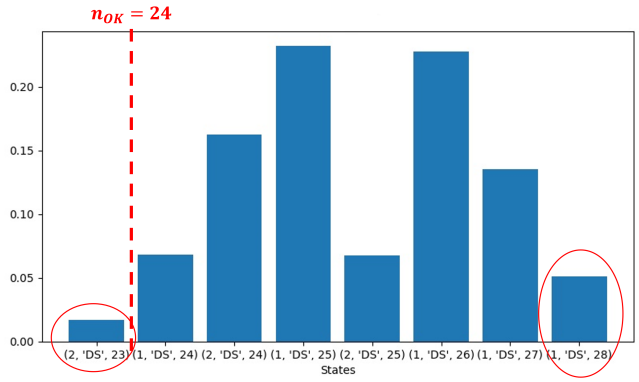


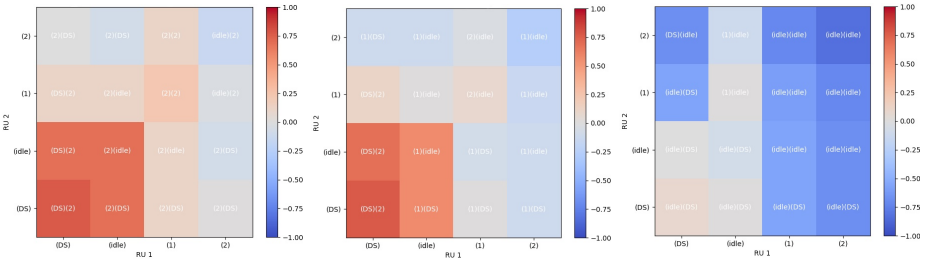
Figure 11: Bar chart of the maximum values in the stationary vector (with a lower KPI penalty)

We notice that the visited states are such that : $23 \leq n_{OK} \leq 28$. Which means that some states that do not guarantee a good enough QoS since they correspond to a budget lower than its acceptable limit. The aforementioned idea that the agent focuses less on KPI acceptance is highlighted here with a "shift" towards the left of the previous stationary vector histogram. Indeed, since having a bad KPI is less penlized, the system will visit states for which : $b_t \leq x$.

• **Wider sliding window**

Let's change another parameter : $T_{OK} = 100$. T_{OK} is the length of the sliding window that contain the values for n_{OK} over the last time steps. Let's increase the value of T_{OK} so that the agent will take more into account its experience (historic data) and act consequently.

With this modification, since $x = 0.8$, the budget limit now corresponds to $n_{OK} \geq 80$. Let's plot the heat map of actions and compare it to the first test :



(a) $n_{OK} = 0$ (b) $n_{OK} = 82$ (c) $n_{OK} = 100$

Figure 12: Heat map of actions for different n_{OK} values (with a wider sliding window) showing the difference of energy consumption between the current BTS configuration and the targeted one

The two first heat maps are similar to the one obtained in the first test. The behavior is different for a maximum budget ($n_{OK} = 100$). Indeed, now the phenomenon observed in the previous plot is even more visible : the QoS is far too good and now there is an urgent need for the agent to shut down every hardware component in the BTS to reduce significantly its energy consumption. Here is why the heat map is now completely blue.

Finally, let's observe the distribution of the stationary vector :

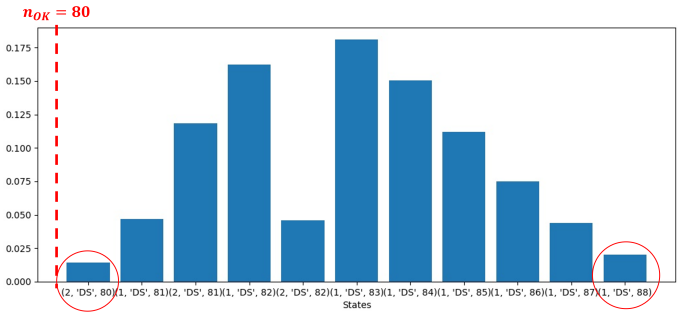


Figure 13: Bar chart of the maximum values in the stationary vector (with a wider sliding window)

The bar chart makes it clear that the visited states are such that : $80 \leq n_{OK} \leq 88$. Thus, the agent targets states that satisfies just enough the QoS for the user connected, which confirms the observation in the agent's heat maps of actions.

4.2.3 Policy comparison

In the previous part, we have shown results provided by the MDP policy. The strategy the agent found seems to make sense in different context. However, we still need to evaluate the performance of this policy by comparing it to other policies.

Let's consider a **random** policy that picks up any hardware configuration at random, at every time step. Let's also choose a simple **greedy** policy that makes sense for anyone: we switch ON every component in the BTS as long as the QoS has not reached its acceptable limit, and once we are over this limit, we switch OFF everything and go to 'Deep Sleep' mode for every RU.

As a reminder, the goal of the project is to find a strategy (policy) that minimizes the **long-term cost**. Let's then compare the long-term cost of these two policies with the one obtained with the strategy the agent learns in the MDP (with the default parameters). Since the cost is the sum of the BTS energy consumption and the KPI penalty, we can also plot the **long-term energy consumption** and the **long-term KPI penalty** which will give us more information on the trade-off. We have the following benchmark :

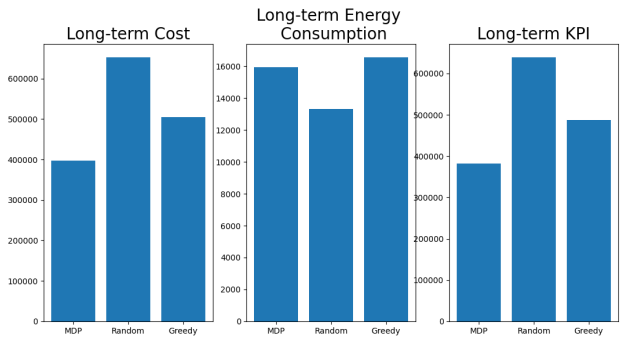


Figure 14: Policy comparison (default parameters)

The output show that the MDP policy provides the better solution for the trade-off since it corresponds to the smallest long-term cost. **We observe on figure** 14 **that the long-term cost obtained with the MDP is respectively 33**% **and 20**% **lower than the ones obtained with the random and greedy policies**. After that, the greedy strategy performs better than the random one overall. We notice though, that the greedy strategy consumes more energy than random (due to the extreme actions that are taken) but provides

a way better KPI since the actions are defined according to the value of the budget instead of being picked up at random.

Nevertheless, the performances of each policy can vary depending on the choice of the parameters, especially $KPI_{PENcoeff}$, regulating how much weight we put on the respect of QoS in the trade-off. If we decrease significantly its value ($KPI_{PENcoeff} = 1000$), we get a different plot :

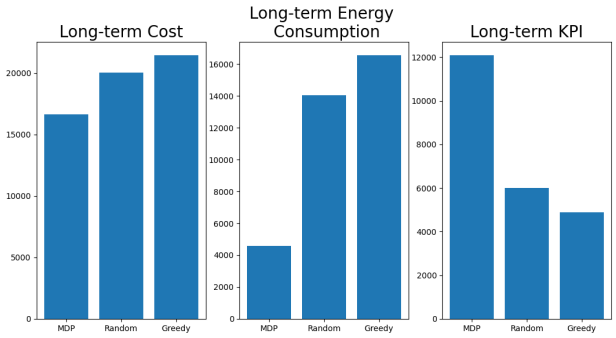


Figure 15: Policy comparison ($KPI_{PENcoeff} = 1000$)

Here again, the MDP policy is still better than the two others. **Figure** 15 **shows that the long-term cost obtained with the MDP is now around 20% and 27% lower than the ones obtained with the random and greedy policies**. However, we observe that the random strategy now provides a smaller long-term cost than the greedy one. We see that, since the energy consumption of the BTS matters more, the MDP provides the solution that minimizes the most the long-term EC, which also explains why the greedy policy performs so bad overall. As well, the long-term KPI has very few effect on the cost : the MDP policy is the worse strategy to minimize this quantity but still remains the best strategy for minimizing the long-term cost.

Consequently, since the KPI penalty is less important, the strategies able to minimize the most the BTS energy consumption no matter the QoS, are the best strategies for the problem.

4.3 Summary

If we summarize all the results of the MDP, we realize that $KPI_{PENcoeff}$ plays a key role in the trade off between EC and QoS. For a given BTS architecture, the energy consumption model does not change over time, meaning that finding the correct KPI penalty function is central in the formulation of the trade-off problem. A too high KPI penalty could lead to a solution with an excessive energy consumption whereas a too small KPI penalty would give a strategy that does not provide a good enough QoS to the users.

Yet, the MDP was applied to a simple model of BTS. The model in real life is more complex especially because the assumptions made to make the MDP work are lifted. The work done on the MDP shows us that Reinforcement Learning can provide us a good energy saving solution in theory, but another model than MDP should be used when dealing with the real life problem in the next part of the report.

5 RL model

The second part of the internship starts here. It deals with all the study and work related to the Reinforcement Learning model which can be seen as an extension of the MDP model previously formulated.

This part aims at assessing the performance of Reinforcement Learning algorithms on a more complex BTS simulator. The idea is to compare the performances and policies obtained thanks to the simulator with those derived from the MDP in the previous part of the project for given algorithms.

5.1 RL basis

In **Reinforcement Learning** (RL), an agent interacts with a dynamic and interactive **unknown environment**. At each step, the agent observes that the environment is in a given **state**. The agent then takes an **action** and receives a feedback called **reward** on the quality of the action. Its goal is to learn a strategy (namely **policy**) that maximizes the long-term sum of rewards. Reinforcement Learning is a branch of Machine Learning.

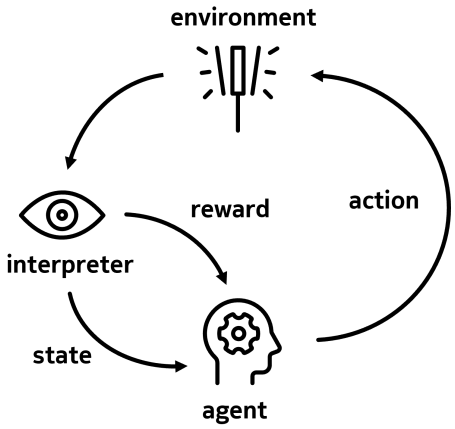


Figure 16: Reinforcement Learning principle

In RL, the problem to resolve is described as a MDP in which probabilities and rewards are unknown. If the problem is well described as a MDP, then RL may be a good framework to use to find solutions.

The **difference between MDP and RL** is that in RL, the agent interacts dynamically with the environment to learn how to optimally behave in it whereas the MDP is a formal representation of this environment which is useful to find what optimal policy to expect in theory.

There are 2 phases when the agent learns : **exploration** (to get information on the environment) and **exploitation** (to optimize the rewards received).

5.2 Proximal Policy Optimization (PPO)

Plenty of RL algorithms have emerged over the past 10 years until now, the new ones always trying to improve the state-of-the-art ones in terms of performance, ease of use and rapidity of convergence.

In RL, we can define two classes of learning algorithms : **model-free** and **model-based**. Model-based approaches use a predictive model of the environment (MDP for instance) to make decisions. On the contrary, model-free ones do not use transition probability distribution or reward functions associated with the MDP since they are part of the problem to be solved. They rely on **trial-and-error** methods to learn the consequences of the actions taken.

In our problem, we indeed need to use model-free algorithms. A few research has lead us to select a family of algorithms called Proximal Policy Optimization (explained in [9]).

Proximal Policy Optimization (PPO) is a powerful class of algorithms for training Reinforcement Learning agents. PPO algorithms are **policy gradient methods** meaning that instead of trying values for states and actions, they directly search in the space of policy.

PPO is the new generation of an older class of algorithms called **Trust Region Policy Optimization** (TRPO), which is based on the same principle of policy search and convergence.

We have chosen to apply PPO to test our BTS simulator due to its better performance, ease of use and its lower computation complexity.

Finally, PPO is an **on-policy** algorithm, which means that it learns directly from the actions taken in the current policy, rather than from a separate set of data. It updates iteratively the policy using data collected from interactions with the environment which ensures its stability and efficiency.

5.3 Implementation details

5.3.1 BTS simulator

In the continuation of this project, the goal is first to develop a BTS simulator that is able to imitate as close as possible the behavior of a telecommunication station in real life. Thus, we will get closer to the real problem that we want to solve.

During the implementation of the MDP, we made assumptions about the model to facilitate its formulation. When building the simulator, we decided to lift some assumptions. Now we assume that:

The traffic and CQI now **vary over time**. The CQI can take random values between 10 and 15 since it corresponds to a fairly good channel quality.
In practice, the load takes sequentially and for a significant period of time high and low values since this is how the traffic usually evolves. According to the expression of the load given in equation (1) (section 3), the parameter that varies when the traffic does is the nominal load L_0. L_0 defines the range of values for the traffic.
There are various ways to model this behavior for L_0 and the CQI values like **normal laws** centered on a relevant value, **Moving Average** or simply according to a **sinusoidal function**.
The use of RL makes sense here since we are not supposed to have information on the environment over time (traffic, CQI).

Time dynamics are taken into account now. Every transition from a configuration to another takes a certain number of time steps and we assume no other actions can be taken while an action is ongoing.

Finally, in the simulator, all configuration transitions are not possible anymore. Indeed if we want to turn a RU into Deep Sleep mode, it is necessary to go first in Idle mode and the other way around. All the other transitions are enabled as in the MDP. Here is the graph of configuration transitions for an RU with 2 cells and 2 antennas in each :

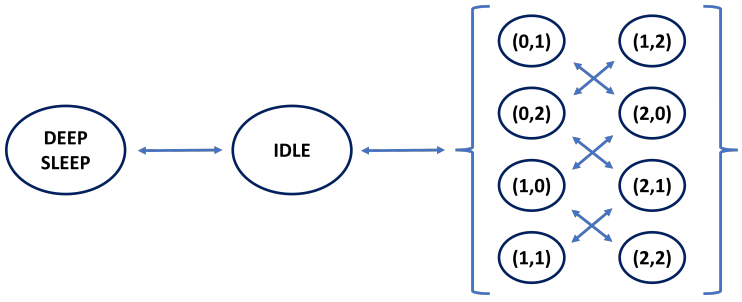


Figure 17: Example of graph of configuration transitions for the simulator

5.3.2 Gym environment

Once the simulator is coded, the objective is to build an RL environment using the **Python Gym library** in order to establish an interface for the simulator on which different RL algorithms can be applied. This second phase will require the use of other libraries such as RLlib or Tianshou able to give access to many RL algorithms (including PPO).

Gym is a standard Application Programming Interface (API) for Reinforcement Learning. It contains a large number of environments on which agents can be trained.

When setting up a Gym environment it is important to inform the action_space, observation_space, how to reset the environment (`reset()`) and a `step()` function that updates the environment after the agent takes an action. We can also specify methods such as `render()` to give an output of the state of the environment and `close()` to close the environment.

It is also possible to vectorize the environment. **Vectorized environments** are environments that "run multiple independent copies of the same environment in parallel using multiprocessing" (see Gym documentation in [10]). It takes as input a batch of actions and returns a batch of observations. This is particularly useful when the policy is defined as a neural network that operates over a batch of operations. That's what we will do to train the agent.

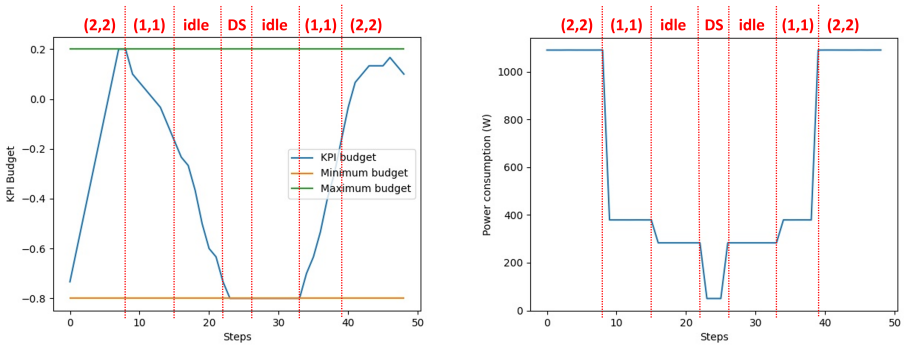
5.4 Observations and ongoing developments

In this part, we will show the first observations about the BTS simulator coupled with the Gym interface. As said in the Introduction, most of the simulations with the PPO algorithm haven't been made yet or are ongoing. Therefore, due to the advancement of the project, we won't plot any results about the training of the agent but we will of course describe what we are expecting to obtain as well as the next steps.

5.4.1 Observations

Before training the agent, it is important to make sure the simulator has been coded correctly. A simple test is to take a series of actions to change the BTS configuration and have a look at how the budget and the energy consumption evolve.

Let's run a test on a BTS composed with only 1 RU containing 2 cells. At first, every antennas are ON and we decide to switch OFF one of the two antennas in each cell. Then we go to Deep Sleep mode with the necessity to go to Idle in between (see section 5.3.1). We repeat those steps the other way around and observe the evolution of the budget over time. We keep the **default parameters** used in the tests for the MDP except for the load and CQI which vary over time now. They are such that : $0 \leq L_0 \leq 20$ and $CQI_1, CQI_2 \in [10; 15]$. We get the following plot :



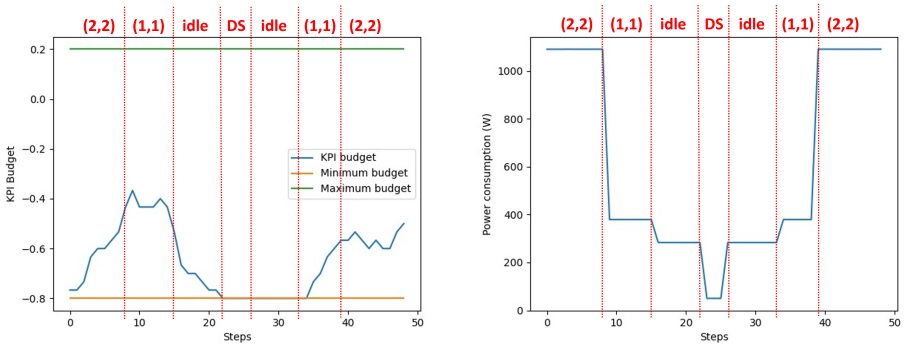
(a) KPI budget evolution

(b) Power consumption evolution

Figure 18: KPI budget and power consumption evolution for $0 \leq L_0 \leq 20$

The outputs make it clear that the budget and power consumption decrease over a few time steps when switching OFF hardware components in the RU and increase when switching them ON. Both figures are coherent with each other : the energy consumption and the QoS follow the same evolution over time.

NB : The choice of the range of values for the nominal load are important since if we have a too small BTS architecture and we set the load values too high, the budget will never manage to reach positive values. Let's use the same BTS architecture as previously but now let's set the nominal load such that : $40 \leq L_0 \leq 60$. Let's observe the result on the following plot for which we take the same series of actions as in the previous simulation :



(a) KPI budget evolution

(b) Power consumption evolution

Figure 19: KPI budget and power consumption evolution for $40 \leq L_0 \leq 60$

We clearly observes that, contrary to the previous plot, the budget cannot take higher values than -0.3, as if it reached a maximum threshold. This means that whatever action the agent takes, the QoS will never be satisfied, which will push the agent to care only about power consumption and then take wrong actions.

For the tests, we can either build a large BTS architecture (which is not optimal since running RL algorithms will take a lot of time due to the large dimensionality explained in section 2) or only consider low traffic in order to make sure the agent is in an environment under normal conditions.

5.4.2 Expected results

In this part, we will describe some simulations that can be made on the agent and the results we expect to get. As mentioned before, we will try out simple BTS architectures to get an idea of what we can expect in terms of optimal policy and to easily check if the agent can learn correctly.

Like in the observations, we chose to give a reasonable range of values for the nominal load : $0 \leq L_0 \leq 20$. With this choice, we will build two different BTS architecture for the tests :
First architecture : 1 RU with 1 cell containing 2 antennas.
Second architecture : 1 RU with 2 cell containing 2 antennas each.

Observations on the first architecture show that the budget can get positive values only when all the antennas are ON in the BTS (mode (2) on figure 20). Therefore, in order to realize the best trade-off possible between EC and QoS, the agent should learn to always take the action of switching ON everything in the BTS.

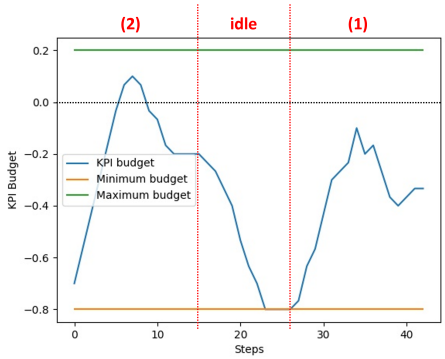


Figure 20: KPI budget evolution in the first architecture (1 RU with 1 cell)

Observations on the second architecture are more interesting. They show that in the configurations in which one cell is fully active (both antennas are ON) and one cell is partially active (only one antenna is ON), the budget oscillates between positive and negative values (see modes (1,2) and (2,1) in figure 21). The only configuration in which the budget is always positive is the one for which everything is ON (mode (2,2)). Yet, in this latter configuration, the budget can take values that go way beyond the acceptable threshold which is not optimal for the trade-off problem.

Thus, in this case, the agent should learn to take actions in order to switch between these 3 configurations so that the budget always get positive but close-to-0 values.

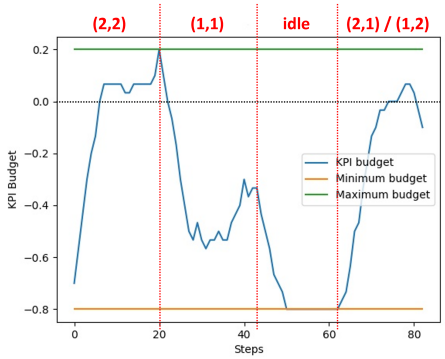


Figure 21: KPI budget evolution in the second architecture (1 RU with 2 cells)

Next tests :
If the agent learns on the previous simple simulations, then we can run some more complicated tests like with much bigger architectures and higher traffic, which will require GPUs for computation. The idea would be to generalize the previous observations to BTS with several RUs and different number of frequency carriers. For example :
Third architecture : 1 RU with 3 cells, 1 RU with 2 cells. Each of the cells containing 2 antennas.

5.4.3 Difficulties

When testing the RL agent, it is complicated to deal with time dynamics. Indeed, according to the choice of implementation, we assume that no other actions can occur while a transition is ongoing. However the agent takes a new action at every time step so it should be aware that the actions taken in the meantime will not be taken into consideration. This is a tricky point that could completely ruin the learning of the agent since this one could avoid taking interesting actions simply because it did not receive any reward (cost here) for them.

Another issue encountered when training the agent is the fact that it is supposedly impossible to go from an active configuration to Deep Sleep. According to simulator implementation, taking such action will result in receiving an error message. However, when training the agent, these messages are not taken into account and the agent receives a penalty for impossible configuration actions which will build a wrong policy.

The plots in the MDP results section showed that the cost (both KPI penalty and energy consumption) can take values up to $O(10^5)$. For some reasons, training the agent is hard when trying configuration actions resulting in a big variation of energy consumption.

To improve the learning of the agent in extreme situations, a choice was made to normalize both the energy consumption and KPI penalty. We now combine the two using a weight factor $a \in [0, 1]$ such that :

$$C_t = a \times EC_t(n_t) + (1 - a) \times l_{KPI}(b_t) \tag{13}$$

So if we set $a = 1$ then all the agent will care about is energy saving, conversely, if a=0 then all it will care about is KPI.

The biggest thing is that the rewards are now close to unity, which seems to greatly improve the training stability. For example, the agent can quickly learn the extreme cases (either every components ON or in Deep Sleep) in a one-RU-one-cell architecture.

6 Conclusion

As a conclusion, this project aims at showing how AI, especially Reinforcement Learning is a very useful tool to solve energy saving problems in the telecommunication industry. We came up with a solution that realizes a better trade-off between Energy Consumption and Quality of Service than many empirical solutions. We worked on simple BTS architectures but our solution should be adaptable to any.

The Markov Decision Process (MDP) helped us find out the optimal strategies to adopt when traffic and channel conditions are constant and known over time.

It is also important to know the MDP provides a policy that is optimal according to the problem formulated. Indeed, this is a question of balance between EC and QoS : depending on the weight granted to EC for instance, the optimal strategy learnt by the agent will be more or less power-greedy.

After that, we started to tackle the problem without knowing the evolution of traffic and channel conditions over time. Reinforcement Learning (RL) model-free algorithms like PPO are expecting to be very efficient for training the agent to dynamically interact with its environment (the live network) under those conditions.

Nevertheless, some developments are still ongoing especially how to train the agent on more complex architecture of the BTS, using PPO algorithm.

The next steps of the project will be, in the short, term to make the RL model work and train the agent correctly to solve the problem. As said in the Introduction, another goal is to patent this project by the end of the year, to conclude these 6 months of research.

In the long run, the mission for Nokia will be to integrate this energy saving feature on real networks and make sure this integration is efficient and provides similar performances to the ones observed in the model.

Other energy saving features will emerge and develop in next-generations live networks. The challenge for the future will be to always develop technologies able to both keep up with the growing number of users and to provide a sustainable solution that conciliates both the digital performance and the preservation of our planet...

Nomenclature

Acronyms

BTS : Base Transceiver Station
RU : Radio Unit
CQI : Channel Quality Indicator
RRM : Radio Resource Management
MIMO : Multiple Input and Multiple Output
PRB : Physical Resource Block
RL : Reinforcement Learning
QoS : Quality of Service
EC : Energy Consumption
MDP : Markov Decision Process
KPI : Key Performance Indicator
DS : Deep Sleep
TRPO : Trust Region Policy Optimization
PPO : Proximal Policy Optimization

Symbols

L_t^c : load of the cell c at time t
L_0 : nominal load
$n_{activeTX}^c$: number of active antennas in the cell c
EC_t^c : energy consumption of the cell c at time t
T^{OK} : length of the sliding window
OK_t : variable defined at time t that equals 1 / 0 if enough / not enough KPI values are acceptable
b_t : budget at time t
x : $\overline{OK_t}$ acceptable limit
s_t : state of the agent at time t in the MDP
n_t : configuration of the BTS at time t
n_{OK_t} : number of times $OK_t = 1$ in the sliding window
a_t : action the agent takes at time t
P_{a_t} : transition matrix associated with the action a_t
$p(n_{t+1}) = Pr(OK_{t+1} = 1|n_{t+1})$: probability that enough KPI values are acceptable on all active cells
C_t : cost the agent gets at time t in the MDP
$l_{KPI}(b_t)$: KPI penalty function
$KPI_{PEN_{coeff}}$: director coefficient of $l_{KPI}(b_t)$
$EC_t(n_t)$: energy consumption of a given BTS configuration n_t
γ : discount factor in the MDP
V : value function in the MDP

References

[1] Grace Donnelly. Green mobile networks: What are they and how can telcos make the transition ?

[2] Lorenzo Maggi, Claudiu Mihailescu, Qike Cao, Simo Aaltonen, Ryo Koblitz, Maunu Holma, Samuele Macchi, Maria Ruggieri, Igor Korenev, and Bjarne Klausen. Energy savings under performance constraints via carrier shutdown with bayesian learning, 02 2023.

[3] Lorenzo Maggi, Alvaro Valcarce Rial, Claudiu Mihailescu, Maunu Elias Holma, and Afef Feki. Optimizing usage of power using switch off of cells, 02 2023.

[4] Chris Dunphy. Mobile Internet Resource Center. Understanding cellular frequencies. 04 2023.

[5] Yu-Ngok Ruyue Li, Mengzhu Chen, Jun Xu, Li Tian, and Kaibin Huang. Power saving techniques for 5g and beyond. *IEEE Access*, 8:108675–108690, 2020.

[6] Martin L. Puterman. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley Sons, Inc., Hoboken, New Jersey, 1994.

[7] Ger Koole. Lecture notes stochastic optimization, 2006.

[8] Richard Weber. Optimization and control, 2016.

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

[10] Python Gym. Gym documentation.