Internship report

Multi-action bandits with non-binary action spaces

Pierre-Eliot Jourdan

LAAS-CNRS (SARA department)

June 10 - August 5 2022

Supervisors : Balakrishna Prabhu - Maaike Verloop

Contents

Chapter 1

Problem formulation

In this project, we consider a base station (BS) delivering packets to a certain number N of mobile devices. The data arriving to a device i is stored in a queue i. At a certain time slot, the number of packets arriving to queue i is Poisson distributed of rate λ_i.

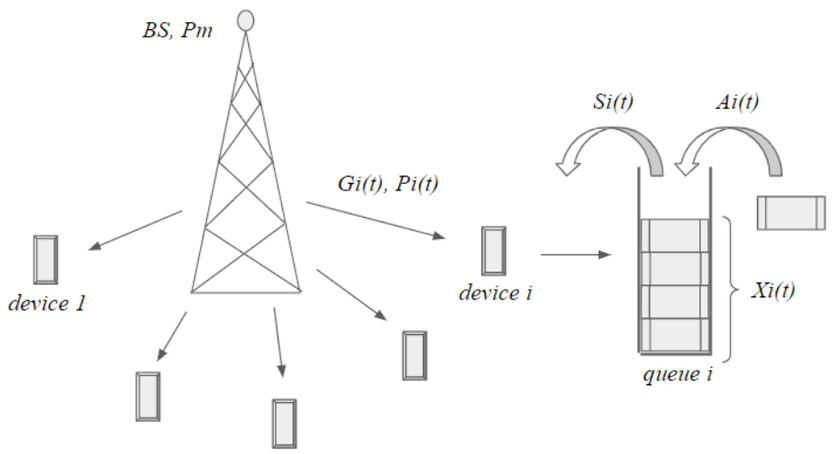


Figure 1.1: Illustration of the problem.

For each channel i, a random variable $G_i(t)$ gives its gain, that is to say the channel condition. The BS has a total power budget of P_{max} in each slot to distribute over the devices. If we note $R_i(t)$ the service rate of packets towards queue i, we have the **linear** formula : $R_i(t) = G_i(t)P_i(t)$, $P_i(t)$ being the power allocated to device i and $\forall i, \forall t, P_i(t) \geq 0$. This linear relation can be used only when the gain $G_i(t)$ takes small enough values. However, in this project we don't want to be limited to small gains this is why we will use a **log** rate function : $R_i(t) = \log(1 + G_i(t)P_i(t))$.

In this problem, we will work with continuous-time Markov chains (CTMC). The reason is that for discrete-time Markov chain (DTMC), several or no events can occur in a single time slot which makes lots of cases to take into consideration.

Let $X_i(t)$ be the random variable giving the number of packets in the queue i at time t. We can assume that the gain $G_i(t)$ is also a CTMC which makes the vector $(X_i(t), G_i(t))$ a CTMC as well when the power allocation is fixed.

Problem : The goal of this project is to search for the power distribution over all the devices in order to minimize the mean sojourn time of tasks. It means we search for the best power allocation to both our devices that satisfies :

$$\begin{cases} \min \lim_{T\to+\infty} \frac{1}{T} \int_1^T \sum_{i=1}^N X_i(t) \\ \forall t, \sum_{i=1}^N P_i(t) \leq P_m \\ \forall i, \forall t, P_i(t) \geq 0 \end{cases} \tag{1.1}$$

We will work on a simple example where there are only 2 mobile devices hence 2 random variables $X_1(t)$ and $X_2(t)$. We will suppose that queue 1 and 2 have a maximum capacity, thus $X_1(t)$ take its values in $\{0; K_1\}$ and X_2 in $\{0; K_2\}$ (maximum number of packets in the queues). $(X_1(t), X_2(t))$ will give us the state.

Here is how we will proceed : in chapter 2, we will have a theoretical approach in order to transform our problem and introduce the fundamental equations and notions to solve it. In chapter 3, we will focus on the performance of the model when the powers allocated to the queues are fixed. Finally we will test two different heuristic policies and compare their performance (chapters 4 and 5).

Stability criterion

It is also very important to make sure that our system is stable. This is a necessary condition to satisfy before applying any of the formula we will define next (the average or discounted costs for instance). To do that, we can plot the values of the rates $R_1(t)$ and $R_2(t)$ $(\forall t)$ on a graph (the policy area) and compare it to the position of (λ_1, λ_2). In order to be stable, it has to be in the domain (R_1, R_2) and the furthest from the border of it. As an example we get the following figure (with $g_1 = 20$ and $g_2 = 15$):

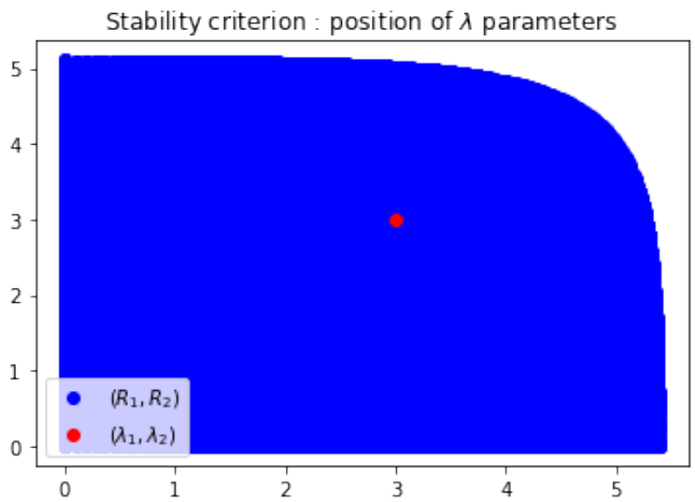


Figure 1.2: Stability of the system.

To obtain the blue area a set of values for $P_1(t)$ and $P_2(t)$ was selected such that $P_1(t) + P_2(t) \leq P_{max} = 11$. The log formula $(R_i(t) = \log(1 + G_i(t) \cdot P_i(t)))$ was used to get the scatter plot for $R_1(t)$ and $R_2(t)$. Here, our system is stable with $\lambda_1 = \lambda_2 = 3$.

Chapter 2

Theoretical approach

Our problem is written as an average-cost **Markov Decision Process (MDP)** (as mentionned in [4]) and we have constraints on the powers to take into consideration the total power budget P_{max} at each time instant. Because of these constraints, the traditional bandits model cannot be applied. Also, solving the MDP problem is complicated hence the necessity to search for heuristic policies.

2.1 Uniformization

The first step is to transform our problem into a discrete-time one. Thus, we will set the time slots so that only one event occurs in each of them : either one packet is arriving to a device's queue or one is leaving it. In this case, the time slots will not have the same duration and we can introduce the mean time slot duration γ with :

$$\gamma = \lambda_1 + \lambda_2 + \max(\max(R_1(t)), \max(R_2(t))) \cdot (\mu_1 + \mu_2), \forall t$$

This discretization of the problem is called **uniformization** and is mentioned in Martin L Puterman's book ([1]). It is also shown that finding an optimal solution with the **uniformization** method is equivalent to finding one in continuous time. Hence our **MDP** problem can be written as follows (the constraints are unchanged) :

$$\begin{cases} \min \lim_{T \to +\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} X_i(t) dt \\ \forall t, \sum_{i=1}^{N} P_i(t) \leq P_{max} \\ \forall i, \forall t, P_i(t) \geq 0 \end{cases} \tag{2.1}$$

The main goal of this project is to find good heuristic policies π for the problem (2.1).

2.2 Average and discounted cost

We can transform the previous equation using the **average-cost** optimality equation shown in [3]. In this new equation, we will consider $V((x_i, g_i), \pi, t)$ the cost for a certain mobile device to be in the state (x_i, g_i) at the time slot t and for a certain fixed policy π. We have :

$$V((x_i, g_i), \pi, t+1) = \frac{1}{t+1} \sum_{i=1}^{N} x_i + \frac{t}{t+1} M^{\pi} V((x_i, g_i), \pi, t) \tag{2.2}$$

In this equation, $M^\pi = M^\pi((x_i, g_i), (x'_i, g'_i))$ is the transition matrix giving the probabilities for a mobile device in the state (x_i, g_i) at time slot t to be in a state (x'_i, g'_i) at the next time slot $t+1$. M^π represents the **expected cost** of a state transition. On the other hand, the sum of packets in each device's queue $\sum_{i=1}^{N} x_i$ represents the **immediate cost** at time $t+1$.

We will use another expression to find the optimal solution of our problem. We can transform (2.1) using the **discounted cost criterion** ([3]) :

$$V((x_i, g_i), \pi, t+1) = \sum_{i=1}^{N} \beta^t X_i(t+1) \tag{2.3}$$

The parameter $\beta \in [0; 1]$ is called the **discount factor**. According to [2], a method is to use a **backward recursion**. We will apply the following recursive formula :

$$V((x_i, g_i), \pi, t+1) = \sum_{i=1}^{N} x_i + \beta M^\pi V((x_i, g_i), \pi, t) \tag{2.4}$$

With this method, the quantity we are interested in studying is $V((x_i, g_i), \pi, t+1) - V((x_i, g_i), \pi, t)$, for all states (x_i, g_i). This is what we will try to minimize and it will represent the cost of the mobile device to be in the corresponding state.

2.3 Algorithms

Now that we know the expression of the average (and discounted) cost we can define recursive algorithms to compute the successive values of $V^\pi(t)$ with a while loop (or a for loop). We will use the backward recursion of Markov reward chains. We can choose between the formula (2.2) or (2.4) :

Algorithm 1 (average cost):

Initialisation : $V^\pi(t=0) = V_0 = (0, 0, ..., 0), \epsilon = 0.01, \Delta = 10.$
1) $V = \frac{1}{t+1} X + \frac{t}{t+1} M^\pi V_0$
2) $\Delta = \max(V - V_0) - \min(V - V_0)$
3) $V_0 = V$
While $\Delta \geq \epsilon$
Return : V

Algorithm 2 (discounted cost):

Initialisation : $V^\pi(t=0) = V_0 = (0, 0, ..., 0), \epsilon = 0.01, \Delta = 10.$
1) $V = (1 - \beta) \cdot (X + \beta M^\pi V_0)$
2) $\Delta = \max(V - V_0) - \min(V - V_0)$
3) $V_0 = V$
While $\Delta \geq \epsilon$
Return : V

For both algorithms, the cost V will converge no matter which state (x_i, g_i) we consider. To prove this, we can study the evolution of each of the coefficients in V or the quantity $|V_0 - V|$ which is then supposed to converge toward 0. This is shown in algorithm 3 :

Algorithm 3 (convergence proof) :

Initialisation : $V^{\pi}(t=0) = V_0 = (0, 0, ..., 0)$, $\epsilon = 0.01$, $\Delta = 10$.
1) $V = X + M^{\pi}V_0$
2) $\Delta = \max(V - V_0) - \min(V - V_0)$
3) $res = |V_0 - V|$
4) $V_0 = V$
While res $\geq \epsilon$
Return : res

Chapter 3

Fixed gains, powers and policies

In this first part of the project, we will simplify our problem so that we can implement a solution. For now, the gains, the allocated power and thus the heuristic policy chosen will be considered fixed.

3.1 Modelling

The equations (2.2) and (2.4) can be written with matrices (we don't take the minimum for now as we are only working with a single fixed policy) :

$$V^{\pi}(t+1) = \frac{1}{t+1}X + \frac{t}{t+1}M^{\pi}V^{\pi}(t) \tag{3.1}$$

$$V^{\pi}(t+1) = X + \beta M^{\pi}V^{\pi}(t) \tag{3.2}$$

Let's determine the shape of the matrices X, M^{π} and $V^{\pi}(t)$.

X is a vector which contains the total number of packets in every state possible : $(0,0),(0,1),...,(K_1,K_2)$. X has a size of $(K_1+1)\times(K_2+1)$.

$$X = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ K_2 \\ 1 \\ \vdots \\ K_1 \times K_2 \end{pmatrix}$$

The vectors $V^{\pi}(t)$ also have a size of $((K_1+1)\times(K_2+1),1)$ and initially we assume that $V^{\pi}(t=0) = (0,0,...,0)^T$.

Our transition matrix M^{π} has a size of $((K_1+1)\times(K_2+1),(K_1+1)\times(K_2+1))$ which follows from the definition of the state space. As we assume only one event can occur at any time instant, we can notice that some states (x_i',g_i') are not accessible from a state (x_i,g_i) (transition probability = 0). Therefore, the transition matrix is block-tridiagonal :

$$M^{\pi} = \begin{pmatrix} \Theta_0 & R_0 & 0 & \cdots & 0 \\ L_1 & \Theta_1 & R_1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & L_{K_1-1} & \Theta_{K_1-1} & R_{K_1-1} \\ 0 & \cdots & 0 & L_{K_1} & \Theta_{K_1} \end{pmatrix}$$

The blocks Θ_i, R_i and L_i $(\forall i)$ are (K_2+1)-by-(K_2+1) matrices. The matrix L_i represents the transition between the states $(i,0),(i,1),...,(i,K_2)$ and $(i-1,0),(i-1,1),...,(i-1,K_2)$. It contains the probabilities for one packet to leave the queue 1 at a certain state and time slot : $p = \frac{\mu_1 r_1}{\gamma}$, r_1 being the rate of packets leaving the queue 1. $\gamma \geq \lambda_1 + \lambda_2 + \max(r_1, r_2)(\mu_1 + \mu_2)$ is the normalization term. Here is its shape of this matrix :

$$L_i = \begin{pmatrix} \frac{\mu_1 r_1}{\gamma} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{\mu_1 r_1}{\gamma} \end{pmatrix}$$

The matrix R_i represents the transition between the states $(i,0),(i,1),...,(i,K_2)$ and $(i+1,0),(i+1,1),...,(i+1,K_2)$, which contains the probabilities for one packet to arrive in the queue 1 at a certain state and time slot $(p = \frac{\lambda_1}{\gamma})$. Here is its shape of this matrix :

$$R_i = \begin{pmatrix} \frac{\lambda_1}{\gamma} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{\lambda_1}{\gamma} \end{pmatrix}$$

We have : $L_1 = L_2 = ... = L_{K_1}$ and $R_0 = R_1 = ... = R_{K_1-1}$. The shape of the matrices Θ_i are more complicated as they are not all equal. Indeed, depending on the position of the block in the matrix, the coefficients on its diagonal will be different. As M^{π} is a transition matrix, the sum of its rows have to be equal to 1.

The matrix Θ_i contains the probabilities for one packet to leave $(p = \frac{\mu_2 r_2}{\gamma})$ or to arrive $(p = \frac{\lambda_2}{\gamma})$ in the queue 2 or none of these events to happen (diagonal coefficients) at a certain state and time slot. We identify 3 different shapes in this family of matrices :

$$\Theta_0 = \begin{pmatrix} 1 - \frac{\lambda_1+\lambda_2}{\gamma} & \frac{\lambda_2}{\gamma} & 0 & \cdots & 0 \\ \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\lambda_2+\mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\lambda_2+\mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} \\ 0 & \cdots & 0 & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\mu_2 r_2}{\gamma} \end{pmatrix}$$

For all $i \in \{1, ..., K_1 - 1\}$:

$$\Theta_i = \begin{pmatrix} 1 - \frac{\lambda_1+\lambda_2+\mu_1 r_1}{\gamma} & \frac{\lambda_2}{\gamma} & 0 & \cdots & 0 \\ \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\lambda_2+\mu_1 r_1+\mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\lambda_2+\mu_1 r_1+\mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} \\ 0 & \cdots & 0 & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\lambda_1+\mu_1 r_1+\mu_2 r_2}{\gamma} \end{pmatrix}$$

$$\Theta_{K_1} = \begin{pmatrix} 1 - \frac{\mu_1 r_1 + \lambda_2}{\gamma} & \frac{\lambda_2}{\gamma} & 0 & \cdots & 0 \\ \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\mu_1 r_1 + \lambda_2 + \mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\mu_1 r_1 + \lambda_2 + \mu_2 r_2}{\gamma} & \frac{\lambda_2}{\gamma} \\ 0 & \cdots & 0 & \frac{\mu_2 r_2}{\gamma} & 1 - \frac{\mu_1 r_1 + \mu_2 r_2}{\gamma} \end{pmatrix}$$

3.2 Results

First the algorithm was tested with fixed values for the allocated powers ($P_1 = P_2 = 10$). We chose $\lambda_1 = \lambda_2 = 3$ to respect the stability criterion defined in the first part. We also set default values for the other parameters $\mu_1 = \mu_2 = 3.5$, $K_1 = K_2 = 10$, $g_1 = 20$, $g_2 = 15$ and the log rate function was used.

We can observe the behaviour of the solution we get with algorithm 1 when $T \to \infty$: we plot the values of the coefficient of the solutions for $t = 0, 1, ..., 1000$. We get the following Figure 3.1:

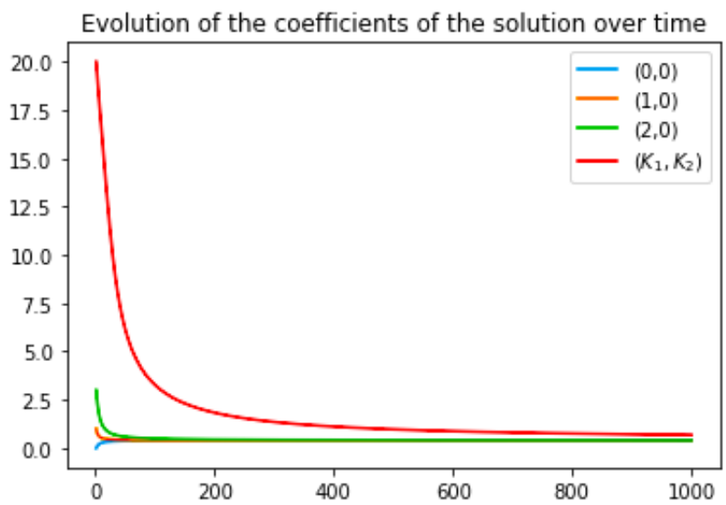


Figure 3.1: Convergence of the solution with average cost.

The result confirms the convergence of the solution in the long run and is coherent with the fact that the algorithm stops after a finite number of iterations.

Here we took fixed values for P_1 and P_2. It would be much more interesting to define policies π that could find the optimal power values in each state in order to find an average cost as small as possible.

Chapter 4

Policy methods

In this part, we still consider the gains as constants $G_1(t) = g_1$ and $G_2(t) = g_2$, $\forall t$ but we will study different power allocations to both devices which define different policies π. The goal of the following algorithms will be to find the optimal policy that minimizes the average or discounted cost.

4.1 Optimal policy : Bellman method

For numerical, we shall discretize the possible power allocations. We can, then, define different actions and compare their performance. We will have : $P_1 = [p_{1,1}, ..., p_{1,n_1}]$ and $P_2 = [p_{2,1}, ..., p_{2,n_2}]$ which gives us $n_1 \times n_2$ different actions $(a_1, ..., a_{n_1 \times n_2})$ where $a_1 = (p_{1,1}, p_{2,1})$, $a_2 = (p_{1,1}, p_{2,2})$,..., $a_{n_1 \times n_2} = (p_{1,n_1}, p_{2,n_2})$. By selecting the best actions among these (the ones that will minimize the cost), we can define an optimal policy : this is the Bellman method.

Algorithm 4 (Bellman) :

At each time instant, we compute the average cost V for every state possible and for every action.
Then, we will select the action that minimizes the vector V. We run the previous steps until the quantity $\Delta = \max(V - V_0) - \min(V - V_0)$ used in algorithms 1 and 2 is inferior to $\epsilon = 0.01$.
At the end of the algorithm, we will get a list containing the index of the best corresponding action for each state in the last time slot as well as the cost V obtained at the last iteration.

We will note $P_1^*(x_1, x_2)$ and $P_2^*(x_1, x_2)$ the optimal action in state (x_1, x_2).

Results

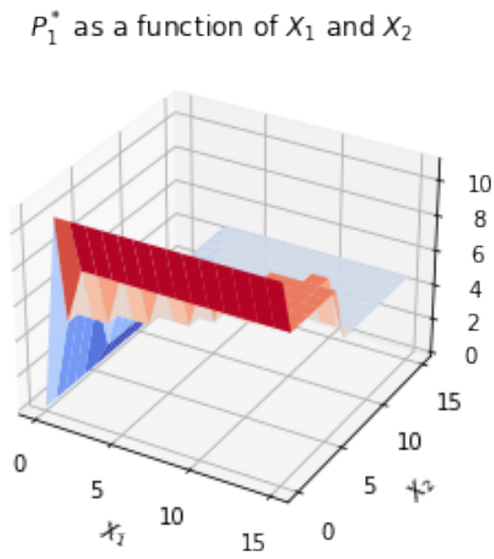
In this part, we will choose $P_{max} = 11$ and we will take a regular subdivision of $[0; P_{max}]$ for P_1 (with 8 values) and P_2 (with 5 values) :

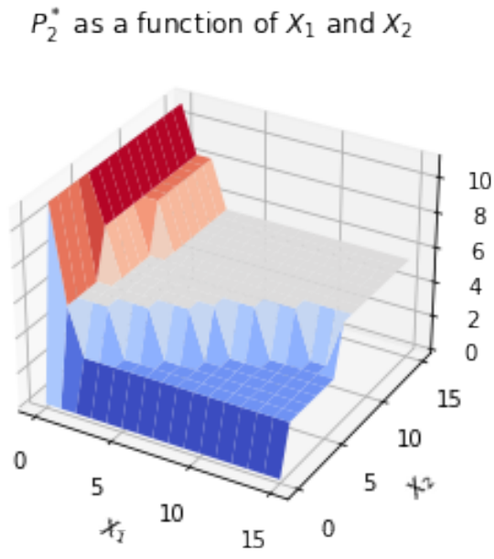
$$P_1 = [0, 1.571, 3.143, 4.714, 6.286, 7.857, 9.429, 11]$$

$$P_2 = [0, 2.75, 5.5, 8.25, 11]$$

Let's run the Bellman algorithm with the default parameters : $\lambda_1 = \lambda_2 = 3$, $\mu_1 = \mu_2 = 3.5$, $g_1 = 20$ and $g_2 = 15$.

We can better interpret the results if we plot the values of P_1^* and P_2^* (that we can get from the list of policies) as a function of X_1 and X_2. We get the following 3D figures ($K_1 = K_2 = 15$):

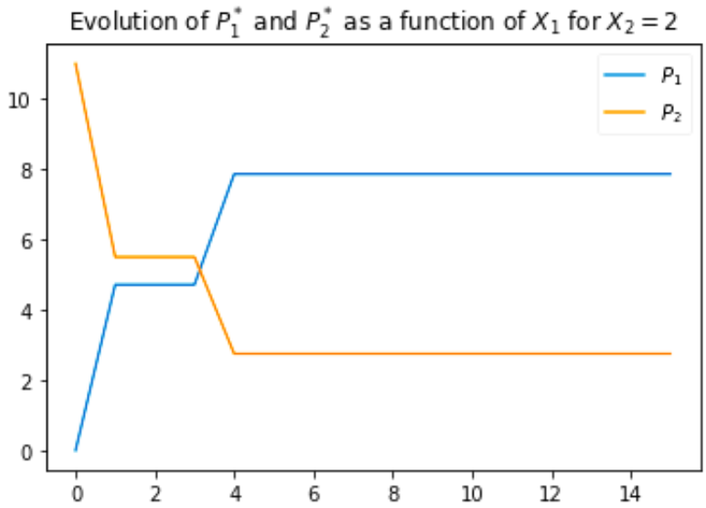


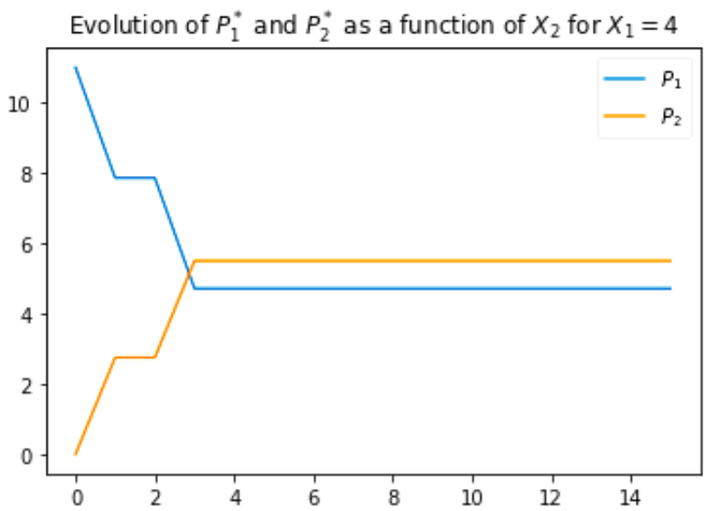


We notice that if we add the surfaces of P_1^* and P_2^* in the figures we almost obtain a plateau at $P = P_{max} = 11$ for every state possible. We can also see it when summing the matrices giving the values of P_1 or P_2 in every state : we obtain a matrix whose values are between 10.2 and 11 (P_{max}). This shows us that almost the entire power budget P_{max} is used in each time instant.

What we also notice is the fact that the power distribution doesn't seem to evolve when the values taken by the random variables X_1 and X_2 tend toward infinity (see the gray plateau on the figures above).

Another way to visualize is to fix X_1 (or X_2) and plot the values of P_1^* and P_2^* as a function of X_2 (or X_1). We give an example below with arbitrary values for X_1 and X_2 :





We can approximate the evolution of the powers P_1^* and P_2^* with polynomials to obtain a better line on the figure.

4.2 Occupation measures

The Bellman algorithm is nevertheless very costly as we test every possible sets of actions (allocation powers) and select the optimal one. Here is why we need to develop a heuristic.

We can define a linear program (LP) with the occupation measures $\psi_n(s, a, t)$ for individual devices as the control variables. The occupation measure $\psi_n(s, a, t)$ is the probability of taking action a in state s at time t for device n for a given policy π. Thus, the quantity we want to minimize here is written below (equation (4.1)) :

$$\min_{\pi}\{\sum_{k=1}^{N}\sum_{X_k,R_k} c_k(X_k, R_k)\cdot\psi_k(X_k, R_k)\} \tag{4.1}$$

where $c_k(X_k, R_k)$ is the cost of taking action R_k (R_k being the transmission rate) for device k in the state X_k and $N = 2$. In our example, $c_k(X_k, R_k) = X_k$.

Our original problem is under the constraint : $\forall t, \sum_{i=1}^{N} P_i(t) \leq P_{max}$. This gives us the following expression :

$$\sum_{k=1}^{N}\sum_{X_k,R_k} P_k\cdot\psi_k(X_k, R_k) \leq P_{max} \tag{4.2}$$

We also have other constraints to take into consideration :

$$\forall X_k, \sum_{R_k}\psi_k(X_k, R_k) = \sum_{X'_k,R'_k}\psi_k(X'_k, R'_k)\cdot M_k(X_k|(X'_k, R'_k)) \tag{4.3}$$

The previous equations can be written with matrices which give the following LP problem :

$$\begin{cases} \min\limits_{\pi} C^T\psi \\ A\psi \leq P_{max} \\ \psi D = \psi M' \\ \forall k, \sum_{X_k,R_k}\psi_k(X_k, R_k) = 1 \\ \forall i, \psi_i \geq 0 \end{cases} \tag{4.4}$$

ψ is a vector with 3 dimensions : $\psi[k][X_k][R_k]$. We vectorize it such that $\psi = (\psi_{k=1}, \psi_{k=2})$ with $\sum_{i=1}^{N}\psi_{k=1}[i] = \sum_{i=1}^{N}\psi_{k=2}[i] = 1$ and :

$$\psi_{k=1} = (\psi[1][X_1 = 0][R_{1,1}], \cdots, \psi[1][X_1 = 0][R_{1,l_1}], \psi[1][X_1 = 1][R_{1,1}], \cdots, \psi[1][X_1 = K_1][R_{1,l_1}])$$

$$\psi_{k=2} = (\psi[2][X_2 = 0][R_{2,1}], \cdots, \psi[2][X_2 = 0][R_{2,l_2}], \psi[2][X_2 = 1][R_{2,1}], \cdots, \psi[2][X_2 = K_1][R_{2,l_2}])$$

We can define ψ^* as the optimal solution of (4.4). $C^T\psi^*$ is the **lower bound** of the cost we can obtain with the chosen parameters : K_1, K_2, λ_1, λ_2, μ_1, μ_2... The matrix C contains the number of packets in a device's queue in every state, for all devices and for all rates R_k:

$$C^T = \begin{pmatrix} 0 & \cdots & 1 & \cdots & K_1 & 0 & \cdots & 1 & \cdots & K_2 \end{pmatrix}$$

The matrix A contains the power levels in every state and for both devices :

$$A = \begin{pmatrix} P_1 & \cdots & P_1 & P_2 & \cdots & P_2 \end{pmatrix}$$

with $P_1 = [P_{1,1}, P_{1,2}, ..., P_{1,l_1}]$ and $P_2 = [P_{2,1}, P_{2,2}, ..., P_{2,l_2}]$ ($l_1 = len(P_1), l_2 = len(P_2)$).

The matrix D is used to sum the values of μ over the rates R_k :

$$D = \begin{pmatrix} D_{10} & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & & & & \vdots \\ \vdots & & D_{1K_1} & & & \vdots \\ \vdots & & & D_{20} & & \vdots \\ \vdots & & & & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & D_{2K_2} \end{pmatrix}$$

with $D_{10} = D_{1K_1} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$ (size $(1, l_1)$) and $D_{20} = D_{2K_2} = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$ (size $(1, l_2)$).

Finally, the matrix M' is the transition matrix between the states X_k for different rate values R_1 or R_2. It has the shape of a block diagonal matrix as follows :

$$M' = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix}$$

where M_1 and M_2 have the same shape with different parameters ($\lambda_1/\lambda_2, \mu_1/\mu_2...$). Let's write the matrix M_1 (block-tridiagonal) where every submatrices $M_{i\to j}$ is a transition matrix between the states $X_1 = i$ and $X_1 = j$. We have :

$$M_1 = \begin{pmatrix} M_{0\to 0} & M_{0\to 1} & 0 & \cdots & 0 \\ M_{1\to 0} & M_{1\to 1} & M_{1\to 2} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & M_{K_1-1\to K_1-2} & M_{K_1-1\to K_1-1} & M_{K_1-1\to K_1} \\ 0 & \cdots & 0 & M_{K_1\to K_1-1} & M_{K_1\to K_1} \end{pmatrix}$$

For $i \in \{0, K_1\}$:

$$M_{i\to i-1} = \begin{pmatrix} \frac{\lambda_1}{\gamma} \\ \vdots \\ \frac{\lambda_1}{\gamma} \end{pmatrix}$$

$$M_{i\to i+1} = \begin{pmatrix} \frac{\mu_1 R_1[0]}{\gamma} \\ \vdots \\ \frac{\mu_1 R_1[l_1]}{\gamma} \end{pmatrix}$$

For $i \in \{1, K_1 - 1\}$

$$M_{i\to i} = \begin{pmatrix} 1 - \frac{\lambda_1 + \mu_1 R_1[0]}{\gamma} \\ \vdots \\ 1 - \frac{\lambda_1 + \mu_1 R_1[l_1]}{\gamma} \end{pmatrix}$$

$$M_{0\to 0} = \begin{pmatrix} 1 - \frac{\lambda_1}{\gamma} \\ \vdots \\ 1 - \frac{\lambda_1}{\gamma} \end{pmatrix}$$

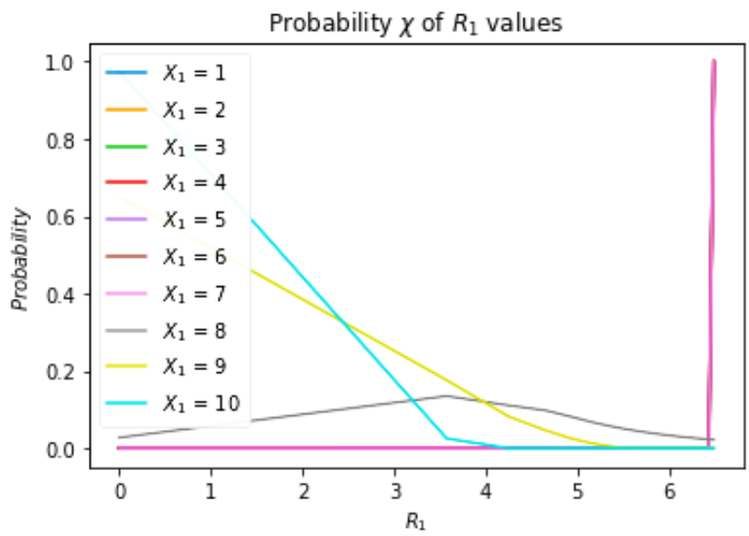
$$M_{K_1\to K_1} = \begin{pmatrix} 1 - \frac{\mu_1 R_1[0]}{\gamma} \\ \vdots \\ 1 - \frac{\mu_1 R_1[l_1]}{\gamma} \end{pmatrix}$$

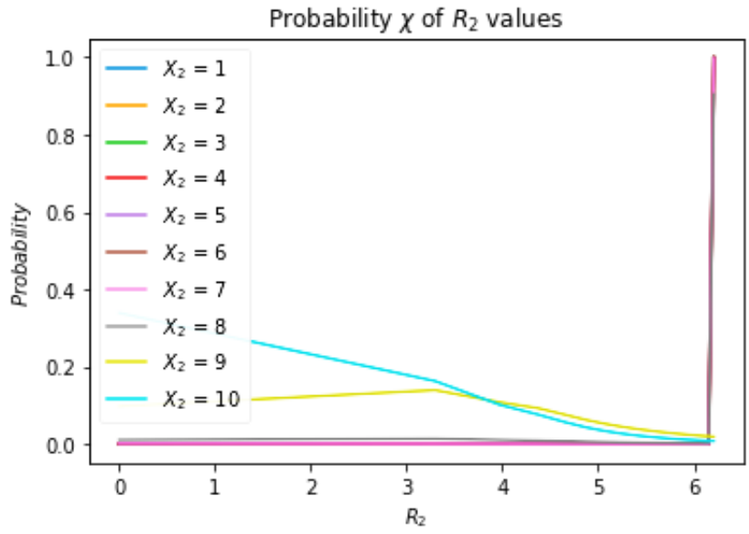
Once we have the solution of the LP problem, we have lower bound value of the average cost related to our initial problem as well as the corresponding occupation measure μ. From this result, we can build some index policy and compare it to the Bellman method.

First let's compute the following vector χ such that :

$$\chi_k(X_k, R_k) = \frac{\psi_k^*(X_k, R_k)}{\sum\limits_{R'_k} \psi_k^*(X_k, R'_k)}$$

The vector χ gives the probability that action R_k is chosen under policy ψ^* given that we are in state X_k. We plot below the probability of R_1 and R_2 values depending on the state X_1 or X_2 (with $K_1 = K_2 = 10$) :





This vector will be used in the following index function :

$$\Phi_k(X_k) = \sum_{R_k} \chi_k(X_k, R_k) \cdot c_k(X_k, R_k)$$

This index function defines the **Occupancy-Measured-Reward (OMR)** index policy ([4]).

Algorithm 5 (OMR - Index in state X_k) :

If $\Phi_1(X_1) \geq \Phi_2(X_2)$: we randomly chooses the value of P_1 according to $\chi_1(X_1, R_1)$ for device 1 and device 2 will get $P_2 = P_{max} - P_1$.
Else : P_2 will be chosen randomly according to $\chi_2(X_2, R_2)$ and $P_1 = P_{max} - P_2$.

Once we found the allocated powers P_1 and P_2 with this binary decision, we may apply the average cost algorithm and compare it to the optimal policy (Bellman).

Chapter 5

Policy comparison

Let's compare the average cost V obtained with the OMR index policy (V_{OMR}) and with the optimal one ($V_{Optimal}$ with the Bellman method). We will execute both algorithm until $\max(V) - \min(V) \leq \epsilon = 0.01$ (while loop), which corresponds to around 37000-40000 iterations. The rest of the parameters will be chosen as in section 2.1 (Bellman method).

To make sure we obtain coherent values, we will choose to select values for λ_1 and λ_2 in [3;4]. We can also test higher values for λ_1 and λ_2 but we will have to select also higher values for K_1 and K_2 in order to avoid the side effects due to the new position of (λ_1,λ_2) and maintain the stability of our system.

Finally, in order to better analyse the results we can introduce the suboptimality gap ϵ (in percentage) :

$$\epsilon = \frac{V_{OMR} - V_{Optimal}}{V_{Optimal}} \times 100$$

The suboptimality gap will give us information about how far from the optimal policy, the OMR index one is.

We set the values of the "Standard" parameters as the ones used to run the Bellman algorithm : $K_1 = K_2 = 10$, $P_{max} = 11$, $\lambda_1 = \lambda_2 = 3$, $g_1 = 20$, $g_2 = 15$ and $\mu_1 = \mu_2 = 3.5$.

Parameters modification	Lower bound	$V_{Optimal}$	V_{OMR}	ϵ
Standard	0.3123	0.3726	0.5235	40.5 %
$P_{max} = 21$	0.2794	0.3454	0.4655	34.7 %
$P_{max} = 5$	0.3646	0.4274	0.6517	52.5 %
$\lambda_1 = 4$	0.3738	0.4407	0.6833	55.1 %
$\lambda_1 = 5$	0.4483	0.5241	0.8124	55 %
$\lambda_2 = 4$	0.3778	0.4485	0.6913	54.1 %
$\mu_1 = 0.7$	2.7938	3.4521	7.9543	130 %
$\mu_2 = 0.7$	3.2179	4.6264	10.3742	124 %

We can also plot the evolution of the cost of both policies over time to highlight this behaviour. You can observe this evolution (with the standard parameters) on figure 5.1 :

The results in the table show that if we increase the values of the parameters λ_1 and/or λ_2, the average cost of the algorithm will **increase** as well, which makes sense since we increase the rate of packets arriving to the devices' queue (Poisson distribution with parameter λ_i).

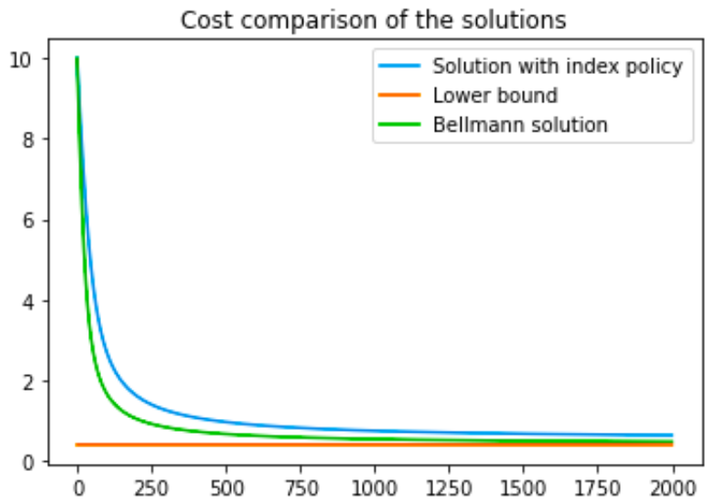


Figure 5.1: Evolution of average cost over time.

On the contrary, if we increase the values of μ_1 and/or μ_2, the average cost will **decrease** as the number of packets leaving one device's queue is represented by the random variable $S_i(t)$ which is exponentially distributed with parameter μ_1/μ_2. We get the same observation when increasing the value of P_{max}, the total power allocation in each time slot.

From numeric we can see that, in some settings, the OMR works rather well (suboptimality gab $= 35\% - 40\%$ from optimum), but in other settings it is still far from optimal. Thus, in future work we want to **investigate how to adapt our heuristic**.

Also we can notice by running the algorithms that already for two queues it takes very long to obtain the optimal policy (using Bellman). With more queues, it will quickly **explode** the time that it takes to find the optimal policy (curse of dimensionality). On the other hand, the index policy is fast to obtain for one queue. So if there are many queues, the time length will just scale with the number of queues (hence does not explode). This means the OMR index policy is also a good solution.

NB : In our example, we get an execution time of around **455 minutes** for the Bellman algorithm whereas it takes only **25 minutes** to run the OMR index policy algorithm. This is also due to the fact that with Bellman, every action (possible combination of P_1 and P_2) is studied and compared at every iteration. On the contrary, for the OMR policy algorithm, we already know the values of P_1 and P_2 to give to each device according to the current state.

Conclusion

In this project,we worked on a special case of the multi-armed bandits problem applied to downlink power control in wireless networks. We implemented policies in order to take some actions in our environment (the wireless network) that would both optimize the distribution of packets and be the less costly. It was necessary to introduce the notions of average or discounted cost as well as some mathematical models to study their evolution over time. By defining policies (represented by a set of actions), we were able to compare their performance.

We compared the performance of two policies : the optimal policy (Bellman method) and the OMR index policy. We also computed, using linear programming, the lower bound of the cost associated with our problem. Our work showed that the index policy performs well and is easier to obtain than the optimal one as it is computationally less expensive. Using the Bellman method will be very costly but we are sure we will take the best actions for our problem (optimal policy).

Bibliography

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

[2] Ger Koole. *Lecture notes Stochastic Optimization*. 2006. URL: `http://www.math.vu.nl/obp/edu/so`.

[3] Richard Weber. *Optimization and Control*. 2016.

[4] Rahul Singh, Jian Li, and Guojun Xiong. "Reinforcement Learning for Finite-Horizon Restless Multi-Armed Multi-Action Bandits". In: *AAAI 2022*. 2022.