Deploying a 5G Network in a country through Metropolis-Hasting algorithm

Team Aquarium
Loïc Busson, Riccardo Cadeï, Anita Dürr
COM-516, EPFL, Switzerland

December 13, 2020

Abstract

In telecommunications, 5G networks are the next generation of broadband cellular networks. Telecommunication companies are actively testing and starting to roll them out in different parts of the world. The task of this project is to deliver a roadmap to test this new network while optimizing the cost of the maintenance of the new installations. We propose a Metropolis-Hastings based algorithm to find an approximate solution to this problem using a simple symmetric random walk on a hypercube as a base chain.

Introduction

The goal of this project is to use Markov chain methods to approximate the maximum $S^*(\lambda)$ of the following function, described in the problem statement:

$$f_\lambda(S) = \sum_{i\in S} v_i - \lambda \cdot n \cdot \frac{\pi}{4} \max_{(i,j)\in S^2} d(x_i, x_j)^2$$

For fixed values of λ, maximizing $f_\lambda : \mathcal{P}(\{0, ..., n-1\}) \to \mathbb{R}$ is the same problem as minimizing its opposite function $g_\lambda := -f_\lambda$. So we will directly apply the Metropolis-Hastings algorithm seen in the lectures. For notation purposes, we write g for the function g_λ. We also re-numerated the cities from 0 to $(n-1)$. This doesn't change anything to the optimal value to approximate. Finally, we call $\tilde{S}$ our approximation of S^*.

In Section 1 we formulate the application of Metropolis-Hasting algorithm to this problem, justifying our choices of base chain and implementation (Question 1). In Section 2 we present how we tuned the hyper-parameters of this model, namely the starting state and the list of βs (Question 1). Finally, in Section 3 we summarize the results of our implementations for two particular instance generators (Question 2 and 3).

1 Algorithm and implementation

As we want to minimize the function g, ideally we would sample from the distribution $\pi_\infty(x) = \mathbb{1}\{x \text{ is a global minimum of g}\}/Z_\infty$ (with Z_∞ the normalization factor). As this is not an easy task, we will instead sample from the distribution $\pi_\beta(x) = e^{-\beta g(x)}/Z_\beta$.

Sample from π_β. For a fixed value of β, we will use the Metropolis-Hastings algorithm to sample from π_β. The general idea is to construct a Markov chain having π_β as its stationary distribution. Running the Markov chain for a certain among of steps –to be determined empirically– we will be able to approximate π_β. To construct this chain, we need to define a *base chain*, *acceptance probabilities* and how both relate to the *new chain*, the chain with stationary distribution π_β. This will be detailed below.

Sample from π_∞. The interest in π_β is that it approximates π_∞ when β grows to infinity. However, if we run the algorithm with a big β, the chain will have a high probability to get stuck at a local minimum. On the other hand, small β's allow the chain to explore a big number of states. This is why we have chosen to run the Metropolis-Hastings algorithm with increasing values of β. This strategy is called *simulated annealing*.

1.1 Base chain

We use the *Symmetric Random Walk* on the hypercube $C = \{0,1\}^n$ as a base chain. The states are notated as vectors x of dimension n and the transition probabilities are $\psi_{xy} = \frac{1}{n}$ if x and y are neighbours and $\psi_{xy} = 0$ otherwise. We say that x and y are neighbours if the two vectors differ by exactly one digit.

Our choice for this base chain is that a state x of this chain describes exactly one subset S_x of cities taken in $\mathcal{P}(\{0,...,n-1\})$: the i-th digit in x indicates whether or not the city i is taken in S_x. This chain has furthermore the necessary hypothesis to be a base chain: it is irreducible and $\psi_{xy} > 0$ iff $\psi_{yx} > 0$. However it is periodic, but one can show (using the same argument of Homework 8 ex. 3) that the thesis of Metropolis-Hasting Theorem is valid even if the base chain is not aperiodic.

1.2 Acceptance probabilities

The acceptance probabilities are defined as $a_{xy} = \min(1\ ;\ \frac{\pi_\beta(y)}{\pi_\beta(x)})$. By writing $\Delta(x,y) := g(y) - g(x)$, we get that for x and y neighbours, $a_{xy} = \min(1, e^{-\beta\Delta(x,y)})$. Otherwise, $a_{xy} = 0$.

1.3 New chain

The transition probabilities of the new chain is defined as $p_{xy} = \psi_{xy}a_{xy}$ if $x \neq y$ and $p_{xy} = 1 - \sum_{v\neq x}\psi_{xv}a_{xv}$ if $x = y$. However, we won't have to compute it as explained in the next paragraph.

1.4 Our implementation

Propose a move. In our implementation of this algorithm, we indeed don't need to compute all the p_{xy}, which would be quite expensive. Instead, at each step we propose a move from x to y in the base chain and then accept it in the new chain with probability a_{xy}. All possible moves are between two neighbour states, i.e. vectors that differ only at the k-th digit. Since all moves are uniformly distributed, to chose a move is simply to randomly sample the digit k.

Compute the acceptance probability for a proposed move To compute a_{xy}, we need to compute $\Delta(x,y)$. However we can do this efficiently, without having to compute both $g(y)$ and $g(x)$. For x and y neighbours, we have the following relation:

$$\begin{aligned}
\Delta(x,y) &= g(y) - g(x) \\
&= f_\lambda(x) - f_\lambda(y) \\
&= [\sum_{i\in S_x} v_i - \lambda\cdot n\cdot\frac{pi}{4}\cdot\max_{(i,j)\in S_x^2} d(x_i,x_j)^2] \\
&\quad - [\sum_{i\in S_y} v_i - \lambda\cdot n\cdot\frac{pi}{4}\cdot\max_{(i,j)\in S_y^2} d(x_i,x_j)^2] \\
&= v_k\cdot(x[k]-y[k]) - \lambda\cdot n\cdot\frac{pi}{4} \\
&\qquad \cdot[\max_{(i,j)\in S_x^2} d(x_i,x_j)^2 - \max_{(i,j)\in S_y^2} d(x_i,x_j)^2]
\end{aligned}$$

First of all, we decided to pre-compute the matrix of distances between all the cities in the dataset. Furthermore note that instead of computing twice the maximum distance for S_x and S_y, we need only to compute the maximum distance for the set containing the city k. If the k-th city doesn't intervene in the maximum distance, then this maximum distance is equal to the maximum distance for the set without it. Hence the difference is null and $\Delta(x,y) = v_k\cdot(x[k]-y[k])$. Otherwise, if the city k increases the maximum distance, then we need to compute the maximum distance for the set without city k and do the difference. This slightly improves the computation of $\Delta(x,y)$.

Memorize the maximum. The goal is to maximize the function f_λ. However the ending state of the Markov chain may not be the maximizing one, even if it had once passed through it. This is why during the whole process, we are keeping in memory the best state among all visited states. It might be expensive to evaluate the objective function at each step. Note however that for a proposed move from state x to y, $f_\lambda(y) = f_\lambda(x) - \Delta(x,y)$. As we already need to compute $\Delta(x,y)$ in order to have the acceptance probability, the evaluation of the objective function f at each step becomes a simple task.

2 Choose the parameters

Even if the structure of the algorithm is defined, we still need to tune some parameters such as the starting state of the chain or the values of β to use. We justify our choices in the following paragraphs, using as a reference the model $\mathcal{G}_1$.

Starting state, Since the domain of f is very big and it increase exponentially with the number of cities, different starting states for the same model for the same instance of the problem can conduct to different approximated solution (e.g. local maxima). We compared three different starting states for 30 instances both for $\lambda = 0.5$ and $\lambda = 1$: a random starting state, the state containing only the most crowded city and the one containing all the cities. We observed that for both the values of λ the second starting state gives the best results and in particular for $\lambda = 1$ there is strong statistical evidence to say that provided by an one-way Anova test ($F - statistic = 26.9358, p - value \approx 0$).

The scale of β. As we explained above, the Metropolis-Hastings algorithm has to be run with several increasing values of β. A log-scale allows to slowly increase the values at the beginning while quickly ending with higher ones. By using this scale (with *numpy*'s `logspace()` function), we make sure that during the first rounds of the Metropolis-Hastings algorithm, the chain doesn't get too quickly stuck in a local maximum of f_λ. We rather allow it to pick some minimum as long as β stays small. We don't need to be that careful with bigger values of beta as the convergence of the chain appears quickly. This is why we prefer to end with a few big values only. This pattern is not presented in a linear scale.

However, we still need to choose the range of β as well as their number. We proceed by a grid search with $\lambda = 0.5$ over different number of β in a range from 1 to $10^{\texttt{max beta}}$. We fixed the total number of steps of the Markov chain to be `total_n_iter = 1 000 000`. This means that for a given number of betas `nb_beta`, we use `total_n_iter // nb_beta` iterations per Metropolis-Hastings algorithm. The result of the grid search is displayed in Table 1.

Observe that both the maximum value of beta and the number of betas influence the approximated maximum value of the objective function. We decided to use the parameters giving the biggest approximated maximum, i.e. in our experience maximum value of β $10^{\mathbf{3}}$ and **7** different values of β.

Maximum β (10-th power)	Numbers of β	Average approximate maximum
3	5	5.476
3	**7**	**5.721**
3	10	4.994
3	13	5.721
4	5	1.618
4	7	5.712
4	10	5.721
4	13	4.861
5	5	2.2
5	7	4.047
5	10	5.593
5	13	5.715

Table 1: *Grid Search result over the range of β.*

3 Behaviour of the Markov chain with models $\mathcal{G}_1$ and $\mathcal{G}_2$

3.1 Evolution of the Markov chain during the algorithm

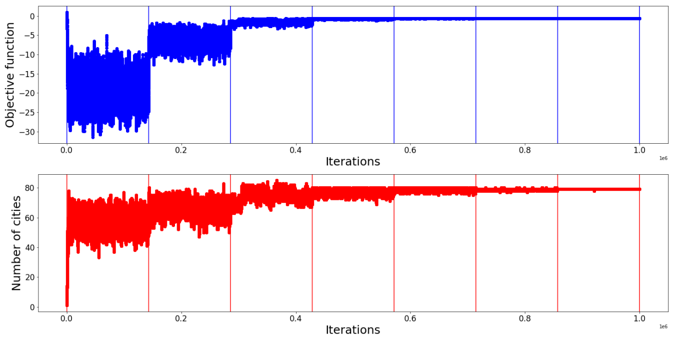

Figure 1: *Evolution of the state visited by the Markov chain with an instance of $\mathcal{G}_1$: the evaluation of the objective function on the visited state is displayed in blue ; the size of the visited state is displayed in red. Vertical lines represent the iterations for which the value of β was changed.*

In Figures 1 and 2, we observe as expected that for small values of β, the chain is moving a lot between states of different sizes while for bigger βs, the chain tends to stay put in one state. This change of behaviour is clearly marked at the change of βs –the vertical lines. We can also note that the global behaviour of the chain is indeed to go to states with bigger values of the objective function. The value of the objective function seems to converge more quickly than the size of the set S. This is because different states (of different sizes) can give similar values of f and that a bigger β is then required for the chain to decide in which state to stay.

If we look at the behaviour of the chain for the first value of $\beta = 1$, in both models the chain reached at the beginning a state with a high objec-

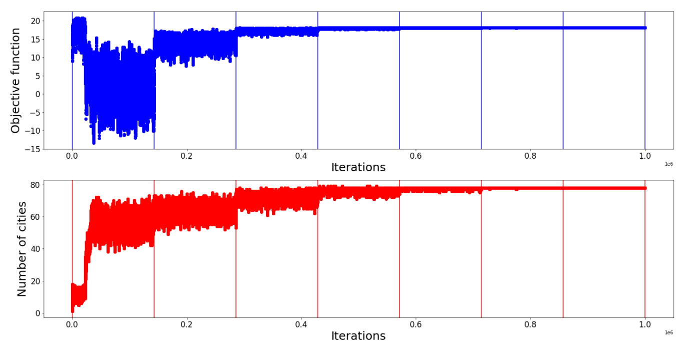

Figure 2: *Evolution of the state visited by the Markov chain with an instance of $\mathcal{G}_2$: the evaluation of the objective function on the visited state is displayed in blue ; the size of the visited state is displayed in red. Vertical lines represent the iterations for which the value of beta was changed.*

tive function, which was never reached afterwards. This isn't such an issue, as we memorize the best state visited and thus output this particular value. However it is interesting to remark that the chain, starting at states selecting only a few cities, quickly goes to states selecting between 40 and 70 cities, even though this choice didn't improve the value of the objective function. This is especially noticeable in Figure 2 were the chain stayed in optimal states for a short time before moving to those average states. In fact, states in the "center" of the hypercube, in contrast to "edge" states, are more likely to be reached: there are many ways to go from a given state to the center. This property leads to a chain that most likely gets stuck in the "center" of the hypercube. Once it reaches such a state, it is very difficult for it to leave this subset. With another base chain in which all the states are equally likely to be reached (something like a hyper-cycle), other states could have been reached and maybe a better maximum would have been found. This observation justifies again the choice of starting the chain at an "edge" state, which otherwise would be visited with a small probability only.

3.2 Influence of λ on the maximum

Assuming that $\tilde{S}(\lambda) \approx S^*(\lambda)$ $\forall\lambda \in [0,1]$ we can evaluate the value of $\mathbb{E}_{\mathcal{G}_1}[f(\lambda, S^*(\lambda))]$ and $\mathbb{E}_{\mathcal{G}_1}[|S^*(\lambda)|]$ with respect to λ knowing $\tilde{S}(\lambda)$ and approximating the expected value with the sample mean. In Figures 3, we can observe as expected that the approximated maximum of f and the size of $\tilde{S}$ decrease as λ increases. Notice though that the approximated maximum of f is always non negative since the value of f evaluated on the starting state is equal to the normalized population of the most crowded city ($\geqslant 0$); and the size of $\tilde{S}$ is always positive (> 0) because f evaluated on the starting state is non negative ($\geqslant 0$) and $f(\lambda, \varnothing) = 0$ $\forall\lambda \in [0,1]$. We correctly note that $\mathbb{E}_{\mathcal{G}_1}[|\tilde{S}(0)|] = 100$, since $|S^*(\lambda)| = 100$ and that for the law of large numbers $\mathbb{E}_{\mathcal{G}_1}[f(0, \tilde{S}(0)] \approx |S^*(0)| \cdot \mathbb{E}_{\mathcal{G}_1}[v] = 50$.

The general trend of the expected approximated maximum of f is similar to a power law while the trend of the expected size of $\tilde{S}$ can be interpreted as a sigmoid function (the curve is concave for (approx.) $\lambda < 0.5$ and becomes convex for (approx.) $\lambda > 0.5$).

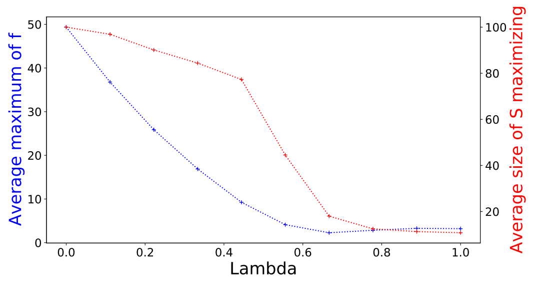

Figure 3: *Evolution of* $\mathbb{E}_{\mathcal{G}_1}[f(\lambda, \tilde{S}(\lambda))]$ *(blue) and* $\mathbb{E}_{\mathcal{G}_1}[|\tilde{S}(\lambda)|]$ *(red) with respect to* λ

We get same trends for the second instances generator too, as expressed in Figure 4. The differences between the two sampling are that in the first one the number of people for each city is sampled as a Uniform distribution, in the second one as log-Normal and so in this new case the majority of cities have few citizens and few outlier cities are very crowded. This difference doesn't impact the trend of the 2 functions but since $\mathbb{E}_{\mathcal{G}_2}[v] = exp(\mu + \frac{\sigma^2}{2}) = 0.8187$ ($> \mathbb{E}_{\mathcal{G}_1}[v] = 0.5$) then $\mathbb{E}_{\mathcal{G}_2}[f(0, \tilde{S}(\lambda))] \approx |S^*(0)| \cdot \mathbb{E}_{\mathcal{G}_2}[v] = 82$ and $\mathbb{E}_{\mathcal{G}_2}[|S^*(\lambda)|]$ changes the concavity later in $\lambda = 1$.

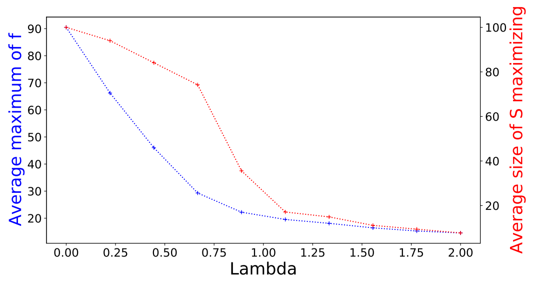

Figure 4: *Evolution of* $\mathbb{E}_{\mathcal{G}_2}[f(\lambda, \tilde{S}(\lambda))]$ *(blue) and* $\mathbb{E}_{\mathcal{G}_2}[|\tilde{S}(\lambda)|]$ *(red) with respect to* λ