CAM Project : Internship Feedback Extended Abstract

Multi-action bandits with non-binary action spaces for wireless networks

Pierre-Eliot Jourdan

ENSEEIHT, Numerical Science Department, 2022

Abstract

This paper explores some ways to optimize the distribution over time of data packets in a wireless network composed of a base station and some devices. Several parameters were taken into account like the gain of the transmission, the distribution rate of the packets or the number of packets in the devices' queue in each time slot.

We transformed our problem into a Reinforcement Learning one that is to say we considered an agent taking actions in an environment with constraints : our actions being the distribution of a data packet to a certain device at time t and our environment being our wireless network. Two functions were defined to measure the cost of every action : discounted and average cost. Then we used policy functions that determine which packets should be sent to which device. We could compare the performance of an optimal policy and a heuristic computed with linear programming.

The results show that the optimal policy gives the best performance but remains very time-costly. On the other hand, the heuristic provides quite satisfying results although these ones could be improved.

Thus, this paper shows how heuristics prove to be a good alternative to optimal policies in the solution of our Reinforcement Learning problem.

Keywords

Optimization, Reinforcement Learning, Multi-action bandits, policies, wireless networks

Introduction

Today, Reinforcement Learning (RL) appears as a useful tool in many optimization problems to build more reliable and high-performance networks. This paper illustrates one of them in a wireless network composed of a base station (BS) delivering data packets to two devices.

In each time slot, the BS has a maximum power budget to distribute ($P_{\max}$). Each transmission channel has a constant gain (g_i) that defines its condition. Random variables are also used to model the number of packets leaving the BS or arriving at one device‘‘s waiting queue.

The goal was to develop an algorithm to optimize the distribution of packets over time by minimizing the number of packets in the network.

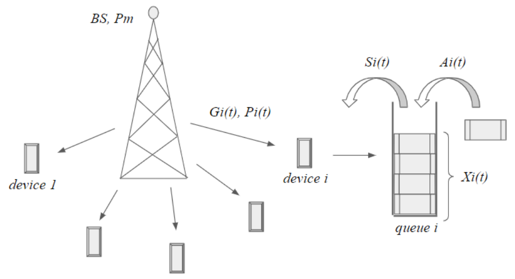

Fig.1. Illustration of the problem.

Details of work and research

First of all, we chose to work with continuous time to ensure that only one action could occur in each non-regular time lapse (either one packet leaves or arrives). Then we discretized our problem as said in [1] to implement solutions :

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

We also supposed that each queue has a maximum capacity (K_i). The number of packets leaving and arriving in queue i is respectively exponential and Poisson distributed with parameters μ_i and λ_i.

We can assimilate our situation as an agent choosing which action should be taken in the next time slot. A set of actions selected is called a policy. This is a variant of a RL problem called Multi-action bandits, in which a function rewards the agent if he selects a good action depending on the state of the problem (number of packets in each queue). For our case, cost functions were chosen instead.

We defined an average cost (AC) function that would compute the mean cost of an action possible at time t. [3] gave us the following recursive formula for a fixed policy π:

$$V((x_i, g_i), t+1) = \frac{\sum_{i=1}^{N} x_i + tM^{\pi}V((x_i, g_i), t)}{t+1} \quad (2)$$

We also used a discounted cost (DC) formula (obtained with backward recursion in [2]) :

$$V((x_i, g_i), \pi, t+1) = \sum_{i=1}^{N} x_i + \beta M^{\pi} V((x_i, g_i), \pi, t) \quad (3)$$

The objective was to minimize the AC or DC over time. Once we were done with problem formulation, we could start implementing solutions.

The first method called Bellman gave us the optimal policy for our problem. Indeed, this method consists in computing the cost for every possible action and selecting the one that minimizes it, in each time slot. However, this algorithm is very costly since there are many possibilities to examine at every iteration.

Hence the necessity to define a heuristic that will take much less time. We chose to use another policy called Occupancy-Measured-Reward (OMR) index explained in [4]. It uses a quantity called occupation measure, computed with linear programming, giving us the time spent in every state.

We tested both algorithms for $P_{\max} = 11$, $K_1 = K_2 = 10$, $g_1 = 20$, $g_2 = 15$, $\lambda_1 = \lambda_2 = 3$ and $\mu_1 = \mu_2 = 3.5$. We can plot the evolution of the AC for 2000 iterations :

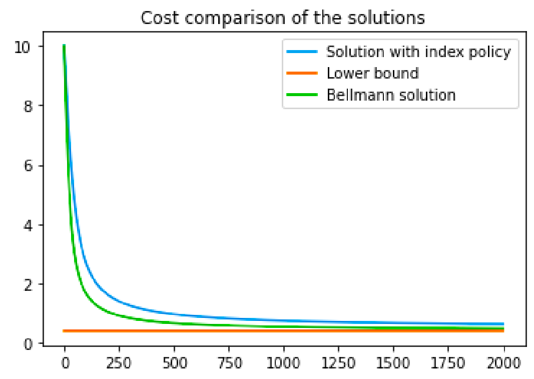

Fig.2. Evolution of the AC over time for both solutions.

The AC converges towards 0.37 for the optimal policy and towards 0.52 for the heuristic (OMR).

Feedback reflection

This internship was my first professional experience in a laboratory and I was able to have an overview of the world of research. I clearly used my skills in applied mathematics, AI and Python with a concrete application in the domain of telecommunications.

Overall, my supervisors were very happy with my work, as I managed to do more than what they expected in the beginning of the internship : the optimal policy was correctly implemented since it seems to converge very close to the lower bound (fig 2). On the other hand, the result obtained with the heuristic could have been improved as the AC converges towards a too big value. This could have been done if the internship was a few weeks longer.

On a more personal level, this professional experience was very enriching for me as I discovered and developed skills in two domains (RL and optimization) in which I would like to specialize next year. I could then pursue my work in the frame of a PhD or other research projects.

Conclusion

The results made it clear that the Bellman method is way too costly to be used as we got an execution time of nearly 7 hours. On the contrary, the OMR algorithm only needs 25 minutes to converge, which makes it a more suitable solution. Also, according to fig 2, the index policy gives a good result since its AC gets quite close to Bellman's.

This paper aimed at giving the reader a simple example of how RL can help to optimize the telecommunication networks we use everyday. The work could be extended to more complex problems or used to search for more efficient solutions that may be useful with the expansion of 5G and 6G very soon...

References

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

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

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

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