Optimizing Channel Allocation in RF and Network Engineering: A Combinatorial Multi-Armed Bandits Approach

Vikram Vasudevan
vikramva@usc.edu

Faezeh Dehghan Tarzjani
dehghant@usc.edu

***Abstract*—In RF and Network Engineering, making the most of available resources is paramount. This paper explores the optimization of channel allocation to maximize performance, especially in crucial technologies like WiFi and 5G networks where speed is essential.**

We delve into the use of Combinatorial Multi-Armed Bandits, a method that enables us to learn and make informed decisions about channel allocation. This approach dynamically adapts to changes in the environment while maintaining a balance between exploration and exploitation to achieve optimal throughput.

***Index Terms*—RF, Network Engineering, Channel Allocation, Optimization, Multi-Armed Bandits, WiFi, 5G, Throughput, Combinatorial, Exploration, Exploitation, Non-stationary Environments**

I. Introduction

Choice is a fundamental aspect of life, involving the preference of one entity or group over others and maintaining a cyclic balance of winners and losers. As human beings, we are motivated by factors that offer us the greatest utility, whether in the short term or long term. This inherent drive for utility represents a choice in itself. Similarly, the field of RF and Network Engineering is no exception to this rule. It is characterized by a plethora of problems involving optimization and the utilization of state-of-the-art ML tools to maximize utility or 'reward,' whether in terms of throughput or spectral efficiency.

In this paper, we focus on a critical problem: learning the optimal allocation of a band of orthogonal channels to users to achieve maximum throughput. This problem is both crucial and relatable. For example, in a typical OFDM system implemented in WiFi or 5G, multiple sub-channels (or subcarriers, to be precise) require assignment to different users. If our objective is to ensure fair capacity for all users, one effective strategy could be allocating the best channels to distant users while assigning poorer channels to nearby users to ensure equal capacity distribution.

However, when the throughput or reward from every user-channel pair (referred to as 'matching') is not a deterministic variable but rather an i.i.d. stochastic one, an additional layer of complexity arises. Furthermore, the non-stationary scenario, attributed to a highly mobile environment, adds another layer of complexity to the problem statement. To address this challenge, we employ a framework called Combinatorial Multi-Armed Bandits, a type of Reinforcement Learning problem. This framework allows the algorithm to simultaneously learn and exploit matchings to maximize overall throughput. We implemented [1] and expanded it for non-stationary scenarios, incorporating a design parameter to control the amount of exploration done by the algorithm.

In the subsequent sections, we first discuss related works that inspired our paper, describe the problem rigorously, walk through the solution approach, present simulation results, and conclude with future scope for this paper.

Identify applicable funding agency here. If none, delete this.

II. Related work

There has been extensive research on Multi-Armed Bandits (MAB), exploring various approaches such as Q-learning and policy-driven methods (our focus), including algorithms like greedy epsilon and Thompson sampling. Combinatorial MAB extends the concept by allowing arms to form combinations, creating 'super-arms.' One notable paper by Auer et al. [2] utilized the UCB1 algorithm to address a combinatorial MAB problem, where each arm represents a combination of user-channel matchings.

However, this approach faces scalability challenges. As the number of users (M) and channels (N) increases, the number of arms grows permutationally as P(N,M), leading to a scalability issue. Additionally, it exhibits linear regret growth over time (the difference in throughput between a magical genie which knows the best arm at every time instant with the algorithm's throughput over time).

Therefore, we turned to [1], which introduced the Matching Learning with Polynomial Storage (MLPS) scheme. MLPS leverages the interdependence among multiple arms, a limitation of UCB1. It incorporates the 'Hungarian Algorithm,' capable of computing optimal combinations in $O(N^3)$ (worst-case scenario when M=N), ensuring polynomial time complexity and scalability. MLPS also demonstrated sublinear regret over time. Zheng et al. [5] extended the MLPS framework for Markovian models too.

While the paper assumes a stationary scenario where the mean throughput of matchings remains constant, we tested the algorithm in non-stationary scenarios. Additionally, we introduced a parameter to control exploration levels.

Problem description

The MAB problem is a classic scenario in probability theory and machine learning. It involves a decision maker repeatedly choosing from multiple options, akin to pulling the arm of a slot machine in a casino. The rewards in this context are stochastic, following a certain distribution. A variation of this problem, known as combinatorial MAB, extends the concept by allowing simultaneous plays of multiple arms, forming what's called a super-arm. This paper focuses on leveraging combinatorial MAB. The problem setup is as follows: there are N orthogonal channels and M users $N \geq M$ waiting to access these channels. At each time step n, the task is to determine the combination of user-channel pairings that maximizes the overall reward over time. This involves a trade-off between exploration (to gather diverse data) and exploitation (using the arm with the highest known reward). Formulating this as a combinatorial MAB problem assigns each permutation (k) of pairings to an arm, where $1 < k < P(N, M)$. The stochastic reward for choosing arm k at time n is given by $\hat{\theta}_k(n) = \sum_{(i,j)\in A_k} \theta_{i,j\in A_k}(n)$ with rewards being i.i.d over time.The objective is to design a policy that minimizes the expected regret over time, defined formally as $R^n_\pi(\Theta) = n\theta^* - \mathbb{E}_\pi\left[\sum_{n=1}^{n} \theta_{\pi_n}(n)\right]$, where $\theta^* = \max_k \sum_{(i,j)\in A_k} \theta_{i,j}$ (the expected reward for the magical genie's arm) and $\Theta = \{\theta_{i,j}\}$. Achieving a sub-linear regret over time is the desired outcome.

Solution approach

For the UCB1 approach, we direct you to [2] for the algorithm. Here, we explain the core concept. At every step n, we play an arm k and update the mean reward corresponding to the arm and the number of times that arm has been played (n_k). The strategy of choosing an arm is governed by $\hat{\theta}_k + \sqrt{\frac{2\ln n}{n_k}}$, where the first term aims to play an arm that has the maximum $\hat{\theta}_k$ at that n, and the second term forces exploration. This method, despite sounding really good, is not scalable if M and N increase, as highlighted repeatedly.

The MLPS approach offers a highly scalable solution. For the exact algorithm, we refer you to [1] and our GitHub page (Github-Page) for the Python implementation.

The first core change involves leveraging $\hat{\theta}_{i,j}$ instead of $\hat{\theta}_k$. This modification allows for the exploitation of dependence across different arms, as different arms may share the same user-channel pair matching. Additionally, we update $n_{i,j}$ instead of n_k.

The second key breakthrough entails feeding $\hat{\theta}_{i,j}$ and $n_{i,j}$ into the Hungarian Algorithm (implemented as the Linear-Sum-Assignment function in Python). This algorithm treats each matching as an edge in a bipartite graph and identifies the optimal combination or arm that maximizes the following expression:

$$\sum_{(i,j)\in A_k} \hat{\theta}_{i,j} + M\sqrt{\frac{(M+1)\ln(n)}{\min_{(i,j)\in A_k} n_{i,j}}}$$

The Hungarian algorithm returns the optimal arm in $O(N^3)$ time complexity (assuming $M = N$), which represents a significant improvement over the previous $P(N, M)$ complexity. For a rigorous mathematical analysis of its sub-linear regret over time, we refer you to [1].

Analysis

Figure 1 displays plots we obtained for the naive schemes (greedy epsilon, UCB1 and Thompson sampling), the reason for them being called naive is their non-scalability. So we assumed a very small no of user channel pairs and thus small combinations i.e arms and simulated their average reward over time steps. It can be clearly seen that Thompson sampling performed the best, followed by UCB1 and then by greedy epsilon (which is the simplest of the lot, ($\epsilon = 0.1$, yielded the best result)).

Whereas for the MLPS one, we have assumed a 3x3 user channel scenario, and simulated its average reward over time, we can clearly see, 2 that the MLPS algorithm converges to the optimal combination's value.

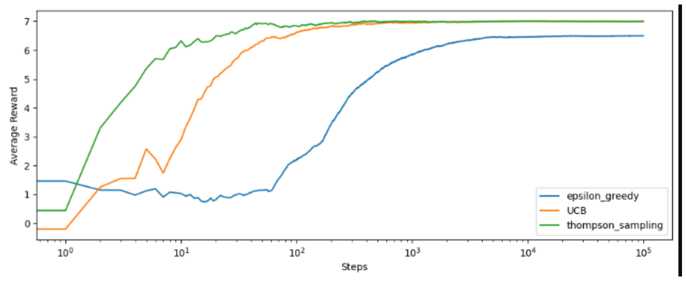

Fig. 1. Performance comparison of naive schemes (greedy epsilon, UCB1, and Thompson sampling) with a small number of user-channel pairs. Thompson sampling outperforms UCB1 and greedy epsilon.

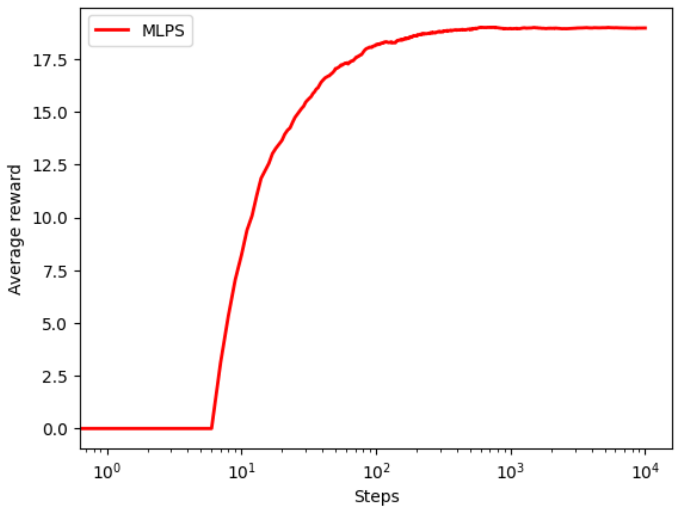

Fig. 2. Convergence of the MLPS algorithm with a 3x3 user-channel scenario under stationary conditions.

Now another interesting scenario is what if the true mean distribution of user channel pairs change? i.e if the channel is

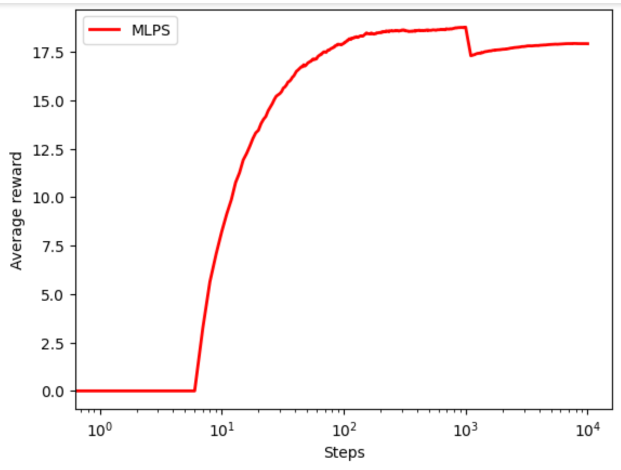

Fig. 3. Adaptation of the MLPS algorithm to non-stationary conditions, where the true mean distribution of user-channel pairs changes over time.

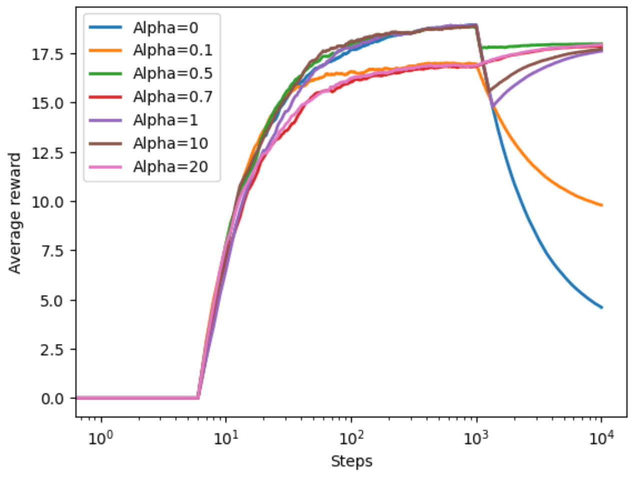

Fig. 4. Impact of the design parameter α on the performance of the MLPS algorithm under non-stationary conditions.

non-stationary? We tried to test how the algorithm responds to this change and we were happy with how it responded, it settled to the new value, albeit, at the cost of some time to respond to that change. Is this time to respond to this change good enough? a good question in its own right for which the next target problem should be on how to accelerate the rate of convergence. This nature of RL is applaudable unlike supervised learning models whose predictions are limited by the input examples they were trained on, thus if the data gets stale, the output is unreliable. We can get around this problem by feeding back Channel State Information (CSI) which is a big overhead on the system and greatly reduces spectral efficiency, whereas in RL you just have to maintain two variables for every user-channel pair, $\hat{\theta_{i,j}}$ and $n_{i,j}$ t current time instant and do away with past values!

We also introduced a design parameter α to the equation, modifying it to $\sum_{(i,j)\in A_k} \hat{\theta}_{i,j} + \alpha M \sqrt{\frac{(M+1)\ln(n)}{\min_{(i,j)\in A_k} n_{i,j}}}$.This addition allowed us to regulate the exploration level. We repeated the simulation under the same non-stationary conditions described earlier, the outcomes are illustrated in Figure 4. Reflecting on these results, we arrived at a simple yet profound realization: ”Excessive focus on either exploitation or exploration can be detrimental; striking a balance is essential in this pursuit”.

CONCLUSION AND FUTURE DIRECTIONS

This paper delves into replicating the Hungarian Algorithm-based Combinatorial MAB framework for a user-channel pair allocation problem, as outlined in [1], while also assessing the algorithm’s performance under non-stationary and controlled exploration conditions. The primary assumption here is the i.i.d. nature of rewards. Moving forward, a potential avenue is to extend this framework to Markovian channels, as discussed in [5]. Additionally, a key focus could be on reducing the algorithm’s convergence time to enhance its applicability in real-time deployments. Another critical aspect to consider is the current centralized coordination (an AP likely) for monitoring mean estimated throughput and related data. Transitioning to a more distributed approach is desirable but comes with its own set of challenges and complexities. We can also extend this approach for many combinatorial MAB based problems and thus has excited the field in our view.

REFERENCES

[1] Y. Gai, B. Krishnamachari and R. Jain, ”Learning Multiuser Channel Allocations in Cognitive Radio Networks: A Combinatorial Multi-Armed Bandit Formulation,” 2010 IEEE Symposium on New Frontiers in Dynamic Spectrum (DySPAN), Singapore, 2010, pp. 1-9, doi: 10.1109/DYSPAN.2010.5457857.
[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” Machine Learning, 47(2-3), 2002.
[3] Available online: https://arxiv.org/abs/2108.07144
[4] Zheng, Z.; Jiang, S.; Feng, R.; Ge, L.; Gu, C. Survey of Reinforcement-Learning-Based MAC Protocols for Wireless Ad Hoc Networks with a MAC Reference Model. Entropy 2023, 25, 101. https://doi.org/10.3390/e25010101
[5] Available online: https://arxiv.org/abs/1012.3005