A Multi-Armed Bandit Model for Non-Stationary Wireless Network Selection

Lluis Martinez
Universitat Politècnica de Catalunya
Barcelona, Spain
lluis.martinez.casanovas@estudiantat.upc.edu

Margarita Cabrera-Bean
Universitat Politècnica de Catalunya
Barcelona, Spain
marga.cabrera@upc.edu

Josep Vidal
Universitat Politècnica de Catalunya
Barcelona, Spain
josep.vidal@upc.edu

Abstract—The amount of wireless networks and technologies such as 5G have been rapidly increasing in the last years. With this growth in number it has become more relevant to be able to select the best network with the goal of maximizing the quality perceived by the user. The Multi-Armed Bandit (MAB) model is a viable approach to describe the problem of the best wireless network selection by a multi-Radio Access Technology (multi-RAT) device. This work proposes a new model that uses real network parameter values that change over time, that is, a non-stationary scenario. While there exist multiple MAB algorithms intended to work in stationary scenarios, this work proposes a new set of algorithms intended to work in non-stationary and more realistic environments. These new algorithms are able to measure the non-stationarity of the environment and adapt accordingly. This is especially relevant when the goal is to select among different network technologies, such as 5G, LTE or WiFi.

I. INTRODUCTION

In recent years the relevance of wireless networks has been drastically increasing. Technologies such as LTE, WiFi or 5G provide access to the internet to more devices everyday. Moreover, the amount of time users dedicate to surfing the internet has also increased. This has been mostly appreciated during the COVID-19 lockdown, where internet traffic hit maximums never observed before. Mobile phones, laptops, IoT devices etc. all use these types of networks to connect to the internet, and the amount of these devices are only expected to increase in the next years.

In order to satisfy these demands, the amount of available networks and network technologies have been increasing. It was only a few years ago that a mobile phone could connect to the internet only using the WiFi from home or the UMTS network. Modern phones can now also connect to the LTE and 5G cellular networks. With the development of new technologies such as 6G the number of networks is only expected to increase. All of these changes contribute to a growth in the number of available networks a multi-Radio Access Technology (multi-RAT) device such as a phone or a laptop can connect to.

While this growth in number allows for users to have more variety and higher quality networks, it comes at a cost: it is harder to choose the best network. This is specially hard in an environment as volatile as the one found in wireless network selection, where the quality of each network may vary because of user movement, number of connected devices etc. Due to this growth in number, the amount of studies focused in solving the best wireless network selection problem has been increasing in recent years. Some works have approached the solution using a Multi-Armed Bandit (MAB).

The MAB problem is one of the simplest settings of reinforcement learning. At each time step $t$, the user must choose an action or arm $i$ from an action set. When an arm $i$ is selected, the environment will return a numerical reward $r$ representing the quality of the selected arm. The goal is to take the adequate decisions in order to maximize the sum of all these rewards. The user does not have any previous information about the environment, so with each decision on the action to take, more knowledge about the environment is gained.

The MAB model has been applied in many fields, e.g. medicine, financial investment etc. One of these fields is the wireless network selection problem. In this application, each arm from the MAB model represents an available network to which a device can connect to. The rewards $r$ that the environment returns usually represent the quality of the chosen network. The MAB model is a good option to represent this problem because we can assume that the selection of an arm will not affect the rewards generated by the environment in future steps.

In [1] a new MAB model is presented: the measure-use Multi-Armed Bandit (muMAB). When a network is selected, there are two possible options: to measure or to use. To use means to connect to the selected network and transfer data. Because data is being transferred, a reward will be generated depending on the measured quality of the selected network. On the other hand, when measuring the device measures the quality of the network without the necessity of connecting to it. Because no data is being transferred, there is no reward. Measuring and using takes a certain amount of time, so when a network is chosen, the device will be locked with that network a number of consecutive steps, $n_m$ when measuring and $n_u$ when using.

In reference [2] a more realistic MAB model for the wireless network selection problem is presented. The proposed model is...
implements a Continuous Time MAB (CT-MAB), where a cost is associated to each arm. In the network selection problem, this cost might symbolize the network access cost, a roaming cost etc. The objective of the algorithms will be not only to maximize the cumulative reward, but also to minimize the cumulative cost.

By looking at multiple implementations of Multi-Armed Bandits for network selection, two major aspects may be observed: the environment is usually considered to be stationary and the reward of each network is normally unrelated to real network parameters. In this paper we focus in designing a more realistic MAB procedure for the best wireless network selection problem. Real network parameter values obtained from multiple LTE datasets have been used to generate the rewards associated to each network. These rewards are non-stationary and represent the quality of the network perceived by the user. Most MAB algorithms are designed for stationary environments, so taking as a baseline the muMAB model presented in [1], we propose a new set of algorithms designed for non-stationary scenarios. These new algorithms are able to measure the non-stationarity of each network and adapt accordingly in order to offer better results and be more flexible.

These algorithms adapted to non-stationarity are presented in section II. In section III the models used for the generation of rewards are explained. In section IV the obtained results are analyzed, and finally in section V the final conclusions are drawn.

II. ALGORITHMS

In this section, the algorithms proposed by this study are presented. Firstly, in subsection II.A the MAB model is introduced. In subsection II.B some classical MAB algorithms are explained. A few variations made to adapt these algorithms to non-stationary environments are discussed in subsection II.C. Finally, in subsection II.D the new adaptive algorithms designed by this study are proposed.

A. The Multi-Armed Bandit Model

In the MAB model, at every time-step \( t_n \) we have to make a choice among \( k \) different actions or arms. Every time you choose an action \( i \) you obtain a numerical reward from a probability density function (usually stationary) related to the chosen action. The goal is to choose the adequate actions in order to maximize the cumulative reward over a certain time period. Usually, minimizing the total regret is used instead of maximizing the cumulative reward. The regret is the expected difference in reward between taking the best possible action and the action that has been chosen. The total regret is defined as the sum of all the regret until the current step. Let us adopt the following notation and definitions:

\[ \mathcal{A} \text{ is the set of possible actions} \]

The reward for action \( i: r \sim f(r|A = i) \) \hspace{1cm} (1)

The mean reward for action \( i: \mu(i) = E\{r|A = i\} \) \hspace{1cm} (2)

The mean reward estimation for action \( i: \hat{\mu}(i) \) \hspace{1cm} (3)

The optimum value: \( V^* = \max_{i \in A} \mu(i) \) \hspace{1cm} (4)

The regret: \( l = E\{V^* - r\} \) \hspace{1cm} (5)

The total regret at step \( n \): \( R_n = \sum_{\tau=1}^{n} V^* - \mu(i_{\tau}) \) \hspace{1cm} (6)

where \( i_{\tau} \) is the arm selected at step \( \tau \).

Like in other reinforcement learning setups, there is a trade-off between exploration and exploitation. When exploiting, the most rewarding action (given the current information) is chosen. On the other hand, when exploring a sub-optimal arm is selected with the purpose of gaining more knowledge about the environment, allowing us to make better decisions in the future. The objective of the MAB problem can also be seen as to calculate the estimate of the mean reward \( \hat{\mu}(i) \) as accurately as possible in order to always choose the arm with the best rewards. With this definition exploiting means to use the arm with the highest estimate, and exploring entails obtaining a more accurate value of the estimates in order to make better decisions in the future.

B. Classical Algorithms

A great number of algorithms have been designed to solve the MAB Problem. One of the simplest ones is the \( \epsilon \)-greedy algorithm. At each step \( n \), the algorithm will exploit with probability \( 1 - \epsilon \), meaning that the arm with the best estimation of the mean \( \hat{\mu}(i) \) will be selected. On the other hand, it will explore with probability \( \epsilon \), as it can be seen in (7). When exploring, a random arm will be selected in an equiprobable manner. The main problem with this algorithm is that it continuously explores because \( \epsilon \) is constant. Usually MAB algorithms perform best when exploring at the beginning, and increase the rate it exploits as there is more information about the environment.

\[ i \leftarrow \begin{cases} \arg \max_i \hat{\mu}(i), & \text{with probability } 1 - \epsilon \\ \text{random arm}, & \text{with probability } \epsilon \end{cases} \] \hspace{1cm} (7)

Another problem with the \( \epsilon \)-greedy algorithm is that when exploring, an arm is selected in an equiprobable manner. This issue is avoided with the Upper-Confidence-Bound action selection I (UCB1) algorithm [4], where the exploration focuses in the most promising arms. This is done by assigning an uncertainty interval to each arm, that will decrease as more information is gathered about that arm, i.e. as the arm is selected more times. The arms will be selected depending on the value of their estimate of the mean \( \hat{\mu} \) and their uncertainty interval, as it can be seen in (8) (where \( N_i(n) \) represents the number of times the arm \( i \) has been selected at step \( n \)):

\[ i \leftarrow \arg \max_i \left( \hat{\mu}(i) + \sqrt{\frac{2 \ln(n)}{N_i(n)}} \right) \] \hspace{1cm} (8)

In [1], where the previously mentioned muMAB model is proposed, two new algorithms designed for this model are also
presented: the measure-use UCB1 (muUCB1) and Measure with Logarithmic Interval (MLI) algorithms. muUCB1 is a variation of the UCB1 algorithm, where action use is selected when the original algorithm would exploit, and action measure is selected when the original algorithm would explore, with the goal of reducing the overhead and the loss of optimality associated to exploration. On the other hand, the MLI algorithm focuses in measuring in periodic intervals, where the interval grows logarithmically.

C. Sliding Window and Discounted variations

Most MAB solutions, such as $\epsilon$-greedy or UCB1, are thought for stationary environments. In this study we assume that the rewards generated by wireless networks are non-stationary as a consequence of the mobility, the traffic generated by users, among other factors. This implies that the classical algorithms must be adapted in order to perform acceptably in these scenarios. In [5] two variations for the UCB1 algorithm are presented. Both propose different ways to limit the memory of the algorithm, so it can be adapted to non-stationarity. The first variation, named Sliding Window UCB1, defines a sliding window of size $\tau$. In this modification, only the last $\tau$ samples are used for the UCB1 algorithm to work, as it is shown in (9) and (10).

$$N_i(n) = \sum_{s=n-\tau+1}^{n} I_{\{t_s=i\}}$$

(9)

$$\hat{\mu}(i) = \frac{1}{N_i(n)} \sum_{s=n-\tau+1}^{n} r_s(i) I_{\{t_s=i\}}$$

(10)

The second variation is the Discounted UCB1 algorithm. The memory of the algorithm is limited by using a discount-factor $\gamma \in (0,1)$. Instead of using a rectangular window, in each step all the previously obtained samples are used to calculate the values by using an exponential window, where the most recent samples have more relevance than the older ones, as it is shown in (11) and (12).

$$N_i(n) = \sum_{s=1}^{n} \gamma^{n-s} I_{\{t_s=i\}}$$

(11)

$$\hat{\mu}(i) = \frac{1}{N_i(n)} \sum_{s=1}^{n} \gamma^{n-s} r_s(i) I_{\{t_s=i\}}$$

(12)

Both variations can also be applied to other algorithms in order to adapt them to non-stationarity, for example muUCB1, MLI or $\epsilon$-greedy. Despite these variations, there is still one major objection: the non-stationarity of each arm might be very different and is unknown previously. This is a problem because the values of $\tau$ and $\gamma$ are defined previously as constants. Ideally, each arm should have its own $\tau$ and $\gamma$ value according to its own non-stationarity. This is especially relevant in the wireless network selection problem, where the quality of each network may vary at very different rates. For example, when a user is moving, the quality perceived of a WiFi network will change much faster than the one from an LTE network.

D. Adaptive algorithms

We propose a new variation, where the non-stationarity of the environment is detected and the memory of the algorithm is changed accordingly. How the non-stationarity is detected is based on a concept presented in [6], where a new version of the LMS algorithm (from the Wiener filter theory) is proposed. The goal of this algorithm is to find the optimum step-size for each filter tap. A unique and different step-size value is used for each filter tap. The algorithm tracks if the current estimate is far or close to converging by measuring the history of signs of the error function $e(n) = d(n) - x(n) * h(n)$, where $d(n)$ is the reference signal, $x(n)$ the input signal and $h(n)$ the filter. When obtaining the same error sign in step $n$ during consecutive steps, it means that the current estimate is very different to the real value. This means that the algorithm is far away from converging at step $n$, so the step-size can increase its value in order to converge faster. On the other hand, when obtaining consecutive alternate signs, it means that the current estimate is very close to the real value. This means that the algorithm is very close to converging in $n$, and that the step-size value must decrease in order to end converging.

The step-size is updated after $e(n)$ has the same sign during $m_1$ consecutive samples or $m_0$ consecutive alternate signs.

The step-size value will increase or decrease by adding or subtracting a previously defined constant $a$ to the step-size value. The maximum step-size, minimum step-size, $m_0$ and $m_1$ values must also be defined previously.

We borrow this concept in order to modify the $\tau$ (sliding window size) and $\gamma$ (discount factor) values depending on the measured non-stationarity of the environment. Each algorithm will monitor the non-stationarity by comparing the obtained reward from arm $i$ to the current estimate $\hat{\mu}(i)$. So the error in this model at step $n$ will be defined as: $e_n(i) = r_n(i) - \hat{\mu}_n(i), r_n(i) \sim f_n(\gamma, i)$. If $e_n(i)$ has the same sign during $m_1$ consecutive samples, it means that the algorithm must explore more, so the $\tau$ or $\gamma$ values must decrease. On the other hand if there is an alternate sign during $m_0$ consecutive samples, the current estimate is very close to the real value, so the memory can be increased in order to take better decisions. This will be done by increasing the values of $\tau$ or $\gamma$.

By applying this concept the adaptive algorithms are obtained, which consist in the Variable Sliding Window and Variable Discounted variations. These variations were applied to the following algorithms: UCB1, muUCB1 and MLI, but they could also be applied to other classical MAB algorithms. For the Variable Sliding Window variations, the sliding window size $\tau$ is shared among all the arms. On the other hand, for the Variable Discounted variations each arm will have a unique discount factor $\gamma$ value. In Algorithm 1 and Algorithm 2, respectively, the pseudo-code for the Variable Sliding Window and Variable Discounted Window variations can be observed. It is important to remark that these pseudo-codes are not an algorithm itself, they are variations that can be made to
In recent years, the number of studies published around how to calculate the QoE has been increasing. Most of these studies use Mean Opinion Scores (MOS) values to represent the quality perceived by the user. These MOS values can be \{1, 2, 3, 4, 5\}, where 1 represents a very bad experience and 5 an excellent one. In order to obtain the MOS scores, the studies assemble a panel of users that will evaluate their experience for different types of traffic and network qualities. A mapping function is generated for each traffic type in order to model the observed results. These mapping functions will be the ones in charge of generating the QoE of each network from the network parameter values.

In our work, the value of the arm rewards generated by the environment are the MOS scores, so the best network for the experience of the user is selected. The mapping functions used are the ones proposed in [7], where 11 traffic types are defined, such as: live video, instant message or meeting video. The mapping functions can be found in (13) and (14), where the \(a, b, c, d, p, q, \alpha, \beta \) and \( \theta \) values will depend on the traffic type used, which is assumed to be known by the user. Three network parameters are used: throughput (\(TH\)), delay (\(DL\)) and loss rate (\(LR\)). \(<\cdot,\cdot>\) is the rounding function.

\[
QoS = a \cdot \exp(b \cdot DL) + \frac{c}{LR + d} + p \cdot \log(TH) + q \tag{13} \\
QoE = <\alpha \cdot \exp(-\beta \cdot QoS) + \theta > \tag{14}
\]

B. Network Parameter Modelling

The network parameter values (\(TH\), \(DL\) and \(LR\)) used in (13) in order to compute the QoE reward are generated in our model by using different realistic random distribution functions. For the throughput (\(TH\)) and delay (\(DL\)), the functions have been designed by observing multiple histograms for real LTE traffic, generated from various LTE datasets. For the packet loss rate (\(LR\)) the histogram found in [8] was used. The same process could be repeated for other technologies, such as WiFi, 5G or even 6G.

To model the throughput an Inverse Gaussian distribution function was proposed, for the delay a Gaussian distribution and for the loss rate the Beta distribution. The values used to generate these functions depend on the quality of the network, but an example used in the simulations is provided below:

- Throughput (\(TH\)): an Inverse Gaussian distribution with \(\mu = 3\) and \(\lambda = 10\). The generated values are in Mbps.
- Delay (\(DL\)): a Gaussian distribution with \(\mu = 0.6511\) and \(\sigma = 0.07\). The generated values are in seconds.
- Loss Rate (\(LR\)): a Beta distribution with \(\alpha = 0.075\) and \(\beta = 10\).

IV. RESULTS

In this section the obtained results are discussed. In section IV.A the different scenarios where the algorithms have been tested are presented. In section IV.B an analysis of the obtained regret is done.
A. Scenarios

In this study, a total of four scenarios have been designed in order to test the algorithms in multiple environments. The first one is stationary and the rest are non-stationary.

The first scenario is the simplest one. There are $k = 5$ available wireless networks. Live Video is the only traffic type considered throughout the entire run. The network parameters are obtained from stationary density functions, and are different for each arm. Since this is a scenario with stationary rewards, the classical algorithms are the ones expected to perform better.

The second scenario also has $k = 5$ available networks, but the traffic type changes every 20000 steps. The used traffic types (in chronological order) are: Instant Message, Voice Call, Online Game, Meeting Video, and Web Browsing. Because the traffic type changes, the constant values used in the QoE mapping functions will change, therefore the generated rewards are non-stationary.

Finally, in the third and fourth scenarios there are $k = 3$ available networks and Live Video is the only traffic type used. The network parameter probability density functions change over time. In the third scenario, the changes are abrupt (every 30000 steps) and in the fourth scenario, they are continuous, therefore rewards are non-stationary in both scenarios.

For all of these scenarios, the simulations were averaged over $N_{\text{Runs}} = 100$ and with the following values: $n_m = 1$, $n_u = 5$. As for the values of the adaptive algorithms, the following have been used: $n_0 = 5$, $n_1 = 10$, $\tau_0 = 5000$, $\tau_{\text{step}} = 500$, $\tau_{\min} = 500$, $\tau_{\max} = 20000$, $\gamma_0 = 0.7$, $\gamma_{\text{step}} = 0.05$, $\gamma_{\min} = 0.05$ and $\gamma_{\max} = 1$.

B. Analysis of Regret

In this section, the evolution of the total regret of the algorithms will be analyzed. The total regret is defined by the difference in cumulative reward between always selecting the best network and the ones selected, so the best results correspond to the lower total regret. The total regret is studied instead of the percentage of succeeding in selecting the best network, because it shows the quality that the user is losing. It is not the same to be wrong with two very similar networks, than being wrong in a situation where there is a clear best network. For simplicity and better understanding of the results, only the UCB1, muUCB1, Sliding Window UCB1, Variable Sliding Window UCB1, Discounted muUCB1 and Variable Discounted muUCB1 algorithms have been considered. It is important to remark that each variation may obtain considerably different results when applied to other MAB algorithms. In figures 1 to 4 the evolution of the accumulated regret for each of these algorithms can be observed.

In figure 1 it can be appreciated that in a stationary scenario the algorithms that perform the best are the original UCB1 and muUCB1. This behavior is expected, since they were designed to work in this kind of stationary environments. It can also be observed that the Sliding Window and Discounted variations obtain a linear regret. This occurs because these algorithms constantly limit the memory and are continuously forced to explore. On the other hand, the adaptive algorithms focus in exploring more at the beginning, and once they detect the environment is stationary, they increase the memory of the algorithm to the maximum. This change in memory stops the linear growth of the regret.

In figures 2, 3 and 4 the results obtained for non-stationary scenarios can be observed. In all these cases, the regret of the classical algorithms does not stop increasing. This is because as the environment changes due to its non-stationarity, the algorithms get stuck in sub-optimal networks. The performance of the Discounted and Sliding Window variations depend on the scenario and algorithm, but despite these differences both algorithms are able to adapt faster to the change of optimal network because they constantly limit the memory. The performance of the adaptive variations also depends on the scenario, but it can be appreciated that the adaptive algorithms take longer to adapt when there is an abrupt change than the Discounted or Sliding Window variations. This is because the adaptive algorithms first need to detect that the environment has changed and then the $\gamma$ or $\tau$ values are modified. Even
though they take longer to detect the change, they still are able to adapt correctly.

In these results the main advantage of the adaptive algorithms can be observed: their versatility. Some algorithms perform remarkably well in a scenario, but in others they may perform poorly. This behavior is not desired in an environment with so many possible and different scenarios as the network selection problem. On the other hand the adaptive algorithms are able to perform well in all the scenarios, independently if they are stationary or not. This can be observed in Table 1, where the percentage of times the best network has been selected by each algorithm may be observed. Note that, once more, the adaptive algorithms present the best behavior in most scenarios, specially the Variable Sliding Window UCB1.

It may also be appreciated that the worse results are obtained in scenario 2. This occurs because it is a scenario with a high number of networks and where the optimal network changes the most frequently.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCB1</td>
<td>99.92%</td>
<td>25.74%</td>
<td>74.60%</td>
<td>72.70%</td>
</tr>
<tr>
<td>muUCB1</td>
<td>99.98%</td>
<td>24.21%</td>
<td>66.97%</td>
<td>51.67%</td>
</tr>
<tr>
<td>Sliding Window UCB1</td>
<td>99.08%</td>
<td>35.80%</td>
<td>97.51%</td>
<td>94.15%</td>
</tr>
<tr>
<td>Discounted muUCB1</td>
<td>94.86%</td>
<td>61.12%</td>
<td>87.49%</td>
<td>80.54%</td>
</tr>
<tr>
<td>Var. Sliding Window UCB1</td>
<td>99.63%</td>
<td>52.30%</td>
<td>97.51%</td>
<td>94.15%</td>
</tr>
<tr>
<td>Var. Discounted muUCB1</td>
<td>98.99%</td>
<td>50.84%</td>
<td>82.23%</td>
<td>74.82%</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

In this study the best wireless network selection problem has been modeled using a Multi-Armed Bandit. Real and non-stationary network parameter values for LTE networks have been used to calculate a reward that represents the quality of experience perceived by the user. It has been demonstrated that classical MAB algorithms do not perform well in non-stationary environments, and that the Sliding Window and Discounted variations are constantly forced to explore. The adaptive algorithms designed by this study have proven that despite they might not be the best algorithms in a specific scenario, they are able to perform globally better in all the proposed scenarios. This flexibility together with a good performance is the key part about an algorithm that will be used in many different network selection scenarios. The results could be further improved by doing more research in highly changing scenarios, such as scenario 2, where smarter tuning of algorithm parameters should provide better results.

REFERENCES