POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Multi-Armed Bandit with Persistent Reward

Relatore: Prof. Francesco Trovò
Correlatore: Dott. Giulia Romano

Tesi di Laurea di:
Andrea Agostini
Matricola 900404

Anno Accademico 2019-2020

Ad Elena e Romano

Abstract

Tasks that requires decision-making under uncertainty are usually linked with the well-known *exploration-exploitation dilemma*, i.e., the search for a balance between exploiting the best possible option given the current knowledge or exploring new alternatives for potential future benefit. The Multi-Armed Bandit (MAB) framework offers an easy to use and theoretically grounded option to solve this dilemma, and has been applied successfully in a wide range of real-world problems. In this setting, an agent must choose, at each time instant, among a finite set of actions, and gains an instantaneous reward drawn from an unknown distribution corresponding to the chosen action. However, a number of real-world scenarios are not cased in the standard MAB formulation, due to the fact that the reward corresponding to an action is gained over multiple time instants following the action. To handle such a scenario, we formalize a variation of the MAB problem, namely MAB with Persistent Reward (MAB-PR), we developed algorithms capable to tackle the addressed problem, and analyzed them from a theoretical perspective. Finally, we performed a thorough experimental analysis that demonstrate the effectiveness of the novel algorithms both on synthetically generated data and real-world data, coming from two applications of recommender systems and dynamic pricing.

Sommario

I problemi di scelta sequenziale in condizioni di incertezza sono spesso legati ad un importante dilemma strategico noto in letteratura con il nome di exploration/exploitation dilemma. Ciò consiste nel trovare il giusto compromesso tra scegliere opzioni promettenti sulla base della conoscenza acquisita sino ad ora (exploitation), o esplorare nuove opzioni alternative allo scopo di ottenere un possibile beneficio futuro (exploration). Il modello Multi-Armed Bandit (MAB) esemplifica e affronta questo dilemma in modo semplice ma con una solida base teorica, infatti, è stato applicato con successo ad una vasta gamma di problemi reali. Nel modello MAB, un agente, ad ogni istante temporale, deve selezionare un'azione da un insieme di alternative valide. La scelta di un'azione fa sì che l'agente guadagni istantaneamente una ricompensa estratta da una distribuzione sconosciuta corrispondente all'azione scelta. Tuttavia, un grande numero di scenari reali non sono rappresentati dal tale modello poiché, nelle loro natura, la ricompensa derivante dall'azione è percepita dall'agente in modo distribuito nel tempo successivo all'azione stessa. Al fine di gestire tali scenari, abbiamo formalizzato una variante del modello MAB chiamata MAB con Persistenza nella Ricompensa (MAB-PR), sviluppato algoritmi specifici per tale modello e li abbiamo analizzati da un punto di vista teorico. Infine, abbiamo condotto un'analisi sperimentale che dimostra l'efficacia dei nuovi algoritmi introdotti sia in configurazioni con dati generati sinteticamente, sia in configurazioni con dati reali derivanti da due applicazioni di sistemi di raccomandazione e dynamic pricing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sequential decision making under uncertainty is one of the most important challenges within the research field of artificial intelligence, as it describes a wide variety of real-world scenarios. In many everyday situations, an *agent*, or a decision maker, has to choose between alternatives to achieve his/her goals. These situations vary from daily routine activities, such as go to work or watch a movie, to complex problems, such as financial investment or web content optimization. In particular, in the former, an agent has to decide how to reach the workplace, and therefore, everyday he/she will choose whether to take the car or go by public transport. Similarly, the latter consists in selecting the best items to display for a given user visit (i.e., page view) from a set of available options. In any case, what makes decision making such a difficult task is the *uncertainty* of the outcome of a decision. Furthermore, the outcome of a decision is usually affected by external factors unknown by the agent. For example, under certain circumstances, driving to work is generally quicker than taking the train, but the traffic conditions could easily invert the situation. In the same way, a particular web content could drastically change its attractiveness due to an abrupt change in social trends. The outcome of a decision, which is typically a *reward*, is revealed to the agent only after the decision has been taken. In the examples mentioned above, the reward can be seen as the time saved on a particular trip from home to work, or as the click-through rate (CTR) obtained by an item shown on a web page. More precisely, consider a sequential decision problem with discrete time and a finite set of actions. At each time step, an agent acting in this environment has to choose an action from the action set, which results in a reward associated with taking that action at that time step. The goal of the agent is to accumulate as much reward as possible over a certain time horizon. To achieve this goal, the agent will learn how to make good actions using the information about rewards acquired from the past actions.

In studying sequential decision making problems it is usually assumed that the reward acquired after an action taken by the agent is a real number provided to the agent after the action has taken place. While this assumption is suitable in many cases, it does not allow the description of a large number of real-world scenarios. For example, consider the wide area of subscription-based services. The subscription business model is a business model in which a customer must pay a recurring price at regular intervals to use a service. When a company running this kind of business, such as a digital streaming platform, an online magazine or a telephone company, enters into a contract with a new customer, it is not able to immediately see the entire reward deriving from that action. This is because the reward is earned over time as a sequence of received payments depending on the period the user remains in the service, which is unknown at the beginning of the contract. Similarly, in a user engagement optimization task, we are interested in selecting the best user interface from a set of alternatives to maximize the time spent by a user in our system. By the nature of the problem, immediately after a user interface is proposed, the reward is unknown. Indeed, the reward is persistently acquired during time according to the engagement of the user with the system. Another example comes from a typical motivating application of this research field which is the optimal design of clinical trials experiments. Consider a realistic experiment with the aim to find the therapy, among a set of alternatives, that maximizes the patient compliance for a particular chronic illness. Patient compliance describes the degree to which a patient correctly follows the therapy, hence after a therapy is assigned we need to take in consideration the level of compliance, which is our reward, obtained at each step of the therapeutic process. We are interested in the type of situations depicted by the examples above in which the reward deriving from an action is spread over the time following the taking of the action.

This thesis studies sequential decision making problems which involve *persistent rewards*, where with the term persistent reward we are referring to a reward that is persistently gained by an agent for a certain timespan after an action has been taken. We design and analyze algorithms tailored for tackle this problem. In particular, we focus our attention on the performance advantage resulting from the exploitation of partial information obtained during the reward acquisition process. We consider a special case of the general problem described above called Multi-Armed Bandit (MAB) problem. We formalize a variation of the classical MAB setting, namely Multi-Armed Bandit with persistent reward (MAB-PR), suitable for handling persistent reward scenarios. The precise problem definition is given in the coming sections.

1.1 Thesis Structure

The Thesis is structured in the following way:

- In Chapter 2 we present the standard MAB problem and the grounding concepts of the thesis by reviewing the relevant literature.
- In Chapter 3 we formally define the MAB-PR problem and the performance measures adopted. We provide some motivating examples and the modelling of two real-world scenarios.
- In Chapter 4 we present in details the novel algorithms developed and their theoretical guarantees.
- In Chapter 5 we perform the experimental analysis of the novel algorithms developed. We include the settings with both synthetically generated and real data.
- In Chapter 6 we draw conclusions. We summarize the main results of our work and we propose some future developments.

Chapter 2

Preliminaries

In this chapter, we present the standard Multi-Armed Bandit problem and the grounding concepts of the thesis by reviewing the relevant literature. In the first section we present the motivating exploration-exploitation dilemma. Then, we give a formal definition of the Multi-Armed Bandit problem and we show the most important policies. Finally, in the last section, we introduce the concept of persistent reward and we make some observations regarding the state-of-the-art.

2.1 The Exploration-Exploitation Dilemma

A popular disease does not have certified treatments yet, to treat patients suffering from it a doctor has to choose among two experimental therapies: a red pill and a blue one. So far, the doctor has treated a total of ten patients, giving the red pill to some of them and the blue pill to the others. Some patients healed and some did not, as shown in figure 2.1.

The red pill seems to perform slightly better than the blue one since it has an efficacy around 43% (three out of seven patients healed) while the other 33% (one out of three patients healed). Which strategy should the doctor follow to treat the next patients? Is it optimal to assign the red pill ignoring the blue one? Or the poor efficacy of the blue pill is due to chance and the doctor has to try it a few more times? How many more times? This illustrates the so-called *exploration-exploitation* dilemma. If the doctor will assign the red pill to the next patient we say that he/she is *exploiting* the knowledge acquired so far, since, apparently, the red pill is performing better. On the other hand, if the doctor will assign the blue pill to the next patient we say that he/she is *exploring* the other therapy to acquire new knowledge, to have a better understanding of the overall situation and possibly to dis-

Patients	1	2	3	4	5	6	7	8	9	10
Red Pill	✗			✗		✓	✓	✓	✗	✗
Blue Pill		✓	✗		✗					

Figure 2.1: Therapy assignment to the first ten patients. Those marked with ✓healed, the others marked with ✗ did not.

cover that the blue pill is better than the red one. The dilemma is that neither exploration nor exploitation can be pursued exclusively to find the optimal action. The doctor must try both the available therapies, managing the trade-off between exploration and exploitation, progressively favoring the one that appear to be best. The exploration–exploitation dilemma has been intensively studied by mathematicians for many decades, yet it remains unresolved (Sutton and Barto, 2018). Finding the right balance between exploration and exploitation is at the heart of the Multi-Armed Bandit problem presented in the following section.

2.2 The Multi-Armed Bandit Problem

Consider the following learning problem. You are faced repeatedly with a choice among K different actions. After each choice you receive a numerical reward generated by the environment you are interacting with. Your objective is to accumulate as much reward as possible over some time period, for example, over 1000 action selections, or *time instants.* In a casino, this kind of sequential decision making problem is obtained when a player is facing many slot machines at once (a "multi-armed bandit"), and must repeatedly choose where to insert the next coin. The name *bandit* refers to the colloquial term for a slot machine ("one-armed bandit" in American slang). There are three fundamental formalizations of the bandit problem depending on the assumed nature of the reward process: stochastic, adversarial (Auer et al., 1995), and Markovian (Anantharam et al., 1987). In this thesis, we focus our attention on the stochastic setting that was first studied by Robbins (1952). In a stochastic *Multi-Armed Bandit* (MAB) problem, at each time instant t, an agent must pull (choose) an arm (action) a_j from an arm set $A = \{1, \dots, K\}$. Pulling an arm a_j at time t produces a reward drawn from an unknown distribution v_j with unknown expectation μ_j. The rewards are defined by random variables $X_{j,t}$ for $1 \leq j \leq K$ and $t \geq 1$, where each j is the index of an arm. Successive pulls of arm a_j yield rewards $X_{j,t_1}, X_{j,t_2}, \dots$ which are independent and identically distributed random variables with unknown expectation $\mu_j \in (0, 1)$. The goal of an agent acting in this environment is to maximize the cumulative reward of its action after n time steps, i.e., $\sum_{t=1}^{n} X_{a_t,t}$, where a_t is the arm pulled at time t. The number of time instants n is called *time horizon.* At each time instant an agent faces the exploration-exploitation dilemma since he/she has to choose between the current best arm (exploitation) or trying other alternatives to have a better knowledge of the environment, hoping to discover a better arm (exploration). To achieve the goal of maximizing the cumulated reward, an agent follows a *policy*, that is, an algorithm that chooses the next arm to pull based on the sequence of past pulls and obtained rewards. In the following subsections we present three of the most important policies for the stochastic Multi-Armed Bandit problem.

UCB1

The policy UCB1 has a frequentist approach and it is based on the principle of *optimism in the face of uncertainty.* This principle states that one should act as if the environment is as nice as plausibly possible. For MAB, this means using the data observed so far to assign each arm a value, called *upper confidence bound* that with high probability it is an overestimate of the unknown mean. An agent that follows the UCB1 policy, at each time instant t, plays the arm a_j that has the largest upper confidence bound $u_j(t)$. The upper confidence bound $u_j(t)$ is computed as the sum of two terms: the empirical mean reward obtained so far $\hat{\mu}_j(t)$ and the uncertainty of the estimate, $c_j(t)$. Essentially, the empirical mean reward $\hat{\mu}_j(t)$ represents the exploitation term of an arm: it says exactly what we obtained so far, when the arm is played it increase or decrease according to the obtained reward. On the other hand, the uncertainty $c_j(t) = \sqrt{\frac{2 \ln t}{T_j(t)}}$, where $T_j(t)$ is the number of times that the arm a_j has been pulled up to time t, gives a contribution in term of exploration since it increases slowly as long as the arm is not pulled, then, as soon as the arm is pulled, it reduces more drastically. The pseudo-code of the UCB1 policy is depicted in Algorithm 1. The first K instants form the initialization phase where each arm is played once.

Auer et al. (2002) provided a theoretical guarantee for the policy UCB1. Given any sub-optimal arm a_j, it has been proved an upper bound on $\mathbb{E}[T_j(n)]$, namely, the expected number of times a policy pulls the sub-optimal arm a_j up to time n.

Theorem 1. *If policy UCB1 is run over a stochastic MAB setting, the expected number of pulls of a sub-optimal arm a_j after n runs is at most:*

$$\mathbb{E}[T_j(n)] \leq \frac{8 \ln n}{\Delta_j^2} + 1 + \frac{\pi^2}{3},$$

Algorithm 1 UCB1

```
 1: for $t \in \{1, \dots, K\}$ do                              ▷ init phase
 2:     pull arm $a_t$
 3: end for
 4: for $t \in \{k+1, \dots, n\}$ do                            ▷ loop phase
 5:     for $j \in \{1, \dots, K\}$ do
 6:         compute $\hat{\mu}_j(t)$
 7:         $c_j(t) \leftarrow \sqrt{\frac{2 \ln t}{T_j(t)}}$
 8:         $u_j(t) \leftarrow \hat{\mu}_j(t) + c_j(t)$
 9:     end for
10:     pull arm $a_i$ such that $i = \arg\max_j u_j(t)$
11: end for
```

where $\Delta_j = \mu^* - \mu_j$ *and* μ^* *is the expected reward of the optimal arm.*

Thompson Sampling

Thompson Sampling (TS) is a policy that adopts a Bayesian approach. It was introduced by Thompson (1933) for the Bernoulli Bandit, and generalized by Agrawal and Goyal (2012) for the general stochastic Bandit. The basic idea is to assume a simple prior distribution on the parameters of the reward distribution of every arm, and at any time instant, play an arm according to its posterior probability of being the best arm. For simplicity, we provide the details of the Thompson Sampling algorithm for the Bernoulli Bandit, i.e. when the rewards are either 0 or 1, and for arm a_j the probability of success (reward $= 1$) is μ_j. We assume Beta distribution as priors because it turns out to be a very convenient choice in this setting. Indeed, Beta distribution is useful for Bernoulli rewards because if the prior is a Beta(α, β) distribution, then after observing a Bernoulli trial, the posterior distribution is Beta($\alpha + 1$, β) or Beta(α, $\beta + 1$), depending on whether the trial resulted in a success or failure, respectively. The TS algorithm initially assumes arm a_j to have prior Beta(1,1) on μ_j, which is equivalent to the uniform distribution. At each time t, having observed $S_j(t)$ successes and $F_j(t)$ failures, the distribution on μ_j are updated as Beta($S_j(t)$,$F_j(t)$). Then, for each arm a_j, it is sampled θ_j from the posterior distributions of μ_j, and the arm having the largest θ_j is pulled. The exploration in Thompson Sampling comes from the randomisation. If the posterior is poorly concentrated, then the fluctuations in the samples are expected to be large and the policy will likely explore. On the other hand, as more data is collected, the posterior

Algorithm 2 `Thompson Sampling`

```
 1: for $j \in \{1, \dots, K\}$ do                                ▷ init phase
 2:     $S_j \leftarrow 1$, $F_j \leftarrow 1$
 3: end for
 4: for $t \in \{1, \dots, n\}$ do                                ▷ loop phase
 5:     for $j \in \{1, \dots, K\}$ do
 6:         sample $\theta_j(t)$ from the Beta($S_j$,$F_j$) distribution
 7:     end for
 8:     pull arm $a_i$ such that $i = \arg\max_j \theta_j(t)$ and observe reward $r_t$
 9:     if $r_t = 1$ then $S_j \leftarrow S_j + 1$ else $F_j \leftarrow F_j + 1$ end if
10: end for
```

concentrates towards the true value and the rate of exploration decreases. The randomness of TS complicates a lot the theoretical analysis and a logarithmic upper bound on $\mathbb{E}[T_j(n)]$, the expected number of times a policy pulls the sub-optimal arm a_j up to time n, has been proved by Agrawal and Goyal (2012) years later the original formulation. The pseudo-code of the TS policy is depicted in Algorithm 2.

Bayes-UCB

Bayes-UCB is a Bayesian algorithm that could be seen as a variant of Thompson Sampling without randomness. Indeed, it eliminates the randomness present in TS by using the quantiles of the posterior distributions on μ_j rather than drawing a sample from them. More precisely, let $Q(1-\alpha, \rho)$, with $0 < \alpha < 1$, be the quantile function associated to the distribution ρ such that: $\mathbb{P}_\rho(X \leq Q(1-\alpha, \rho)) = 1 - \alpha$. At each time instant t, for each arm a_j, the Bayes-UCB algorithm computes an index $q_j(t)$ in the following way:

$$q_j(t) = Q(1 - \tfrac{1}{t(\log n)^c}, \rho_j(t)),$$

where $\rho_j(t)$ is the distribution associated to μ_j at time t, n is the time horizon and c is a parameter. The arm a_j with the largest index $q_j(t)$ is pulled and the distribution on μ_j updated according to the reward received. In the derivation of theoretical guarantees the parameter c is set to $c >= 5$ but it is set $c = 0$ to achieve better experimental results. We can see that Bayes-UCB, at each time $t > 1$, for each arm a_j, selects an upper bound of the mean adopting the principle of optimism in the face of uncertainty. Similar to the policy UCB1 described above, as long as an arm is not pulled its upper bound grows. We provide the pseudo-code of the policy in Algorithm

Algorithm 3 `Bayes-UCB`

```
1: for $j \in \{1, \dots, K\}$ do                                    ▷ init phase
2:     $S_j \leftarrow 1$, $F_j \leftarrow 1$
3: end for
4: for $t \in \{1, \dots, n\}$ do                                    ▷ loop phase
5:     for $j \in \{1, \dots, K\}$ do
6:         $q_j(t) \leftarrow Q(1 - 1/t(\log n)^c, Beta(S_j, F_j))$
7:     end for
8:     pull arm $a_i$ such that $i = \arg\max_j q_j(t)$ and observe reward $r_t$
9:     if $r_t = 1$ then $S_j \leftarrow S_j + 1$ else $F_j \leftarrow F_j + 1$ end if
10: end for
```

3 for the case of Bernoulli Bandit with Beta distribution as prior. Kaufmann et al. (2012) proposed the Bayes-UCB algorithm and provided a logarithmic upper bound on $\mathbb{E}[T_j(n)]$, the expected number of times a policy pulls the sub-optimal arm a_j up to time n.

2.3 Introducing Persistent Rewards

The motivation behind this thesis can be resumed in one straightforward question: what does happen when the reward is not a number immediately available, but it is spread over the time following the pull of an arm? This is the case of many real-world scenarios as described in Chapter 1 or in the Examples 1, 2, 3 provided in Chapter 3. The aim of our thesis is to study the Multi-Armed Bandit problem with *persistent reward*, where with the term persistent reward we are referring to a reward that is persistently gained by an agent for a certain timespan after an arm has been pulled. From a certain perspective, one could consider the reward as a single variable available after the reward acquisition process is terminated. This consideration reduces the persistent reward problem to a delayed reward problem. MAB problem with delayed reward has been extensively studied in the literature. Our major interest is to design methodologies which exploit the partial information during the reward acquisition process without the need to wait its termination to see the reward associated to the pull of an arm. Let us consider a subscription business pricing problem where a seller, that repeatedly signs new contracts, wants to discover which is the monthly price to assign to his/her service to maximize the total reward. At this purpose, we want to develop algorithms that constantly take in consideration the monthly payments made by the user, avoiding to wait the end of the contract to see the total reward acquired thanks to that contract. To the best of our knowledge no previous study addressed the Multi-Armed Bandit problem with persistent reward depicted so far. Standard MAB methodologies can not be directly used in the persistent reward setting. For this reason, we provide the formalization of the Multi-Armed Bandit problem with persistent reward (MAB-PR) and we propose new algorithms specifically designed to tackle this problem.

2.4 Related works

An overwhelming majority of the literature focuses on scenarios that assume instantaneous rewards. Even such a simple model have been successfully applied to a wide range of application scenarios, like advertising allocation problems Gatti et al. (2015); Nuara et al. (2020, 2018), dynamic pricing Trovò et al. (2018); Trovo et al. (2020), security scenarios Bisi et al. (2017), and network routing Li et al. (2016). Nonetheless, in most common real-world problems the feedback is not provided as the learner performs the action, but it is provided with a given delay, e.g., Nuara et al. (2019), or provided over a timespan, as mentioned above.

In the Bayesian MAB setting, the problem of delayed feedback was first proposed by Anderson (1964). Joulani et al. (2013) studied Multi-Armed Bandit problem under delayed feedback and proposed the Delayed-UCB1 algorithm. Delayed-UCB1 is a modification of the standard UCB1 algorithm, depicted in Algorithm 1, adapted to take in input delayed rewards. After this work, the delayed setting has been extended to more complex bandit scenarios, *e.g.*, linear, contextual, non-stationary bandit under delayed feedbacks and/or rewards (Vernade et al., 2020b,a; Gael et al., 2020; Pike-Burke et al., 2018).

Another set of techniques are exploited to solve delayed feedback problems in specific real-world scenarios. For instance, some techniques focus on solving the so called dynamic pricing problem. More specifically, each available price is an arm of a bandit, and the goal is to sell multiple units of the same item maximizing the reward gained in the process. Dynamic pricing has been studied as a bandits problem in the literature for both the stochastic (Babaioff et al., 2015) and the adversarial setting (Amin et al., 2013). These works overcome the classical assumption in the economic theory for which the demand curve is known, using an online approach in which the curve is learned using the interaction with the buyer, e.g., holding a posted price auction, to determine the optimal price for the good. Notice that this approach can be adopted only when there are multiple units on sale. On the other side, when there is a unique item to sell, it is difficult to learn from the experience because of the limited interaction. Typically, in this scenario, a worst-case competitive analysis is performed, as in (Babaioff et al., 2017; Romano et al., 2020).

Another significant example is the Interned advertising problem, in which bandits techniques are widely applied (Badanidiyuru et al., 2013; Nuara et al., 2018; Avadhanula et al., 2021). In this setting, the delayed feedback comes from the fact that clicks can be observed within a few seconds after an ad displays but the corresponding sale, if any, will take hours to happen. This has been solved with a specific delay-feedback algorithm in Vernade et al. (2017).

Notice that, to the best of our knowledge, there is not work specifically addressing the setting in which the reward of a single pull of an arm spreads over multiple time instants.

Chapter 3

Problem Formulation

In this chapter we formally introduce the Multi-Armed Bandit problem with persistent rewards. After defining all the elements and details necessary to depict the interaction between the learning agent and the environment, we specify the performance measures adopted. Finally, we formalize two real-world scenarios which will be deeply analyzed throughout the thesis.

3.1 The MAB-PR Problem

The MAB problem with Persistent Rewards (MAB-PR) that we are going to analyze in this thesis is formalized in this section. Over a finite time horizon, composed of N time instants, at each time instant t, an agent must pull (choose) an arm (action) a_t from an arm set $A = \{1, \dots, K\}$. When we pull an arm a_j at time t, the environment returns a realization of a random variable $R_{j,t}$ and one of a random vector $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \dots, Z_{j,t,Tmax})$. The vector $\boldsymbol{Z_{j,t}}$ represents the persistency of the feedback $R_{j,t}$, meaning that each component $Z_{j,t,m}$ describes which fraction of $R_{j,t}$ the agent will collect at the m-step. At each time instant from the pull of an arm, the learner will collect a reward called *instantaneous reward* defined as follows:

Definition 1 (Instantaneous Reward). *We define the instantaneous reward achieved at time s, consequently to the pull of the arm j at time t, as:*

$$r_{j,t,s} = R_{j,t} Z_{j,t,s-t+1} \ ,$$

where $s \in \{t, \dots, t + T_{\max} - 1\}$ *and* $1 \leq j \leq K$, $1 \leq t \leq N$, $1 \leq m \leq T_{\max}$.

By setting the size of the vector $\boldsymbol{Z_{j,t}}$ to $T_{\max}$, which is a fixed constant, we impose that the lasting of the feedback can be at most $T_{\max}$ steps. We assume $T_{\max}$ to be known to the agent. We do not have any preliminary knowledge regarding the distributions of $R_{j,t}$ and $\boldsymbol{Z_{j,t}}$, we only know that $R_{j,t}$ has support in $[R_{\min}, R_{\max}]$ and $Z_{j,t,m}$ has support in $[0,1]$. Each component $Z_{j,t,m}$ is a random variable with expectation $\gamma_{j,m}$. Each realization of $\boldsymbol{Z_{j,t}}$ is characterized by the number of steps that we have to wait before having a positive component $Z_{j,t,m}$. We call this quantity *delay* and we formalize it in the following way:

Definition 2 (Delay). *We define the delay of a realization $\boldsymbol{Z_{j,t}}$ as:*

$$d_{j,t} = \sum_{m=1}^{T_{\max}} \mathbb{1}_{\{Z_{j,t,m}=0 \ \wedge \ \forall k<m \ \ Z_{j,t,k}=0\}} \ .$$

We are also interested in the position of the last positive component of a realization $\boldsymbol{Z_{j,t}}$. This quantity represents the true number of steps that we need to wait to collect all the instantaneous rewards achievable after the pull of an arm. For this reason, we call it *true length* and we define it as follows:

Definition 3 (True Length). *We define the true length of a realization $\boldsymbol{Z_{j,t}}$ as:*

$$l_{j,t} = \sum_{m=1}^{T_{\max}} \mathbb{1}_{\{\exists k \ \ k\geq m \ \mid \ Z_{j,t,k}>0\}} \ .$$

To better suit a variety of scenarios that require a persistence reward framework, we devise two distinct configurations:

- **General Persistency** We do not assume anything regarding $\boldsymbol{Z_{j,t}}$. This configuration turns to be suitable for scenarios in which the instantaneous reward could be missing at a certain instant, $Z_{j,t,m} = 0$, and then reappear at a later time.

- **Tight Persistency** We impose that, giving a realization of a persistency vector, every non-zero component must be adjacent. More formally, we say that we are in *Tight Persistency* configuration if for each realization $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \dots, Z_{j,t,Tmax})$ the following condition holds:

$$\sum_{m=1}^{T_{\max}} \mathbb{1}_{\{Z_{j,t,m}>0\}} = l_{j,t} - d_{j,t} \ .$$

We now present two examples derived from practical cases with the aim to highlight the needs that motivate the two configurations mentioned above.

Example 1 (Pricing of a magazine subscription). *We are the seller of an online magazine that works via subscription. To have access to our service, a* *new user can stipulate a contract with a fixed duration and monthly fees. We allow to suspend and restart the service at every moment during the contract, simply stopping or making the monthly payments. Intuitively, we think that high prices discourage a continuous usage of the service while low prices could lead to stable subscriptions but with the risk of generating unsatisfactory profit. We are facing the problem of finding the best monthly price to assign at the service to maximize the revenue. This scenario can be directly modeled as a MAB persistent problem in* ***General Persistency****. Each arm can be assigned to a specific fee designed as a valid option. When the agent pulls an arm the extracted feedback* $R_{j,t}$ *will be the price related to that arm. The persistency vector* $\boldsymbol{Z_{j,t}}$*, on the other side, will capture the adherence of the user to the service and will have a size of* $T_{\max}$ *equal to the number of months of the contract. Every component of the vector will be a Bernoulli variable that takes the value 1 if the user has made the payment for a certain month or the value 0 in the opposite case.*

Example 2 (Medical Trial)**.** *We want to conduct an ethical clinical trial to define which is the best medical treatment for a specific chronic illness. Consider the case where we have two options available, a red pill and a blue one, hence we model them as two arms. Every day the agent must choose which one of the two therapies administer to a new patient on the basis of previous observations. Differently from prior MAB application for this task, in this setting we want to consider also the life quality of a patient in addition to his/her lifespan. For this reason after the treatment administration, a patient is tested every day and an index of his/her health status is computed. We assume that this index is ranging from 0 to 1, where 1 represents a perfect state of health and 0 means that the patient is dead. This scenario could be easily addressed as a MAB persistent problem in* ***Tight Persistency*** *configuration with delay steps equal to zero for each realization of the persistency vector. As a matter of fact, we can set* $T_{\max}$ *at the maximum lifespan possible after the diagnosis of the considered illness, and we can model every component of the vector* $\boldsymbol{Z_{j,t}}$ *as the health status index. For this scenario,* $R_{j,t}$ *could be fixed to a constant equal for each arm, letting the role of capturing the reward only to the persistency vector* $\boldsymbol{Z_{j,t}}$*. The Tight Persistency condition holds, as a matter of fact, it does not make sense to have a positive index health status after the death of the patient.*

The nature of the presented problem leads us to introduce two definitions of reward achievable pulling an arm. In a straightforward manner, we call *Pull Reward* the the sum of the instantaneous rewards gained thanks to the pull. In both Example 1 and Example 2, the goal of the learning agent was to find the arm able to maximize this quantity. However, in some scenarios could be reasonable to take in consideration also the time needed to collect all the instantaneous rewards of a pull. In particular, we call *Normalized Pull Reward* the sum of instantaneous rewards divided by the true length of the persistency vector. This measure is particularly relevant when we consider case studies in which we want to allocate resources and we must take into consideration possible vacant periods, as outlined in Example 3. Formal definitions of rewards are provided below.

Definition 4 (Pull Reward)**.** *We define the pull reward achieved pulling the arm j at time t as:*

$$X_{j,t} = \sum_{s=t}^{t+T_{\max}-1} r_{j,t,s} \ .$$

Definition 5 (Normalized Pull Reward)**.** *We define the normalized pull reward achieved pulling the arm j at time t as:*

$$Y_{j,t} = \frac{\sum_{s=t}^{t+T_{\max}-1} r_{j,t,s}}{l_{j,t}} \ .$$

We now give an example of a scenario in which we are interested in maximizing the Normalized Pull Reward.

Example 3 (Pricing of a Cloud Computing Service)**.** *A cloud computing company has a new set of servers at its disposal and is facing the problem of deciding the daily price to rent a server. Once a specific price has been chosen, the company will disclose its offer online and, later, will enter into a contract of a fixed duration with the purchaser. Each day of the contract, the user will pay a fixed cost concerning the rent, in addition to a variable cost related to the resources usage. The company assumes that by publishing an offer with an high price it will take a long time to find a buyer, on the contrary, with a very low price it will immediately be able to rent it but with little profit. In this scenario, we see how the unused server time affects the income, therefore, not only the accumulated reward must be taken into account but also the time necessary to find a buyer. The problem is well modeled in* ***Tight Persistency****. Each arm a_j is associated to a deterministic daily price $R_{j,t}$ and the delay $d_{j,t}$ of each persistency vector $\boldsymbol{Z_{j,t}}$ will represent the days between the publication of the offer and the stipulation of the contract. Hence, $R_{j,t}$ can be seen as the price of one day of full use of the service, and finally each positive component $Z_{j,t,m}$ will indicate the fraction of $R_{j,t}$ to be daily paid by the user. We are interested in finding the arm that maximizes the* ***Normalized Pull Reward****, taking into account also the penalty imposed by the vacant periods.*

Table 3.1: Configurations-Rewards scenarios. The combination General-Persistency/Normalized Pull Reward leads to cases that are not of practical interest.

	Pull Reward	Normalized Pull Reward
General Persistency	Spotify Scenario Example 1	
Tight Persistency	Example 2	Rent Scenario Example 3

Successive plays of arm a_j yield pull rewards $X_{j,t_1}, X_{j,t_2}, \ldots$ which are random variables independent and identically distributed according to an unknown distribution with unknown expectation μ_j. In the same way, we assume that the normalized pull rewards $Y_{j,t_1}, Y_{j,t_2}, \ldots$ are random variables i.i.d. with unknown expectation η_j. For the sake of simplicity, we will refer to a generic reward of arm a_j at time t as $X_{j,t}$, to adopt the same notation of the standard MAB literature. However, the definitions stated below will apply evenly to the *Pull reward* and the *Normalized pull Reward*, unless otherwise specified.

3.1.1 Performance measures

The goal of a learning agent is to maximize its accumulated reward, the pulling strategy adopted to accomplish this task is referred as *policy*. To measure the performance of a policy, we compare its behaviour with the one of a fictitious algorithm, called *Oracle*, which for any horizon of n time steps constantly plays the optimal arm. For this purpose, we introduce the concept of *Regret*.

Definition 6 (Regret). *The Regret of a policy accumulated after n plays is defined as:*

$$\mathfrak{r}_n = \max_{j=\{1,\ldots,k\}} \sum_{t=1}^{n} X_{j,t} - \sum_{t=1}^{n} X_{a_t,t} \ ,$$

where a_t is the arm played by the learner at time t and the first term $\max_{j=\{1,\ldots,k\}} \sum_{t=1}^{n} X_{j,t}$ represents the reward accumulated by the Oracle up to time n.

Since both the rewards and the player's actions are stochastic, we introduce a form of average regret called *pseudo-regret*.

Definition 7 (Pseudo-Regret)**.** *The Pseudo-Regret of a policy accumulated after n plays is defined as:*

$$\mathfrak{R}_n = n\mu^* - \sum_{t=1}^{n} \mu_{a_t} \ ,$$

where $\mu^* = \max_{j=\{1,\dots,k\}} \mu_j$ *is the expected reward of the optimal arm and* μ_{a_t} *is the expected reward of the arm played at time t.*

For clarity, we explicate the definition of *Normalized-Pseudo Regret* in the following way:

Definition 8 (Normalized Pseudo-Regret)**.** *The Normalized Pseudo-Regret of a policy accumulated after n plays is defined as:*

$$\mathfrak{N}\mathfrak{R}_n = n\eta^* - \sum_{t=1}^{n} \eta_{a_t} \ ,$$

where $\eta^* = \max_{j=\{1,\dots,k\}} \eta_j$ *is the expected normalized pull reward of the optimal arm and* η_{a_t} *is the expected normalized pull reward of the arm played at time t.*

The Pseudo-Regret form is more suitable for the purpose of our analysis respect to the Regret. Therefore, in what follows of the thesis we will evaluate the algorithms in terms of Pseudo-Regret and Normalized Pseudo-regret in the case we are considering Pull Reward or Normalized Pull Reward respectively. In the next chapters, in order to simplify the notation, we will omit the term pseudo.

3.2 Modeling of real-world scenarios

3.2.1 The Spotify Playlist Problem

Recommender systems represent user preferences for the purpose of suggesting items to purchase or examine. They have become fundamental applications in electronic commerce and information access, providing suggestions that effectively prune large information spaces so that users are directed toward those items that best meet their needs and preferences (Burke, 2002). One of the most common problems in recommender systems is the *cold-start* problem. The cold-start problem typically happens when the system does not have any form of data on new users and on new items (Bobadilla Sancho et al., 2012). There are two distinct categories of cold start: the item cold start and the user cold start. The new user case refers to when a new user enrolls in the system and for a certain period of time the recommender has to provide recommendations without relying on the user's past interactions, since none has occurred yet (Moghaddam and Elahi, 2019). This problem is particularly important when the recommender is part of the service offered to users, since a user who is faced with recommendations of poor quality might soon decide to stop using the system before providing enough interaction to allow the recommender to understand his/her interests. Spotify is a digital music service which has a great interest in recommender systems. In 2018 it was the organizer of the ACM Conference on recommender systems. We model the problem of recommending a playlist to a new Spotify user as a MAB-PR problem, proposing a new approach to mitigate the cold-start problem.

Formulation

When a new user accesses the system, a playlist is proposed. Subsequently, The user will start the reproduction of the playlist and for each song, in any moment, he/she could decide to skip to the next song till the end of the playlist. We are interested in finding the playlist that maximizes the overall listening time. We model the problem as a MAB-PR problem with the specifics reported below.

- Each playlist is a set of 20 songs.
- Each arm a_j is associated to a playlist.
- The feedback $R_{j,t}$ is fixed to a constant equal for each arm, since we are only interested in finding the playlist with the best persistency.
- Based on an official dataset released from Spotify, it is known if a user listened a song for the first 25%, 50%, 75% or 100% of its duration. This granularity lead us to model each song with four adjacent components of the persistency vector $\boldsymbol{Z_{j,t}}$, where each component $Z_{j,t,m}$ represents a quarter of a song. Each component is a *Bernoulli variable* that takes the value of 1 if the user has listened the song up to that quarter or 0 in the opposite case. The persistency vector will capture the adherence of the user during the playlist, hence its size $T_{\max}$ will be equal to the number of songs of a playlist times the granularity, in this case $T_{\max} = 4 \times 20 = 80$. An example of a realization of the persistency vector is provided in figure 3.1.
- We want to find the playlist that give us the highest listening time, hence we want to maximize the *pull reward.*

1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0
song 1				song 2				song 3				song 4				song 5			

Figure 3.1: Example of realization of the persistency vector in the Spotify Playlist Problem. The persistency vector is truncated at the fifth song for visualization purposes. Songs 1 *and* 3 *were listened completely, while song 2 was listened to the* 50% *of its duration. Song* 4 *and Song* 5 *were skipped entirely.*

3.2.2 The Rental Pricing Problem

A company owns a large number of rooms with the same characteristics. These rooms are rented with fixed duration contracts and monthly fees. Once the contract has been signed, the tenant can choose to stay until the expiration date or to cancel, ending his stay before the expiry date. When the company publishes the rental announcement, it is aware that by setting a high fee for the room, it can have a long vacancy period. Furthermore, a high fee could discourage the tenant from staying until the end of the contract. At the same time, by setting a fee too low, the company could make unsatisfactory profits. The problem of choosing the best fee is modeled as a MAB-PR problem with the specifics reported below.

- Each arm a_j is associated to a specific fee designed as a valid option.
- The feedback $R_{j,t}$ is set equal to the fee of the arm a_j. It is deterministic, meaning that $R_{j,t} = R_j \ \forall t$.
- The persistency vector $\boldsymbol{Z_{j,t}}$ represents the period of time ranging from the publication of the rental announcement to the deadline of the contract. Each component of the vector $Z_{j,t,m}$ is a *Bernoulli variable* that represents a month. It will take the value of 1 if the tenant has made the payment for a certain month or 0 in the opposite case.
- Each realization of the persistency vector $\boldsymbol{Z_{j,t}}$ is characterized by a delay $d_{j,t}$ equal to the number of vacant months. The vacant months represent the period between the publication of the announcement and the signing of a new contract.
- We define the maximum delay as d_{max} and the maximum duration of a contract as c_{max}. The size of the persistency vector $\boldsymbol{Z_{j,t}}$ is $T_{\max} = d_{max} + c_{max}$.
- We want to find the fee that allows us to maximize our profit keeping in consideration the vacant periods, where we don't receive payments. For this purpose, we want to maximize the *normalized pull reward.*

0	0	1	1	1	1	1	1	0	0

Figure 3.2: Example of realization of the persistency vector in the Rental Pricing Problem. In this example $T_{\max} = 10$, the delay $d = 2$, the true length $l = 8$. The contract last for six months, since we have six ones. The last two zeros are not relevant, a possible interpretation could be that $c_{max} = 6$ and $d_{max} = 4$.

- We assume that once a contract is canceled, it is not possible to reenter. This implies that we are in *tight persistency*. A realization of the persistency vector $\boldsymbol{Z_{j,t}}$ will be a sequence of zeros representing the delay $d_{j,t}$, followed by a sequence of ones representing the actual contract, followed by a sequence of zeros with length $= T_{\max} - l_{j,t}$. An example is provided in figure 3.2.

Chapter 4

Novel Algorithms

In this chapter we present in details the novel algorithms and their theoretical guarantees. In the first section, we give an overview of the MAB-PR algorithms. Then, we describe the frequentist and the Bayesian policies developed.

4.1 Introduction to the Novel Algorithms

As outlined in White (2012), Multi-armed Bandit algorithms have to actively select which data you should acquire and analyze that data in real-time. Indeed, bandit algorithms exemplify two types of learning: *active learning*, which refers to algorithms that actively select which data they should receive; and *online learning*, which refers to algorithms that analyze data in real-time and provide results on the fly. This means that to evaluate algorithms we need to design a simulation environment where we can mimic real-scenarios dynamics. Informally, we call *bucket* a realization $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \ldots, Z_{j,t,Tmax})$ of the persistency vector and we call *bin* the m-th element $Z_{j,m,t}$ of the persistency vector $\boldsymbol{Z_{j,t}}$. During the simulation, each bucket will be parsed according to the experiment time t, meaning that the learner can only visit the bin of the bucket containing the information gathered in the past. As stated in Chapter 3, when at time t an arm a_j is played, the environment generates a bucket $\boldsymbol{Z_{j,t}}$ and a feedback $R_{j,t}$ that are collected by the learner. Changing the point of view, we can say that, during the experiment, each arm a_j collects a sequence of pulls, characterized by the extracted feedback-bucket pairs $(R_{j,t}, \boldsymbol{Z_{j,t}})$. The algorithms described below have a structural difference from the standard ones designed for traditional Multi-armed Bandit. Indeed, in the presented framework, more than one arm can have a set of non-terminated buckets (not totally parsed yet)

Table 4.1: For each policy, we depict the operating characteristics: the family; the reward maximized; the exploitation of the partial information deriving from buckets not fully parsed. Note that the policies that do not exploit partial information are considered as baselines.

	Family		**Reward**		**Partial Information Exploit**	
Policy	*Frequentist*	*Bayesian*	*P.R.*	*N.P.R.*	*Yes*	*No*
PR-T-UCB-P	x		x			x
PR-T-UCB-NP	x			x		x
PR-BW-UCB-P	x		x		x	
PR-NT-UCB-P	x		x		x	
PR-T-TS-P		x	x			x
PR-T-TS-NP		x		x		x
PR-BW-BayesUCB-P		x	x		x	
PR-BW-BayesUCB-NP		x		x	x	

simultaneously. This parallelism implies that at each time instant we need to update the terms of every arms and not only of the last one played. Some of the algorithms presented will work without distinction with both pull reward and normalized pull reward. For the sake of clarity, we define a generic reward $W_{j,t}$ with unknown expectation ω_j that acts as a proxy variable. Depending on the setting addressed, $W_{j,t}$ will represent the pull reward $X_{j,t}$ or the normalized pull reward $Y_{j,t}$. The table 4.1 summarizes the algorithms presented and their operating characteristics.

4.1.1 Farsighted and Myopic configuration

We can consider a bucket $\boldsymbol{Z_{j,t}}$ terminated (fully parsed) according to two different criteria: (i) $T_{\max}$ time instants have passed since t; (ii) $l_{j,t}$ time instants have passed since t. We say that the algorithm is in *myopic* and *farsighted* configuration in the case we are adopting (i) or (ii) respectively. More formally:

- In *myopic* configuration, at time t, we consider a bucket $\boldsymbol{Z_{j,s}}$ terminated if $t \geq s + T_{\max}$;

- In *farsighted* configuration, at time t, we consider a bucket $\boldsymbol{Z_{j,s}}$ terminated if $t \geq s + l_{j,s}$, where $l_{j,s}$ is the true length of the bucket $\boldsymbol{Z_{j,s}}$.

Algorithm 4 `Frequentist Policy`

```
Require: arm set $A = \{a_1, a_2, \dots, a_k\}$, time horizon $N$, update function $U$
 1: function POLICY($A$,$N$,$U$)
 2:     for $t \in \{1, \dots, k\}$ do                                ▷ init phase
 3:         pull arm $a_t$
 4:         call $U$
 5:     end for
 6:     for $t \in \{k+1, \dots, N\}$ do                              ▷ loop phase
 7:         pull arm $a_i$ such that $i = \arg\max_j u_j$
 8:         call $U$
 9:     end for
10: end function
```

Table 4.2: For each policy, we report the order of the bound on the expected regret when the number of pulls t → ∞ . The policies are assumed in myopic configuration.

Policy	**Regret Bound Order**
PR-T-UCB-P	$O(T_{\max}^2 \ln(t))$
PR-BW-UCB-P	$O(T_{\max}^2 \ln(t))$
PR-NT-UCB-P	$O(T_{\max} \ln(t))$

4.2 Frequentist Algorithms

To facilitate the understanding of the code, the algorithms described in this section will be decoupled into two functions: the *Policy* and the *Update*. The policy function will describe the interaction of the learner with the environment and will require an update function passed as an argument. The update function will be responsible to update the knowledge of the learner coming from new data and compute the indices needed by the policy to take decisions, concretizing the overall strategy. The pseudo-code of a generic frequentist policy is depicted in Algorithm 4. The first k instants form the initialization phase, during which each arm is chosen once. After the k^{th} time instant the loop phase begins. Here, at time t, the agent plays the arm a_j having the largest index u_j, the upper confidence bound of the arm a_j. After the play of an arm, the update function U occurs. Algorithm 4 requires in input an arm set A, the time horizon N, and an update function U. Below we propose three update functions that can be passed in input to the generic policy depicted in Algorithm 4, concretizing the following policies: PR-T-UCB, PR-BW-UCB-P, PR-NT-UCB-P. The theoretical guarantees obtained are summarized in Table 4.2.

4.2.1 PR-T-UCB (Frequentist Baseline Algorithm)

PR-T-UCB (Persistent Reward - Terminated Buckets - UCB) is an algorithm that extends the idea of the well known UCB1 algorithm in the case of persistent reward. The reward is considered as a unique variable available after the termination of the bucket. The pseudo-code is provided in Algorithm 5. This algorithm takes inspiration from the work of Joulani et al. (2013) on delayed feedback. Indeed, evaluating the reward only after termination of the bucket reduces the persistent feedback problem to a delayed feedback problem. For this reason we consider it as a baseline algorithm with which the other proposed strategies are compared. It works for both the cases when we want to maximize the pull reward ($\omega_j = \mu_j$) or when we want to maximize the normalized pull reward ($\omega_j = \eta_j$). In the former we will extend the name of the algorithm to PR-T-UCB-P, in the latter to PR-T-UCB-NP. For each arm a_j, the update function will detect the terminated buckets according to the configuration (farsighted or myopic) adopted. At each time t, for each arm a_j, the algorithm computes: the empirical mean reward $\hat{\omega}_j(t)$ obtained from the terminated buckets of the arm a_j, ignoring the non-terminated buckets (line 3); the exploration term $c_j(t)$ (line 4). Let $\epsilon(t) = \sqrt{\frac{2 \ln t}{B_j(t)}}$, the exploration term $c_j(t)$ is computed in the following way.

- if we are maximizing the *Pull Reward*:

$$c_j(t) = R_{\max} T_{\max} \epsilon(t) = R_{\max} T_{\max} \sqrt{\frac{2 \ln t}{B_j(t)}} \ ;$$

- if we are maximizing the *Normalized Pull Reward*:

$$c_j(t) = R_{\max} \epsilon(t) = R_{\max} \sqrt{\frac{2 \ln t}{B_j(t)}} \ ;$$

where with $B_j(t)$ we indicate the number of terminated buckets of the arm a_j at time t. The term $\epsilon(t)$ is the exploration term when the reward has support in [0,1]. With $c_j(t)$ we indicate the exploration term considering the support of the reward in our case.

Finally, the index $u_j(t)$ is computed by summing the current empirical mean reward $\hat{\omega}_j(t)$ and exploration term $c_j(t)$ (line 5). In case an arm a_j does not have any terminated bucket, its index $u_j(t)$ is set to infinite. If we are dealing with settings where the feedback $R_{j,t}$ is deterministic ($R_{j,t} = R_j$ for each arm a_j, for each t), we can replace $R_{\max}$ with R_j in the computation of the exploration term $c_j(t)$. This let us to have a smaller or equal exploration term $c_j(t)$, and possibly, an improvement on the performances.

Algorithm 5 `PR-T-UCB (Frequentist Baseline)`

```
Require: bucket size $T_{\max}$, maximum feedback $R_{\max}$
1: function UPDATE($T_{\max}$,$R_{\max}$)
2:     for each arm $a_j \in A$ do
3:         compute empirical mean reward $\hat{\omega}_j$ from the terminated buckets
4:         compute $c_j$
5:         $u_j \leftarrow \hat{\omega}_j + c_j$
6:     end for
7: end function
```

PR-T-UCB Theoretical Analysis

Joulani et al. (2013) provide the Delayed-UCB1 algorithm, a modification of the UCB algorithm for MAB problem with delayed feedback. Delayed-UCB1 algorithm enjoys the same regret guarantees compared to its non-delayed version, up to an additive penalty depending on the delays. We can rewrite their result, adapting it to our setting with deterministic R_j, in the following way:

Theorem 2. *For $K \geq 1$, if policy PR-T-UCB-P is run in myopic configuration on K arms having deterministic feedback $R_1, \ldots, R_K$, then the expected regret after any number of pulls t is at most:*

$$\mathbb{E}[\mathfrak{R}_t] \leq \sum_{i:\Delta_i>0} \left[\frac{8R_i^2(T_{\max} - T_{\min})^2 \ln t}{\Delta_i} + \left(1 + \frac{\pi^2}{3}\right)\Delta_i \right] + \sum_{i=1}^{K} \Delta_i \mathbb{E}[G^*_{i,n}],$$

where $G^*_{i,n}$ is the maximum number of missing feedbacks from arm i during the first n time steps and $\Delta_i = \mu^* - \mu_i$.

Notice that in the Persistent Reward setting $G^*_{i,n} < T_{\max}$ and we consider $T_{\min} = 0$, therefore, in the worst case, the regret can be bounded by

$$\mathbb{E}[\mathfrak{R}_t] \leq \sum_{i:\Delta_i>0} \left[\frac{8R_i^2 T_{\max}^2 \ln t}{\Delta_i} + \left(1 + \frac{\pi^2}{3}\right)\Delta_i \right] + T_{\max} \sum_{i=1}^{K} \Delta_i.$$

4.2.2 PR-BW-UCB-P

The algorithm PR-BW-UCB-P (Persistent - Bin-Wise - UCB - Pull Reward) is tailored for scenarios where we want to maximize *Pull Reward* and we have a deterministic feedback R_j for each arm a_j. Our goal is to exploit also the information given by the non-terminated buckets using bin-wise upper confidence bounds. The idea behind Algorithm 6 is to estimate the average

Algorithm 6 PR-BW-UCB-P

```
Require: bucket size $T_{\max}$
1: function UPDATE($T_{\max}$)
2:     for each arm $a_j \in A$ do
3:         for $m \in \{1, \dots, T_{\max}\}$ do
4:             compute $\overline{z}_{j,m}$ from the available buckets
5:             $c_{j,m} \leftarrow \sqrt{2 \ln t / |V_{j,m}(t)|}$
6:         end for
7:         $u_j \leftarrow R_j \sum_{m=1}^{T_{\max}} \min(1, \overline{z}_{j,m} + c_{j,m})$
8:     end for
9: end function
```

bin value $\overline{z}_{j,m}(t)$ given by the m-th bins of the buckets gained by pulling the arm a_j. To compute $\overline{z}_{j,m}(t)$, the algorithm considers only the buckets gained by a_j that have been already parsed at position m (informally called "available buckets" at line 4). We indicate with $V_{j,m}(t)$ the set of buckets $\boldsymbol{Z_{j,s}}$ that at time t have been already parsed at position m. Depending on the configuration adopted, we say that:

- In *myopic* configuration $\boldsymbol{Z_{j,s}} \in V_{j,m}(t)\ if\ t \geq s + m - 1$;

- In *farsighted* configuration $\boldsymbol{Z_{j,s}} \in V_{j,m}(t)\ if\ t \geq s+m-1 \vee\ t \geq s+l_{j,s}$. As a matter of facts, the farsighted configuration allow us to consider a bucket $\boldsymbol{Z_{j,t}}$ fully parsed after that it has been parsed at position true length $l_{j,t}$.

At each time t, for each arm a_j, for each position m, the algorithm computes $\overline{z}_{j,m}(t)$ and $c_{j,m}(t)$ (line 4 and line 5 respectively). $\overline{z}_{j,m}(t)$ is computed in the following way:

$$\overline{z}_{j,m}(t) = \frac{1}{|V_{j,m}(t)|} \sum_{\boldsymbol{Z_{j,s}} \in V_{j,m}(t)} Z_{j,s,m} \ .$$

Finally, for each arm a_j, the upper confidence bound $u_j(t)$ is calculated by multiplying the feedback R_j with the sum, over m, of the average bin value $\overline{z}_{j,m}(t)$ and the exploration term $c_{j,m}(t)$ (line 7). Each element of the summation is upper bounded to 1. This is a help we give to the learner to speed up the learning process. Indeed, this is known a priori considering that each bin has support in [0,1].

PR-BW-UCB-P Theoretical Analysis

Theorem 3. *For $K \geq 1$, if policy PR-BW-UCB-P is run in myopic configuration on K arms having deterministic feedback $R_1, \ldots, R_K$, then the expected regret after any number of pulls t is at most:*

$$\mathbb{E}[\mathfrak{R}_t] \leq \sum_{i:\tilde{\Delta}_i > 0} \left(\frac{8 R_i^2 T_{\max}^2 \ln t}{\tilde{\Delta}_i} \right) + T_{\max} \left(1 + \frac{\pi^2}{3} \right) \sum_{i=1}^{K} \tilde{\Delta}_i$$

where $\tilde{\Delta}_i = R^* \sum_{m=1}^{T_{max}} \gamma_m^* - R_i \sum_{m=1}^{T_{max}} \gamma_{i,m}$.

Proof of Theorem 3 We summarize the notation as follows:

- t is the current time;
- $\bar{z}_{s_i,m,t} = \bar{z}_{i,m}(t)$ is the average bin-value given by the m-th bins gained by the arm i at time t, after that it has been pulled s_i times;
- $c_{s_i,m,t}$ is an exploration term associated to $\bar{z}_{s_i,m,t}$;
- $n_{s_i,m,t} = |V_{i,m}(t)|$ is the number of buckets whose m-th element is already parsed at time t;
- $\gamma_{i,m}$ is the expectation of $z_{i,m}$;
- $\mathfrak{T}_i(n)$ is the number of pulls of the arm a_i at time n.

Starting from the proof of UCB1 provided by of Auer et al. (2002) we can rewrite:

$$\mathfrak{T}_i(n) \leq \ell + \sum_{t=1}^{\infty} \sum_{s=\ell}^{t-1} \sum_{s_i=\ell}^{t-1} \left\{ R^* \left(\sum_{m=1}^{T_{max}} \bar{z}^*_{s,m,t} + c_{s,m,t} \right) \leq R_i \left(\sum_{m=1}^{T_{max}} \bar{z}_{s_i,m,t} + c_{s_i,m,t} \right) \right\}. \tag{4.1}$$

Note that $R^* \big(\sum_{m=1}^{T_{max}} \bar{z}^*_{s,m,t} + c_{s,m,t} \big) \leq R_i \big(\sum_{m=1}^{T_{max}} \bar{z}^*_{s_i,m,t} + c_{s_i,m,t} \big)$ implies that at least one of the following must hold:

$$R^* (\bar{z}^*_{s,m,t} - \gamma^*_m + c_{s,m,t}) \leq 0 \quad \forall m = 1 : T_{max} \tag{4.2}$$

$$R_i (\bar{z}_{s_i,m,t} - \gamma_{i,m} - c_{s_i,m,t}) \geq 0 \quad \forall m = 1 : T_{max} \tag{4.3}$$

$$R^* \sum_{m=1}^{T_{max}} \gamma^*_m - R_i \sum_{m=1}^{T_{max}} \left(\gamma_{i,m} + 2 c_{s_i,m,t} \right) < 0. \tag{4.4}$$

For all $m = 1 : T_{max}$, we bound the probability of event (4.2) using Hoeffding bound and $c_{s,m,t} = \sqrt{\frac{2\ln(t)}{n_{s,m,t}}}$:

$$\begin{aligned}
&\mathbf{P}(R^*(\bar{z}^*_{s,m,t} - \gamma^*_m + c_{s,m,t}) \leq 0) = \\
&\mathbf{P}(\bar{z}^*_{s,m,t} \leq \gamma^*_m - c_{s,m,t}) \leq \\
&e^{-2n_{s,m,t}c^2_{s,m,t}} = \\
&e^{-2n_{s,m,t}\frac{2\ln(t)}{n_{s,m,t}}} = \\
&e^{-4\ln(t)} = t^{-4}. \qquad (4.5)
\end{aligned}$$

Similarly, for all $m = 1 : T_{max}$, we bound the probability of event (4.3) using Hoeffding bound and $c_{s_i,m,t} = \sqrt{\frac{2\ln(t)}{n_{s_i,m,t}}}$:

$$\mathbf{P}(R_i(\bar{z}^*_{s_i,m,t} - \gamma^*_m + c_{s_i,m,t}) \leq 0) \leq t^{-4} \qquad (4.6)$$

Now we look for a $n_{s_i,t}$ so that inequality (4.4) does not hold. This is equivalent to find a solution such that:

$$R^* \sum_{m=1}^{T_{max}} \gamma^*_m - R_i \sum_{m=1}^{T_{max}} \left(\gamma_{i,m} + 2\sqrt{\frac{2\ln(t)}{n_{s_i,m,t}}} \right) \geq 0. \qquad (4.7)$$

We denote $R^* \sum_{m=1}^{T_{max}} \gamma^*_m - R_i \sum_{m=1}^{T_{max}} \gamma_{i,m}$ by $\tilde{\Delta}_i$. Notice that:

$$\begin{aligned}
&\tilde{\Delta}_i - 2R_i \sum_{m=1}^{T_{max}} \sqrt{\frac{2\ln(t)}{n_{s_i,m,t}}} \geq \\
&\tilde{\Delta}_i - 2R_i \sum_{m=1}^{T_{max}} \sqrt{\frac{2\ln(t)}{\max\{0, n_{s_i,t} - m + 1\}}} \geq \qquad (4.8) \\
&\tilde{\Delta}_i - 2R_i \sum_{m=1}^{T_{max}} \sqrt{\frac{2\ln(t)}{\max\{0, n_{s_i,t} - T_{max} + 1\}}} \geq \qquad (4.9) \\
&\tilde{\Delta}_i - 2R_i T_{max} \sqrt{\frac{2\ln(t)}{\max\{0, n_{s_i,t} - T_{max} + 1\}}}. \qquad (4.10)
\end{aligned}$$

Assuming that $n_{s_i,t} \geq T_{max}$, we find the following bound for $n_{s_i,t}$:

$$\begin{aligned}
&\tilde{\Delta}_i - 2R_i T_{max} \sqrt{\frac{2\ln(t)}{n_{s_i,t} - T_{max} + 1}} \geq 0 \\
&n_{s_i,t} \geq T_{max} - 1 + \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2}. \qquad (4.11)
\end{aligned}$$

The bound on the average number of time we pull suboptimal arm i is the following:

$$\begin{aligned}
\mathbb{E}[\mathfrak{T}_i(n)] \leq & \left\lceil T_{max} - 1 + \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2} \right\rceil \\
& + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} \sum_{m=1}^{T_{\max}} \mathbf{P}(R^*(\bar{z}_{s,m,t}^* - \gamma_m^* + c_{s,m,t}) \leq 0) \qquad (4.12) \\
& + \mathbf{P}(R_i(\bar{z}_{s_i,m,t} - \gamma_m + c_{s_i,m,t}) \leq 0) \\
& \leq T_{max} + \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2} + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} \sum_{m=1}^{T_{\max}} 2t^{-4} \\
& \leq T_{max} + \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2} + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_i=\ell}^{t-1} 2T_{\max} t^{-4} \\
& \leq T_{max} + \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2} + T_{\max} \frac{\pi^2}{3} \\
& \leq \frac{8R_i^2 T_{max}^2 \ln(t)}{\tilde{\Delta}_i^2} + T_{\max} \left(1 + \frac{\pi^2}{3}\right).
\end{aligned}$$

4.2.3 PR-NT-UCB-P

PR-NT-UCB-P (Persistent - Non-Terminated Buckets - UCB - Pull Reward) is an algorithm tailored for scenarios where we want to maximize *Pull Reward* and we have a deterministic feedback R_j for each arm a_j. With this algorithm we want to exploit all the information of the buckets acquired during the learning process, considering both the terminated and non-terminated ones. To do that, at each time instant t, for each arm a_j, we fill the non-terminated buckets with fake realizations (line 5). To fill a non-terminated bucket $\boldsymbol{Z_{j,s}}$ means that we impose each component $Z_{j,s,m}$ not-parsed yet to be equal to a fake realization. At this point, we consider the altered bucket $\boldsymbol{Z_{j,s}}$ fully parsed. We fill half of the non-parsed components of a bucket with ones and half with zeros. At each time instant t, for each arm a_j, the exploration term $c_j(t)$ and the index $u_j(t)$ are computed in the following way (respectively at line 8 and at line 9):

$$c_j(t) = \sqrt{\frac{2T_{\max} \ln t}{n_j(t)} + \frac{T_{\max}(T_{\max} - 1)}{2n_j(t)}} \ ,$$

$$u_j(t) = R_j \left(\frac{\sum_{s=1}^{t} \sum_{m=1}^{T_{\max}} Z_{j,s,m}}{n_j(t)} + c_j(t) \right) = \frac{\sum_{s=1}^{t} X_{j,s}}{n_j(t)} + R_j c_j(t),$$

Algorithm 7 PR-NT-UCB-P

```
Require: bucket size $T_{\max}$
 1: function POLICY($T_{\max}$)
 2:     for each arm $a_j \in A$ do
 3:         for each bucket $\boldsymbol{Z_{j,s}}$ do
 4:             if $\boldsymbol{Z_{j,s}}$ is not terminated then
 5:                 fill $\boldsymbol{Z_{j,s}}$ with fake realizations
 6:             end if
 7:         end for
 8:         compute $c_j$
 9:         compute $u_j$
10:         for each bucket $\boldsymbol{Z_{j,s}}$ do
11:             remove fake realizations from $\boldsymbol{Z_{j,s}}$ if any
12:         end for
13:     end for
14: end function
```

where $n_j(t)$ is the number of pulls totalized by the arm a_j at time t. The index $u_j(t)$ is the sum of two terms: (i) the average Pull Reward computed considering all the acquired buckets; (ii) the exploration term. Once $u_j(t)$ has been computed, all the fake realizations are removed from the buckets (line 11).

PR-NT-UCB-P Theoretical Analysis

Theorem 4. *For $K \geq 1$, if policy PR-NT-UCB-P is run in myopic configuration on K arms having deterministic feedback $R_1, \dots, R_K$, then the bound on the expected regret, when the number of pulls $t \to \infty$, is of order $O(T_{\max} \ln(t))$.*

Proof of Theorem 4 It is possible to show that any algorithm that uses an arbitrary combination of zeros and ones achieves a regret bound of the same order. When we fill half of the non-parsed components with ones and half with zeros we get the best regret bound, however, it differs from the worst-case bound only by a small multiplicative constant. Therefore, in the following, we analyze the case in which we substitute ones. Notice that the same proof can be extended to any other algorithm substituting combinations of zeros and ones.

Simplifying the notation, we consider the bucket $\boldsymbol{z_i} = (z_{i,1}, \dots, z_{i,T_{\max}})$.

We denote with n_i the number of pulls of the arm i.

$$\phi(z_{1,1}, \dots, z_{1,T_{max}}, \dots, z_{n_i,1}, \dots, z_{n_i,T_{max}}) = \sum_{j=1}^{n_i} \sum_{k=1}^{T_{max}} z_{j,k}.$$

$$\sup_{z_{1,1},\dots,z_{1,T_{max}},\dots,z_{j,k},z_{j',k'},\dots,z_{n_i,1},\dots,z_{n_i,T_{max}}} |\phi(z_{1,1}, \dots, z_{1,T_{max}}, \dots, \dots, z_{j,k}, \dots, z_{n_i,1}, \dots, z_{n_i,T_{max}}) - \phi(z_{1,1}, \dots, z_{1,T_{max}}, \dots, z_{j',k'}, \dots, z_{n_i,1}, \dots, z_{n_i,T_{max}})| \leq 1.$$

We apply McDiarmid's Inequality (Rio et al., 2013):

$$\mathbb{P}(\phi - \mathbb{E}[\phi] \geq \bar{\varepsilon}) \leq \exp\left(\frac{-2\bar{\varepsilon}^2}{\sum_{\ell=1}^{n_i T_{max}} 1^2}\right) = \exp\left(\frac{-2\bar{\varepsilon}^2}{n_i T_{max}}\right),$$

$$\mathbb{P}\left(\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi]}{n_i} \geq \varepsilon\right) \leq \exp\left(\frac{-2\varepsilon^2 n_i}{T_{max}}\right).$$

Function ϕ is the approximation of the true function ϕ_v. Function ϕ sums the fake elements used to complete the vectors, while ϕ_v uses the true realizations. Notice that, when filling the missing elements with ones, $\mathbb{E}[\phi] \geq \mathbb{E}[\phi_v]$.

We need to bound the probability of the following events:

- $\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi_v]}{n_i} \leq -\varepsilon$;
- $\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi_v]}{n_i} \geq \varepsilon$.

We fix the exploration term ε as:

$$\varepsilon = c_{t,n_i} = \sqrt{\frac{2T_{max} \ln t}{n_i}} + \frac{T_{max}(T_{max} - 1)}{2n_i},$$

so that the probability can be bounded as shown below.

For the first event we show that:

$$\mathbb{P}\left(\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi_v]}{n_i} \leq -\varepsilon\right) \leq \mathbb{P}\left(\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi]}{n_i} \leq -\varepsilon\right) \leq \exp\left(-\frac{2\varepsilon^2 n_i}{T_{max}}\right) \leq \tag{4.13}$$

$$\exp\left(-\frac{2\left(\sqrt{\frac{2T_{max} \ln t}{n_i}}\right)^2 n_i}{T_{max}}\right) \leq \exp(-4 \ln t) \leq t^{-4}. \tag{4.14}$$

Second event:

$$\mathbb{P}\left(\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi_v]}{n_i} \geq \varepsilon\right) = \mathbb{P}\left(\frac{\phi}{n_i} - \frac{\mathbb{E}[\phi]}{n_i} \geq \varepsilon + \frac{\mathbb{E}[\phi_v]}{n_i} - \frac{\mathbb{E}[\phi]}{n_i}\right) \leq \tag{4.15}$$

$$\exp\left(-\frac{2\left(\varepsilon - \frac{\mathbb{E}[\phi]-\mathbb{E}[\phi_v]}{n_i}\right)^2 n_i}{T_{max}}\right) \leq \exp\left(-\frac{2\left(\varepsilon - \frac{T_{max}(T_{max}-1)}{2n_i}\right)^2 n_i}{T_{max}}\right) \leq \tag{4.16}$$

$$\exp\left(-\frac{2\left(\sqrt{\frac{2T_{max}\ln t}{n_i}}\right)^2 n_i}{T_{max}}\right) \leq \exp(-4\ln t) \leq t^{-4}. \tag{4.17}$$

In the bound at line 4.16 we use the maximum difference between $\mathbb{E}[\phi]$ and $\mathbb{E}[\phi_v]$. Assuming that $n_i \geq T_{max}$, the worst-case scenario is when arm i has been pulled T_{max} consecutive times. In this scenario there is a maximum number of missing elements equal to $\frac{T_{max}(T_{max}-1)}{2}$, that are all replaced by ones and summed in ϕ. In the worst case the true realizations of are all equal to zero.

Now we compute a n_i such that the probability of event $\frac{\phi^*}{n_i} - \frac{\phi_i}{n_i} - 2\varepsilon < 0$ is equal to zero. We denote $\frac{\phi^*}{n_i} - \frac{\phi_i}{n_i}$ by Δ_i.

$$n_i \geq \frac{1}{\Delta_i^2}\left(\Delta_i T_{max}(T_{max}-1) + 4T_{max}\ln(t) + \sqrt{16T_{max}^2\ln^2(t) + 8\Delta_i T_{max}^2(T_{max}-1)\ln(t)}\right)$$

Notice that as $t \to \infty$, the bound is of order $O(T_{\max}\ln(t))$.

4.3 Bayesian Algorithms

In this section we present the novel algorithms that adopt a Bayesian approach. We developed the following policies: PR-T-TS-P, PR-T-TS-NP, PR-BW-BayesUCB-P, PR-BW-BayesUCB-NP.

4.3.1 PR-T-TS-P (Bayesian Baseline Algorithm)

PR-T-TS (Persistent - Terminated Buckets - Thompson Sampling - Pull Reward) is an algorithm that extends the idea of the well known Thompson Sampling algorithm in case of persistent reward and deterministic feedback R_j, where we want to maximize the pull reward. Similarly to Algorithm 5, PR-T-TS considers the reward as a unique variable available after the termination of the bucket. The pseudo-code of PR-T-TS is provided in Algorithm 8. The algorithm, for each arm a_j, has a Beta distribution representing the persistency $\mathfrak{p}_j = \frac{\mu_j}{R_j Tmax}$. At each time t, having observed $S_j(t)$ successes

Algorithm 8 `PR-T-TS`

```
Require: arm set $A = \{a_1, a_2, \dots, a_k\}$, time horizon $N$, bucket size $T_{\max}$
 1: function POLICY($T_{\max}$)
 2:     for each arm $a_j \in A$ do                                    ▷ init phase
 3:         $S_j \leftarrow 1$ and $F_j \leftarrow 1$
 4:     end for
 5:     for $t \in \{1, \dots, N\}$ do                                 ▷ loop phase
 6:         for each arm $a_j \in A$ do
 7:             sample $\theta_j$ from the Beta($S_j$,$F_j$) distribution
 8:         end for
 9:         pull arm $a_j$ such that $j = \arg\max_i \theta_i R_i$
10:         for each arm $a_j \in A$ do                                ▷ update phase
11:             for each bucket $\boldsymbol{Z_{j,s}}$ do
12:                 if $\boldsymbol{Z_{j,s}}$ is terminated then
13:                     compute $\mathfrak{p}_{j,s}$
14:                     Perform a Bernoulli trial with success probability $\mathfrak{p}_{j,s}$
    and observe output r
15:                     if r = 1 then
16:                         $S_j \leftarrow S_j + 1$
17:                     else
18:                         $F_j \leftarrow F_j + 1$
19:                     end if
20:                 end if
21:             end for
22:         end for
23:     end for
24: end function
```

and $F_j(t)$ failures, the distributions on $\mathfrak{p}_j$ are updated as Beta$(S_j(t), F_j(t))$. For each arm a_j, it is sampled θ_j from the distributions on $\mathfrak{p}_j$. The arm having the largest $\theta_j R_j$ is pulled. Note that $\mathfrak{p}_j R_j T_{\max} = \mu_j$, hence we are trying to pull the arm that has the best expected reward based on the previous successes and failures (we do not consider $T_{\max}$ in the choice of the arm since it is a constant). When a bucket $\boldsymbol{Z_{j,s}}$ of the arm a_j is terminated, the persistency $\mathfrak{p}_{j,s}$ is computed as follows: $\mathfrak{p}_{j,s} = \frac{\sum_{m=1}^{T_{\max}} Z_{j,s,m}}{T_{\max}}$. At this point, the success and failures counters are updated based on a Bernoulli trial with success probability $\mathfrak{p}_{j,s}$. This implementation allows us to adopt the idea behind the stochastic TS algorithm proposed by Agrawal and Goyal (2012).

4.3.2 PR-T-TS-NP (Bayesian Baseline Algorithm)

PR-T-TS-NP (Persistent - Terminated Buckets - Thompson Sampling - Normalized Pull Reward) is an algorithm that extends the idea of the well known Thompson Sampling algorithm in case of persistent reward. It is tailored for scenarios where we want to maximize the normalized pull reward and we have deterministic feedback R_j. It differs from PR-T-TS-P, previously described, only for the definition of the persistency $\mathfrak{p}_j$. Since we are dealing with the normalized pull reward, we consider $\mathfrak{p}_j = \frac{\eta_j}{R_j}$. When a bucket $\boldsymbol{Z_{j,s}}$ is terminated, $\mathfrak{p}_{j,s}$ is computed as follows: $\mathfrak{p}_{j,s} = \frac{\sum_{m=1}^{T_{\max}} Z_{j,s,m}}{l_{j,s}}$, where $l_{j,s}$ is the true length. the pseudo code of PR-T-TS-NP is depicted in Algorithm 8.

4.3.3 PR-BW-BayesUCB-P

PR-BW-BayesUCB-P (Persistent - Bin-Wise - BayesUCB - Pull Reward) is an algorithm based on the BayesUCB algorithm proposed by Kaufmann et al. (2012). It maximizes the pull reward and assumes deterministic feedback R_j. We want to exploit the partial information deriving from the non-terminated buckets, trying to learn the distribution of each single bin of a bucket. More precisely, for each arm a_j, we maintain a Beta distribution for each bin $Z_{j,m}$ with $m = 1, \dots, T_{\max}$. At each time t, having observed $S_{j,m}(t)$ successes and $F_{j,m}(t)$ failures, the distributions of $Z_{j,m}$ are updated as Beta($S_{j,m}(t)$,$F_{j,m}(t)$). At each time t, we compute $q_{j,m}(t) = Q(1 - \frac{1}{t}, Beta(S_{j,m}(t), F_{j,m}(t)))$, where with $Q(\alpha, \rho)$ we indicate the quantile function associated to the distribution ρ. The arm a_j which has the largest $\tilde{\mu_j}(t) = R_j \sum_{m=1}^{T_{\max}} q_{j,m}(t)$ is pulled. The counters $S_{j,m}(t)$ and $F_{j,m}(t)$ are updated every time a component $Z_{j,s,m}$ is parsed. More precisely, depending on the configuration we will have that:

- in *myopic* configuration: $Z_{j,s,m}$ is parsed when: $t = s + m - 1$;
- in *farsighted* configuration: $Z_{j,s,m}$ is parsed when: $t = s + m - 1 \vee (t = s + l_{j,s} - 1 \wedge m \geq l_{j,s})$.

The update is done according to a Bernoulli Trial with success probability $Z_{j,s,m}$. The pseudo-code of the PR-BW-BayesUCB-NP is provided in Algorithm 9.

4.3.4 PR-BW-BayesUCB-NP

PR-BW-BayesUCB-NP (Persistent - Bin-Wise - BayesUCB - Normalized Pull Reward) is an algorithm based on the BayesUCB algorithm. It max-

Algorithm 9 PR-BW-BayesUCB-P

```
Require: arm set $A = \{a_1, a_2, \dots, a_k\}$, time horizon $N$, bucket size $T_{\max}$
1: function POLICY($T_{\max}$)
2:     for each arm $a_j \in A$ do                                    ▷ init phase
3:         for each $m \in (0, \dots, T_{\max})$ do $S_{j,m} \leftarrow 1$ and $F_{j,m} \leftarrow 1$ end for
4:     end for
5:     for $t \in \{1, \dots, N\}$ do                                  ▷ loop phase
6:         for each arm $a_j \in A$ do
7:             for each $m \in (0, \dots, T_{\max})$ do compute $q_{j,m}$ end for
8:         end for
9:         pull arm $a_j$ such that $j = \arg\max_i \sum_{m=1}^{T_{\max}} q_{i,m} R_i$
10:        for each component $Z_{j,s,m}$ parsed at time $t$ do          ▷ update phase
11:            Perform a Bernoulli trial with success probability $Z_{j,s,m}$ and observe output r
12:            if r = 1 then
13:                $S_{j,m} \leftarrow S_{j,m} + 1$
14:            else
15:                $F_{j,m} \leftarrow F_{j,m} + 1$
16:            end if
17:        end for
18:    end for
19: end function
```

imizes the normalized pull reward and assumes deterministic feedback R_j. The pseudo-code of PR-BW-BayesUCB-NP is provided in Algorithm 10. Similarly to the pull reward version (PR-BW-BayesUCB-P) previously described, we want to exploit the partial information deriving from the non-terminated buckets. More precisely, for each arm a_j and for each $m = 1, \dots, T_{\max}$, the algorithm maintains a Beta distribution for the delay $d_{j,m}$ and one for the bin $Z_{j,m}$. At each time t, having observed $S_{j,m}(t)^d$, $S_{j,m}(t)^z$ successes and $F_{j,m}(t)^d$, $F_{j,m}(t)^z$ failures, the distributions of $Z_{j,m}$ are updated as Beta($S_{j,m}(t)^z$,$F_{j,m}(t)^z$) and the ones of $d_{j,m}$ are updated as Beta($S_{j,m}(t)^d$,$F_{j,m}(t)^d$). At each time t, we compute:

- $q_{j,m}(t)^{z_{ucb}} = Q(1 - \frac{1}{3t}, Beta(S_{j,m}(t)^z, F_{j,m}(t)^z))$;
- $q_{j,m}(t)^{z_{lcb}} = Q(\frac{1}{3t}, Beta(S_{j,m}(t)^z, F_{j,m}(t)^z))$;
- $q_{j,m}(t)^{d_{lcb}} = Q(\frac{1}{3t}, Beta(S_{j,m}(t)^d, F_{j,m}(t)^d))$.

With $Q(\alpha, \rho)$ we indicate the quantile function associated to the distribution ρ. At this point, a lower confidence bound of the true length $\tilde{l}_j(t)$ is computed

Algorithm 10 `PR-BW-BayesUCB-NP`

```
Require: arm set $A = \{a_1, a_2, \dots, a_k\}$, time horizon $N$, bucket size $T_{\max}$
 1: function POLICY($T_{\max}$)
 2:     for each arm $a_j \in A$ do                                          ▷ init phase
 3:         for each $m \in (0, \dots, T_{\max})$ do $S^z_{j,m} \leftarrow 1$ and $F^z_{j,m} \leftarrow 1$ end for
 4:         for each $m \in (0, \dots, T_{\max})$ do $S^d_{j,m} \leftarrow 1$ and $F^d_{j,m} \leftarrow 1$ end for
 5:     end for
 6:     for $t \in \{1, \dots, N\}$ do                                        ▷ loop phase
 7:         for each arm $a_j \in A$ do
 8:             for each $m \in (0, \dots, T_{\max})$ do compute $q^{z_{ucb}}_{j,m}, q^{z_{lcb}}_{j,m}, q^{d_{lcb}}_{j,m}$ end
    for
 9:         end for
10:         pull arm $a_j$ such that $j = \arg\max_i \tilde{\eta}_i$
11:         for each component $Z_{j,s,m}$ parsed at time $t$ do            ▷ update phase
12:             Perform a Bernoulli trial with success probability $Z_{j,s,m}$ and
    observe output r
13:             if r = 1 then
14:                 $S^z_{j,m} \leftarrow S^z_{j,m} + 1$
15:             else
16:                 $F^z_{j,m} \leftarrow F^z_{j,m} + 1$
17:             end if
18:             if $m \leq d_{j,s}$ then
19:                 $S^d_{j,m} \leftarrow S^d_{j,m} + 1$
20:             else
21:                 $F^d_{j,m} \leftarrow F^d_{j,m} + 1$
22:             end if
23:         end for
24:     end for
25: end function
```

as $\tilde{l}_j(t) = \sum_{m=1}^{T_{\max}} q_{j,m}(t)^{z_{lcb}} + q_{j,m}(t)^{d_{lcb}}$. Finally, an upper confidence bound on the normalized pull reward is computed as $\tilde{\eta_j}(t) = \frac{R_j \sum_{m=1}^{T_{\max}} q_{j,m}(t)^{z_{ucb}}}{\tilde{l}_j(t)}$. At each time t, the arm a_j having the largest $\tilde{\eta_j}(t)$ is pulled. The update of the failures and success counters is done in the same way of the algorithm PR-BW-BayesUCB-P, with the required adaptation for the delay $d_{j,m}$ (line 11 - 22).

Chapter 5

Experimental Analysis

In this chapter, we present the experimental results of our analysis. In particular, we run all the policies considered so far in a variety of configurations and compare their performance in terms of pseudo-regret and normalized pseudo-regret. In the first section, we describe the settings of our synthetic experiments and the corresponding results. In the second and third section, we present the analysis of the two real-world scenarios formalized in Chapter 4.

5.1 Synthetic Experiment Settings

All the synthetic experiments analyzed are in *tight persistency* with delay $d_{j,t} = 0$ for each realization $\boldsymbol{Z_{j,t}} = (Z_{j,t,1}, \ldots, Z_{j,t,T_{\max}})$ of the persistency vector, where each component $Z_{j,t,m}$ is a *Bernoulli variable.* The feedback $R_{j,t}$ is assumed to be deterministic for each arm a_j, therefore we will omit the index t. We assume that $T_{\max}$ is known by the learner. For each experiment, we want to maximize the accumulated *Pull Reward*, so we will evaluate the results in terms of *Pseudo-Regret* (the smaller the better). In this scenario, a generic realization of the persistency vector, informally called *bucket*, will be a sequence of $T_{\max}$ elements composed by a sub-sequence of ones followed by a sub-sequence of zeros. Note that, given a bucket $\boldsymbol{Z_{j,t}}$, since the delay $d_{j,t}$ is assumed to be 0 and we are in tight persistency, the length of the sub-sequence of ones is equal to the true length $l_{j,t}$.

To generate synthetic data, at each time instant t, we sample the true length $l_{j,t}$ of a new bucket $\boldsymbol{Z_{j,t}}$ from a distribution associated to the pulled arm a_j. More specifically, each arm a_j is associated with a distribution Beta(α_j,β_j), where the parameters α_j and β_j are specified according to the considered setting. The expected pull reward μ_j of an arm a_j is computed

Table 5.1: Experimental Analysis Summary.

	Persistency		**Reward**	
Experiment name	*General*	*Tight*	*P.R.*	*N.P.R.*
synthetic A,B,C		x	x	
Spotify Scenario	x		x	
Rent Scenario		x		x

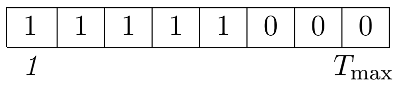

Figure 5.1: Example of bucket in synthetic setting. In this example the length of the vector is $T_{\max} = 8$. The first 5 components are successes (1), meaning that the true length of the vector is $l = 5$.

in the following way:

$$\mu_j = R_j \sum_{i=1}^{T_{\max}} i\left(F_j\left(\frac{i}{T_{\max}}\right) - F_j\left(\frac{i-1}{T_{\max}}\right)\right),$$

where F_j is the cumulative distribution function of the Beta(α_j,β_j). Below we present three significant settings analyzed: *Synthetic A*, *Synthetic B*, and *Synthetic C*.

All the synthetic settings presented are designed as to exemplify real problems. We tested all the algorithms developed tailored to maximize pull reward, in both the myopic and farsighted configurations. We consider as baselines the ones that do not exploit partial information. The list of the tested algorithms is the following: PR-T-UCB-P, PR-BW-UCB-P, PR-NT-UCB-P, PR-T-TS-P, PR-BW-BayesUCB-P. Where PR-T-UCB-P and PR-T-TS-P are the algorithms considered as baselines.

5.1.1 Synthetic A

In this setting, the value of the true length $l_{j,t}$ is sampled from a Beta distribution with parameters $a_j = b_j = 1$, for each arm a_j, for each time instant t. In this configuration the Beta distribution is equivalent to the uniform distribution. The feedback R_j is set incrementally for each arm a_j. Here, the best arm is the one with maximum feedback R_j. Indeed, the magnitude of R_j does not influence the true length $l_{j,t}$ which is generated uniformly at random. We set $T_{\max} = 50$. The experiment is repeated for 50 independent runs. The full description of the arms is provided in Table 5.2.

Table 5.2: Description of the arms in setting Synthetic A.

Arm	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
R	1	2	3	4	5	6	7	8	9	10
$\boldsymbol{\mu}$	25.5	51	76.5	102	127.5	153	178.5	204	229.5	255

Table 5.3: Description of the arms in setting Synthetic B.

Arm	a_0	a_1	a_2	a_3	a_4	a_5
R	1	1	1	1	1	1
$\boldsymbol{\alpha}$	2	4	6	8	10	12
$\boldsymbol{\beta}$	8	8	8	8	8	8
$\boldsymbol{\mu}$	10.50	17.17	21.93	25.50	28.28	30.3

This setting is analysing the case in which the rewards are uniformly spread over the $T_{\max}$ time instants subsequent to the selection of an arm.

5.1.2 Synthetic B

In this setting, for each time instant t, the true length $l_{j,t}$ is sampled from a Beta distribution depending on the pulled arm a_j. For each arm a_j, the feedback is $R_j = 1$. This implies that the best arm is the one which is associated to the Beta with the highest mean. We set $T_{\max} = 50$. The experiment is repeated for 50 independent runs. The full description of the arms is provided in Table 5.3, and a visual representation of the Beta distributions considered are provided in Figure 5.2. This setting could be seen as a simplified version of the Medical Trial problem proposed in Example 2, where we are interested in capturing only the lifetime of a patient.

5.1.3 Synthetic C

In this setting, we model the common situation where high feedback discourage long persistency, as previously discussed in Chapter 3. At each time t, the true length $l_{j,t}$ is sampled from a Beta distribution depending on the pulled arm a_j. For each arm a_j, the feedback is R_j is set such that to higher Beta mean corresponds lower feedback. We set $T_{\max} =\in \{50, 100, 150, 200\}$. For each $T_{\max}$ adopted, the experiment is repeated for 50 independent runs. The full description of the arms is provided in Table 5.4, where with $\mu_{T_{\max}}$ we indicate the expected pull reward in the configuration where $T_{\max}$ is adopted. The visual representation of the Beta distribution is provided in Figure 5.2. This settings exemplifies a case of dynamic pricing of a subscription-based

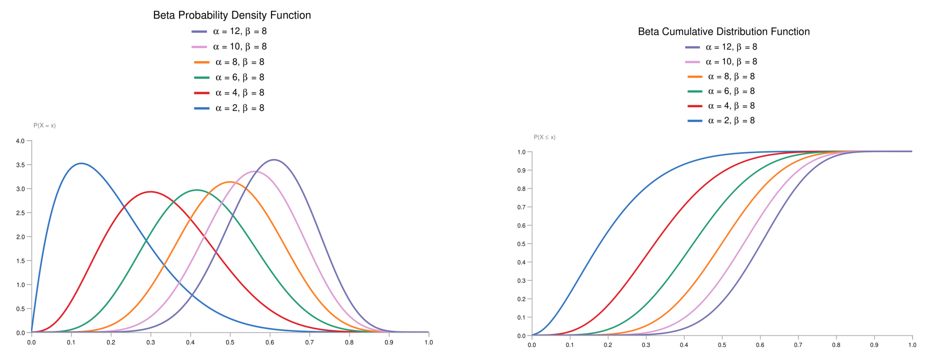

Figure 5.2: Probability density function and cumulative distribution function of the Beta distributions considered in settings Synthetic B and Synthetic C.

Table 5.4: Description of the arms in setting Synthetic C.

Arm	a_0	a_1	a_2	a_3	a_4	a_5
R	6	5	4	3	2	1
$\boldsymbol{\alpha}$	2	4	6	8	10	12
$\boldsymbol{\beta}$	8	8	8	8	8	8
$\boldsymbol{\mu_{50}}$	63.00	85.83	87.71	76.50	56.55	30.50
$\boldsymbol{\mu_{100}}$	123.00	169.16	173.43	151.50	112.11	60.50
$\boldsymbol{\mu_{150}}$	183.00	252.50	259.14	226.50	167.67	90.5
$\boldsymbol{\mu_{200}}$	243.00	335.83	344.85	301.50	223.22	120.50

service. Indeed, large prices corresponds short time in terms of the subscription, and viceversa.

Results on Synthetic Datasets The experimental results of the the settings Synthetic A and Synthetic B, depicted in Figure 5.3 and 5.4, respectively, show that the frequentist algorithms which exploit partial information outperforms the frequentist baselines. Similarly, for the Bayesian algorithms the baselines have worse performance compared to the algorithms that exploit partial information, where the difference is less evident. Synthetic A and Synthetic B settings, despite having different characteristics, obtain the same ranking in terms of algorithms: PR-BW-BayesUCB-P is the best overall, and PR-NT-UCB-P is the best among the frequentist ones.

The results for the setting Synthetic C, provided in Figure 5.5, shows an interesting situation. Indeed, in this setting the PR-NT-UCB-P algorithm performances degrades more than the other we analysed as the value of

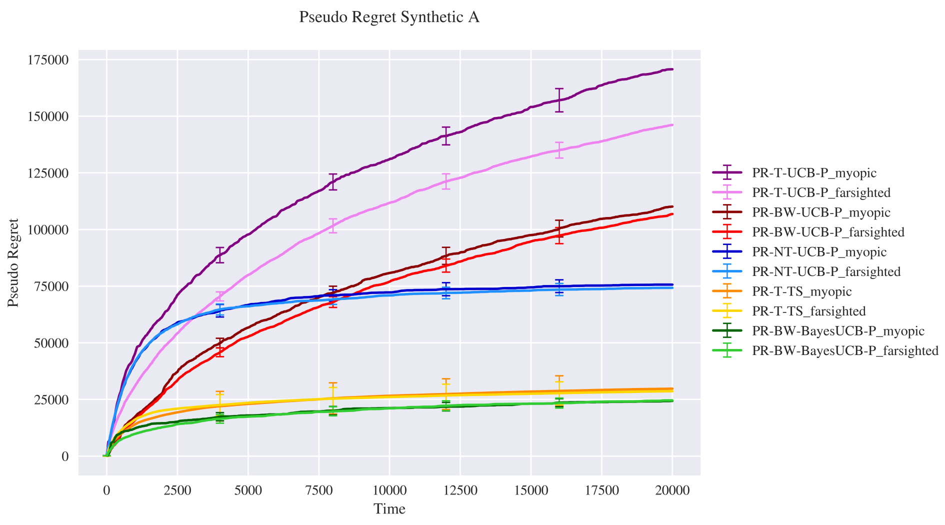

Figure 5.3: Pseudo regret plot of the experiment Synthetic A.

$T_{\max}$ increases. In Figure 5.5 we notice that, when $T_{\max} = 50$, PR-NT-UCB-P is the best frequentist algorithm, but as $T_{\max}$ increases, it performs worse than the others. Conversely, for the other algorithms we have a result comparable to the one obtained in Synthetic A and Synthetic B. Nonetheless, extending the time horizon, as depicted in Figure 5.6, we notice that, with an adequate time horizon, PR-NT-UCB-P still outperforms all the other frequentist algorithms. This is in line with our theoretical results summarized in Table 4.2, telling that this algorithm should improve in this setting over the others for its linear dependence w.r.t. $T_{\max}$.

The experimental results show that an algorithm in farsighted configuration performs better than the corresponding myopic one. This was expected since the farsighted configuration anticipates information to the learner. However, this improvement is significant only in a few cases, for example for PR-T-UCB-P in Figure 5.3, and PR-BW-BayesUCB-P in Figure 5.5 with $T_{\max} = 200$. A final note is that in all the analyzed synthetic settings, the policies that adopt a Bayesian approach achieves better performance compared to ones adopting the frequentist approach. This is a well-known fact from the bandit literature, even if in principle their theoretical guarantees are the same.

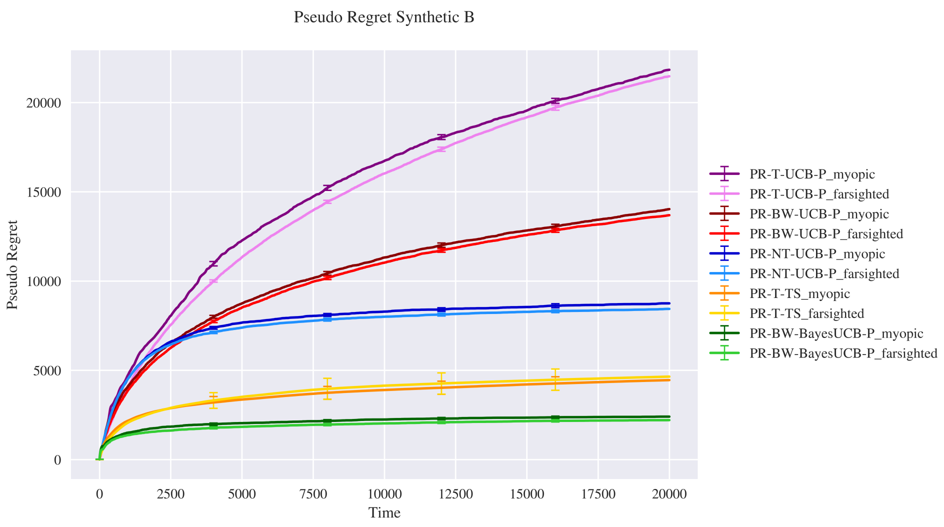

Figure 5.4: Pseudo regret plot of the experiment Synthetic B.

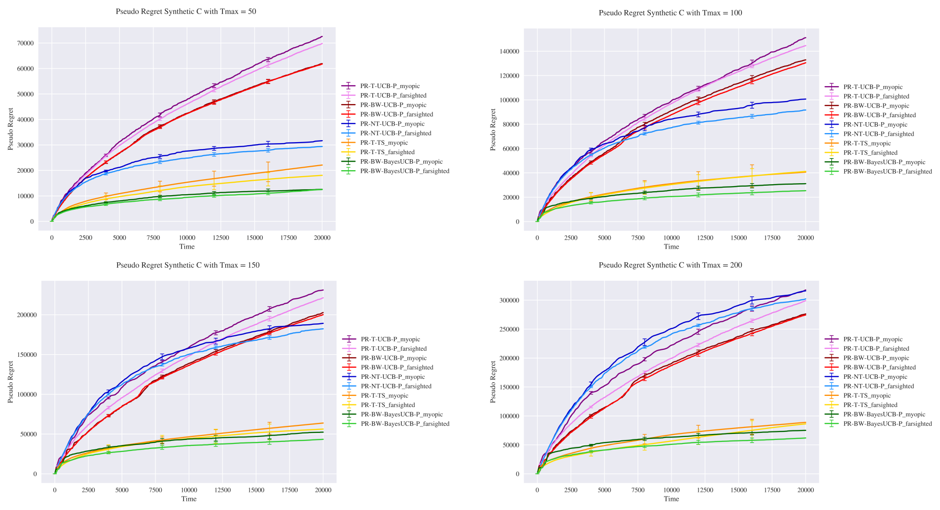

Figure 5.5: Pseudo regret plots of the experiment Synthetic C. From the top: $T_{\max} \in \{50, 100, 150, 200\}$.

5.2 Spotify Playlist Problem Setting

We perform an experimental analysis of the problem formalized in Section 3.2.1. At each round t the buckets $\boldsymbol{Z_{j,t}}$ is sampled from a portion of an official dataset released by Spotify (Brost et al., 2019). More precisely, we extract from the dataset the listening sessions corresponding to $K = 6$ playlists, which are our arms. Then, at each round t, we sample a listening session of the pulled playlist and we encode the bucket as indicated in the formal-

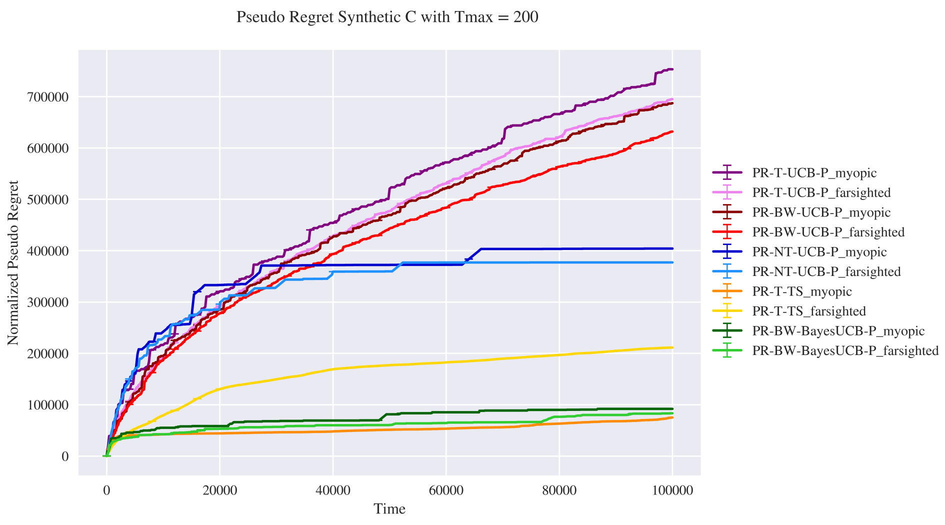

Figure 5.6: Pseudo regret plot of the experiment Synthetic C with $T_{\max} = 200$ and extended time horizon $n = 10^5$. The plot represents the pseudo-regret achieved in a single run experiment.

Table 5.5: Description of the arms in The Spotify Playlist Problem Setting.

Arm	a_0	a_1	a_2	a_3	a_4	a_5
$\boldsymbol{\mu}$	38.59	52.35	38.44	43.89	23.48	36.20
$\boldsymbol{\sigma(\mu)}$	21.83	20.11	23.09	23.14	23.48	23.8

ization. The expected pull reward of the seven playlists selected and the relative standard deviation are depicted in Table 5.6. Since we are in general persistency setting and we want to maximize the pull reward, we test all the algorithms tailored for the pull reward in myopic configuration. We evaluate the averaged pseudo regret obtained in 50 independent runs.

Results for the Spotify Setting We recall that differently from the aforementioned synthetic settings, the Spotify Playlist problem is in general-persistency, therefore there are no assumption on the bucket composition. While the tight persistency condition impose that every positive bin must be adjacent, the general persistency condition does not impose any constraints on how the bins are distributed in the bucket. The results of the Spotify experiment are provided in Figure 5.7. Among the frequentist algorithms PR-BW-UCB-P is really close to the baseline PR-T-UCB-P and, in the Bayesian ones, the baseline PR-T-TS-P overtakes PR-BW-BayesUCB-P. This result suggests that the bin-wise approach suffers in the general-persistency setting.

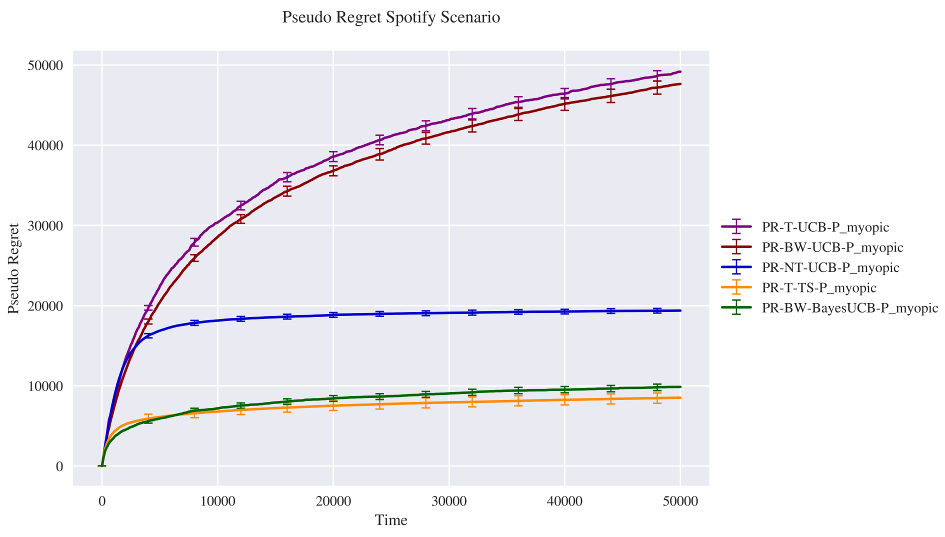

Figure 5.7: Pseudo regret plot of the experiment Spotify Playlist Problem.

As a matter of facts, while in the synthetic settings the reward is concentrated at the start of the buckets as depicted in Figure 5.1, in this scenario the reward is uniformly distributed over the bucket. This greatly undermine the advantage of the bin-wise approach. The PR-NT-UCB-P algorithm, which exploit the non terminated buckets filling them with fake realizations, does not suffer the issue of the reward distribution along the bucket and outperforms both the baseline PR-T-UCB-P and the bin-wise frequentist algorithm PR-BW-UCB-P. As a final remark, since some of the presented algorithms are able to learn quickly the best playlist, it could be interesting to compare the MAB-PR strategy with other solutions that mitigate the cold users issue.

5.3 Rental Pricing Problem Setting

We perform an experimental analysis of the problem formalized in Section 3.2.2. For this scenario we use a dataset provided by a company that works in the rental room business containing information about past rental contracts, whose information are retained for NDA reasons. Once an arm, which represents a fee, is pulled, a bucket is encoded sampling a contract from the sub-set of the dataset which contains the contracts characterized by the pulled fee. We identify a set of $K = 8$ different fees. A description of the arms is provided in Table 5.6. We test all the algorithms suitable for maximizing the normalized pull reward and we evaluate the results in terms of normalized pseudo regret. We evaluate the averaged pseudo regret obtained

Table 5.6: Description of the arms in The Rental Pricing Problem Setting.

Arm	a_0	a_1	a_2	a_3	a_4	a_5	a_5	a_5
R	550	560	570	580	590	600	610	630
$\boldsymbol{\eta}$	477	462	490	483	466	512	530	445

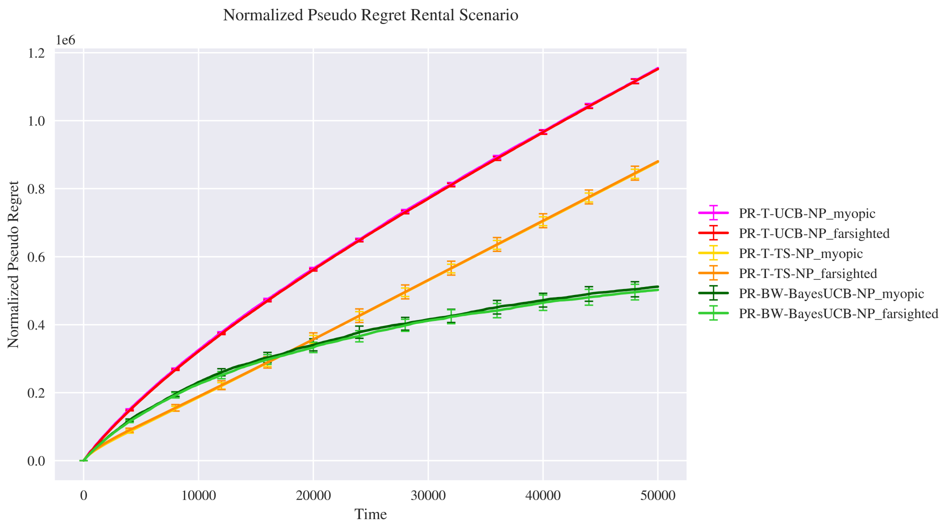

Figure 5.8: Normalized pseudo regret plot of the experiment Rental Pricing Problem.

over 50 independent runs.

Results for the Rental Pricing Setting The Rental Pricing problem described in Section 3.2.2, appears more difficult than the other settings. This is evident looking at Figure 5.8, where both the frequentist and Bayesian baselines are not able to identify the best arm quickly, since their regret do not curve in the normalized pseudo-regret plot. Nonetheless, the PR-BW-BayesUCB-NP algorithm is the one outperforming the others even in this scenario.

Chapter 6

Conclusions

The Multi-Armed Bandit model is an important framework for decision making under uncertainty, as it presents one of the clearest examples of the trade-off between exploration and exploitation. In problems of this type, an agent, at each time instant, is required to choose an action among a given set. Each action, when chosen, generates a reward for the agent. The goal of the agent is to maximize the cumulative reward over a finite time horizon. Although the standard bandit model assumes that the reward is a real number immediately available after an action has been taken, this assumption is inappropriate for a variety of real-world scenarios. In this thesis, motivated by real-world application needs, we studied the specific case in which the reward deriving from an action is spread over the time following the taking of the action. Applications that fall under persistent reward include, for example, dynamic pricing, web content optimization, recommender systems, and adaptive clinical trials.

We firstly formalized a new bandit model, namely Multi-Armed Bandit with Persistent Reward (MAB-PR), suitable to handle the persistent reward scenarios. Then, we designed a set of algorithms, following the Bayesian and frequentist framework, tailored to tackle this novel bandit model. Our major interest was to develop and analyze novel algorithms able to exploit the partial information obtained during the reward acquisition process. For this purpose, we introduced two different approaches: the bin-wise approach and the non-terminated approach. The former one takes advantage of the fact that the total reward deriving from an action can be seen as a sum of smaller rewards (bins) acquired at each time instants after the take of the action. Conversely, the latter one exploits the not yet terminated reward acquisition processes considering them terminated by faking the part of the process not yet occurred with fictitious information. The algorithms that base their operating criteria on these approaches have been compared with those that evaluate the reward as a unique variable available only at the end of the acquisition process that, namely the so called delayed bandit algorithms. Indeed, to consider the reward as a number available after that the acquisition process is terminated, reduces the persistent reward problem to a delayed reward problem.

For the frequentist algorithms we provided theoretical guarantees which, under mild conditions, state that both the delayed and the bin-wise algorithm PR-BW-UCB-P achieve a regret bound of the order of $O(T_{\max}^2 \ln t)$ in our setting. Furthermore, we proved that the algorithm PR-NT-UCB-P, which relies on the non-terminated approach, achieves a regret bound of the order of $O(T_{\max} \ln t)$, improving over the previous one of a multiplicative factor of $T_{\max}$.

We performed a thorough experimental analysis of the developed algorithms which includes both synthetically generated and real-world data. The experimental results show that, in the large majority of the settings analyzed, the novel algorithms that exploit partial information achieved better results compared to their corresponding baselines. Furthermore, the theoretical results are perfectly reflected in the outcome of the experimental analysis, where, for each addressed scenario, the algorithm PR-NT-UCB-P outperforms all the other frequentist algorithms when a sufficiently large time horizon is provided. Moreover, experimental evidence shows that the way in which the reward is distributed over the time severely affects the performance of the algorithms. More specifically, if a large portion of the total reward generated by an action is concentrated immediately after that the action has taken place, the advantage of capturing this information without waiting is significant. Conversely, if the reward is uniformly distributed over the timespan following the take of an action, the advantage gathered from an early evaluation is limited. In the worst case, it can even be misleading to exploit partial information if, in an initial phase, these are not representative of the total reward deriving from the take of an action. In our analysis, the Bayesian algorithms always achieve better performance compared to the frequentist ones. We show experimentally that the Bayesian way to manage the exploration exploitation trade-off is the best in the persistent reward scenario. This confirms what we expected since previous works shows that Thompson Sampling, which inspired our Bayesian algorithms, is more robust under delayed feedback thanks to its randomness (Chapelle and Li, 2011).

The main future directions of this work is to prove theoretical bounds for all the algorithms which have not been theoretically analyzed here, in particular the ones using the Bayesian approach. Our experimental results highlighted how the persistent reward settings is sensible to the the way in which the reward is distributed along its acquisition process. An interesting research line could be to theoretically formalize this aspect and to devise new strategies which explicitly exploits this information. Alternatively, could be also interesting to prove criteria that, based on the reward distribution along its acquisition process, determine when it is convenient to approach a specific persistent reward scenario with an algorithm that exploits partial information.

Extending the PR-MAB framework to handle more complex scenarios is another interesting development of the current work. In particular, we believe that the two most limiting assumptions for our setting are: (i) the absence of side information; and (ii) the stationarity of the reward. Indeed, these two assumptions are very significant in a lot of real problems that could be modeled as MAB-PR problem. For example, in a recommendation task, the absence of side information means that the algorithm considers that all the users to be served are identical, or, in a dynamic pricing problem, the stationarity of the reward implies that the algorithm is not able to react to the market variations. Indeed, could be interesting to mix our MAB-PR model with Contextual Bandits (Agarwal et al., 2014) or non-stationary Bandits (Trovo et al., 2020; Di Benedetto et al., 2020) to cope with (i) and (ii), respectively.

Bibliography

A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and R. Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646. PMLR, 2014.

S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

K. Amin, A. Rostamizadeh, and U. Syed. Learning prices for repeated auctions with strategic buyers. *arXiv preprint arXiv:1311.6838*, 2013.

V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part ii: Markovian rewards. *IEEE Transactions on Automatic Control*, 32(11): 977–982, 1987. doi: 10.1109/TAC.1987.1104485.

T. Anderson. Sequential analysis with delayed observations. *Journal of the American Statistical Association*, 59(308):1006–1015, 1964.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of IEEE 36th Annual Foundations of Computer Science*, pages 322–331, 1995. doi: 10.1109/SFCS.1995.492488.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine learning*, 47(2):235–256, 2002.

V. Avadhanula, R. Colini-Baldeschi, S. Leonardi, K. A. Sankararaman, and O. Schrijvers. Stochastic bandits for multi-platform budget optimization in online advertising. *arXiv preprint arXiv:2103.10246*, 2021.

M. Babaioff, S. Dughmi, R. Kleinberg, and A. Slivkins. Dynamic pricing with limited supply, 2015.

M. Babaioff, L. Blumrosen, S. Dughmi, and Y. Singer. Posting prices with unknown distributions. *ACM Transactions on Economics and Computation (TEAC)*, 5(2):1–20, 2017.

A. Badanidiyuru, R. Kleinberg, and A. Slivkins. Bandits with knapsacks. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 207–216. IEEE, 2013.

L. Bisi, G. De Nittis, F. Trovò, M. Restelli, and N. Gatti. Regret minimization algorithms for the followers behaviour identification in leadership games. In *Proceedings of the Conference on Uncertainty in Artificial (UAI)*, pages 1–10, 2017.

J. Bobadilla Sancho, F. Ortega Requena, A. Hernando Esteban, and J. Bernal Bermúdez. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26, 2012.

B. Brost, R. Mehrotra, and T. Jehan. The music streaming sessions dataset. In *Proceedings of the 2019 Web Conference*. ACM, 2019.

R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 2002.

O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

G. Di Benedetto, V. Bellini, and G. Zappella. A linear bandit for seasonal environments. *arXiv preprint arXiv:2004.13576*, 2020.

M. A. Gael, C. Vernade, A. Carpentier, and M. Valko. Stochastic bandits with arm-dependent delays. In *International Conference on Machine Learning*, pages 3348–3356. PMLR, 2020.

N. Gatti, A. Lazaric, M. Rocco, and F. Trovò. Truthful learning mechanisms for multi-slot sponsored search auctions with externalities. *ARTIF INTELL*, 227:93–139, 2015.

P. Joulani, A. Gyorgy, and C. Szepesvári. Online learning under delayed feedback. In *International Conference on Machine Learning*, pages 1453–1461. PMLR, 2013.

E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *Artificial intelligence and statistics*, pages 592–600. PMLR, 2012.

S. Li, B. Wang, S. Zhang, and W. Chen. Contextual combinatorial cascading bandits. In *International conference on machine learning*, pages 1245–1253. PMLR, 2016.

F. B. Moghaddam and M. Elahi. Cold start solutions for recommendation systems. *Big Data Recommender Systems, Recent Trends and Advances. IET*, 2019.

A. Nuara, F. Trovo, N. Gatti, and M. Restelli. A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

A. Nuara, N. Sosio, F. Trovò, M. C. Zaccardi, N. Gatti, and M. Restelli. Dealing with interdependencies and uncertainty in multi-channel advertising campaigns optimization. In *The World Wide Web Conference*, pages 1376–1386, 2019.

A. Nuara, F. Trovò, N. Gatti, and M. Restelli. Online joint bid/daily budget optimization of internet advertising campaigns. *arXiv preprint arXiv:2003.01452*, 2020.

C. Pike-Burke, S. Agrawal, C. Szepesvari, and S. Grunewalder. Bandits with delayed, aggregated anonymous feedback. In *International Conference on Machine Learning*, pages 4105–4113. PMLR, 2018.

E. Rio et al. On mcdiarmid's concentration inequality. *Electronic Communications in Probability*, 18, 2013.

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

G. Romano, G. Tartaglia, A. Marchesi, and N. Gatti. Online posted pricing with unknown time-discounted valuations. *arXiv preprint arXiv:2012.05774*, 2020.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

F. Trovò, S. Paladino, M. Restelli, and N. Gatti. Improving multi-armed bandit algorithms in online pricing settings. *International Journal of Approximate Reasoning*, 98:196–235, 2018.

F. Trovo, S. Paladino, M. Restelli, and N. Gatti. Sliding-window thompson sampling for non-stationary settings. *Journal of Artificial Intelligence Research*, 68:311–364, 2020.

C. Vernade, O. Cappé, and V. Perchet. Stochastic bandit models for delayed conversions. *arXiv preprint arXiv:1706.09186*, 2017.

C. Vernade, A. Carpentier, T. Lattimore, G. Zappella, B. Ermis, and M. Brueckner. Linear bandits with stochastic delayed feedback. In *International Conference on Machine Learning*, pages 9712–9721. PMLR, 2020a.

C. Vernade, A. Gyorgy, and T. Mann. Non-stationary delayed bandits with intermediate observations. In *International Conference on Machine Learning*, pages 9722–9732. PMLR, 2020b.

J. M. White. *Bandit Algorithms for Website Optimization*, volume 1. O'Reilly, 2012.