University of Alberta

MULTI-ARMED BANDIT PROBLEMS UNDER DELAYED FEEDBACK

by

Pooria Joulani

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

in

Statistical Machine Learning

Department of Computing Science

Abstract

In this thesis, the multi-armed bandit (MAB) problem in online learning is studied, when the feedback information is not observed immediately but rather after arbitrary, unknown, random delays. In the "stochastic" setting when the rewards come from a fixed distribution, an algorithm is given that uses a non-delayed MAB algorithm as a black-box. We also give a method to generalize the theoretical guarantees of non-delayed UCB-type algorithms to the delayed stochastic setting. Assuming the delays are independent of the rewards, we upper bound the penalty in the performance of these algorithms (measured by "regret") by an additive term depending on the delays. When the rewards are chosen in an adversarial manner, we give a black-box style algorithm using multiple instances of a non-delayed adversarial MAB algorithm. Assuming the delays depend only on time, we upper bound the performance penalty of the algorithm by a multiplicative factor depending on the delays.

Acknowledgements

I would like to thank my supervisor, Csaba Szepesvári, for his guidance, patience, support and encouragement. He patiently corrected my mistakes, showed me the way to do research and advised me through the course of our work. This research could not be done without what I learnt from him, and could not be ready on time without his help and support. Also, I would like to thank András György for the many hours of insightful discussions and ideas, as well as reviewing the thesis and providing valuable comments. He was like a co-supervisor for me, and this thesis owes a lot to him.

I would also like to thank the members of my M.Sc. committee, Dr. Majid Khabbazian and Dr. Martin Mueller, for their constructive comments as well as their cooperation and help. Their suggestions were very valuable for improving my thesis.

The subject of my M.Sc. thesis originated in discussions with Dávid Pál and Lihong Li. I would like to thank both of them, especially Dávid for his enthusiastic and encouraging attitude. Dávid was also my co-supervisor initially. During the course of research, I benefited from insightful discussions with Yevgeny Seldin and Lev Reyzin. I would like to thank them for their time and ideas. It is also a pleasure to thank my friends in the RLAI lab at University of Alberta, especially Yasin Abbasi-Yadkori and Gábor Bartók. I learnt a lot from you!

I will never be able to express my deep gratitude to my parents. Their encouragement have always been with me during my studies, and their supports made it possible for me to attend and complete this M.Sc. program. My deepest loves go to my beloved wife. This dissertation would not exist if it was not for her love, support, and the nice and loving environment that she creates at home.

Table of Contents

1 Introduction **1**
1.1 Outline of the thesis . . . 2
1.2 Notation . . . 2
1.3 The multi-armed bandit problem . . . 3
1.3.1 Stochastic multi-armed bandits . . . 3
1.4 Multi-armed bandits with delayed feedback . . . 4
1.5 Performance measures . . . 5

2 Related work **8**
2.1 Non-delayed multi-armed bandit algorithms . . . 8
2.1.1 Stochastic setting . . . 8
2.1.2 Adversarial setting . . . 10
2.2 Related work on delayed online learning . . . 10
2.2.1 Bayesian setting . . . 10
2.2.2 Full-information online learning . . . 11
2.2.3 Label prediction . . . 12
2.2.4 Stochastic optimization . . . 13
2.2.5 Bandit feedback . . . 14
2.3 Relation to parallel and distributed computing . . . 15
2.4 Contributions . . . 16

3 Adversarial multi-armed bandits **18**
3.1 Interface of a non-delayed MAB algorithm . . . 18
3.2 Black-box use of non-delayed algorithms . . . 19
3.2.1 Regret analysis: the case of time-dependent delays . . . 20
3.3 A note on optimality . . . 23
3.4 Time and memory usage of ABBD . . . 23
3.4.1 Why black-box? . . . 24

4 Stochastic multi-armed bandits **25**
4.1 Properties of the stochastic delayed MAB problem . . . 25
4.2 Black-box use of non-delayed stochastic MAB algorithms . . . 27
4.2.1 Regret analysis . . . 28
4.2.2 Time and memory usage of SBBD . . . 31
4.3 Modification of non-delayed algorithms . . . 31
4.3.1 UCB-type algorithms in delayed environments . . . 32
4.3.2 UCB1 under delayed feedback . . . 34
4.3.3 KL-UCB under delayed feedback . . . 36
4.4 Lower bounds for fixed delays . . . 39

5 Experiments **41**
5.1 General setup . . . 41
5.2 Regret penalty due to delays . . . 42
5.3 Effect of delay distributions . . . 43
5.4 Black-box vs. white-box . . . 44

6 Conclusion **55**
6.1 Future work . . . 56

Bibliography **57**

List of Tables

2.1 Summary of work on delayed online learning. A $*$ indicates a multiplicative penalty in the regret, while a $+$ shows an additive penalty. L denotes a matching lower bound. G is the abbreviation for *gap*, the number of consecutive time steps the agent does not get any feedback. I denotes the area for which there is an improved result in the thesis, together with the relevant theorem numbers. The term τ_{const} indicates that the results are for constant delays only. For the work of Desautels et al. (2012), C_1 and C_2 are positive constants, with $C_1 > 1$, and B denotes the upper bound on delays. In that case, the regret grows by the multiplicative constant C_1 independent of delays, as well as an additive term depending on delays. 11

List of Figures

1.1 The multi-armed bandit game. 3
1.2 The multi-armed bandit game with delayed feedback. 5

2.1 An example of parallel online learning. Core 1 makes prediction using the learned parameters in the shared memory, while cores 2 through 4 make updates using the resulting feedback. During the time an update is done, multiple predictions must be made, and thus the predicting core uses parameters that are not updated with the feedback from the most recent prediction. 16

4.1 The SBBD algorithm. The algorithm uses a non-delayed stochastic MAB algorithm (BASE) as a black-box, and simulates a non-delayed environment for it. When the rewards of an action are delayed, the SBBD algorithm keeps choosing the last action chosen by the black-box algorithm. The rewards that arrive later are buffered in the queues, and given to the black-box algorithm BASE when it chooses that action again. 29

5.1 Regret of Delayed-UCB1 for different constant delays. 42
5.2 Regret of Delayed-UCB1 for different constant delays. The regret has stabilized at time $n = 1000$ for delays between 50 and 100. 43
5.3 Regret of Delayed-UCB1 on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean. 45
5.4 Regret of Delayed-KL-UCB on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean. 46
5.5 Regret of SBBD, with UCB1 as base algorithm, on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean. . 47
5.6 Regret of SBBD, with KL-UCB as base algorithm, on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean. 48
5.7 The growth in the regret of UCB1 vs KL-UCB at time step $n = 1000$. Error bars are two times the standard error in mean. KL-UCB seems to be more sensitive to delays, though its total regret is always significantly below that of UCB1. 49
5.8 Regret of Delayed-UCB1 vs SBBD, with UCB1 as base algorithm, as a function of time. The regret curves of the two algorithms are closely matching each othe except for the case of constant delays, when the regret of SBBD(UCB1) is first larger than that of UCB1, but then the ordering swaps and eventually the curves converge. . . . 50
5.9 Regret of Delayed-UCB1 vs SBBD, with UCB1 as base algorithm, at time step $n = 1000$. Error bars are two times the standard error in mean. The regret of the black-box SBBD(UCB1) algorithm is slightly smaller, but this is not statistically significant. 51
5.10 Regret of Delayed-KL-UCB vs SBBD, with KL-UCB as base algorithm, as a function of time. As opposed to UCB1, the regret of the black-box method, SBBD(KL-UCB) is at times significantly different with the regret of Delayed-KL-UCB. 52
5.11 Regret of Delayed-KL-UCB vs SBBD, with KL-UCB as base algorithm, at time step $n = 1000$. Error bars are two times the standard error in mean. As in the case of UCB1, the regret of the black-box algorithm, SBBD(KL-UCB) is slightly better than the regret of Delayed-KL-UCB, but again the difference is not statistically significant. 53
5.12 Memory usage of the buffers of SBBD under four different delay distributions, run separately with KL-UCB or UCB1 as the base algorithm. After an initial transient phase, the memory usage stabilizes. 54

Chapter 1

Introduction

Online decision making problems arise in different context, such as news article recommendation (Li et al., 2010), web content optimization (Agarwal et al., 2009), experiment design (Robbins, 1952), or adaptive pre-fetching in computer memory architectures (Warmuth et al., 2011; Weinberger and Ordentlich, 2002). In this type of problems, an *agent* (or *algorithm*) sequentially chooses actions and receives feedback about them, helping it to choose the next actions in a better way so as to maximize an objective function.

More precisely, consider an online decision making problem with discrete time steps $t = 1, 2, \ldots,$ and a set of actions $\mathcal{K}$. An agent acting in this environment has to choose, at each time step, an action from the set of actions, which results in a reward (a real number) associated with taking that action at that time step. The goal of the agent is to accumulate as much reward as possible. To achieve this goal, the agent uses the information about the rewards of its actions, which is fed back to it by the environment.

The agent's ability and performance in learning to make good actions changes with the type and amount of the feedback information that it is provided. In a *full-information* feedback setting, at the end of each time step the agent observes the rewards of all the actions it could have chosen in that time step. In a *bandit* feedback setting, at the end of each time step the agent only observes the reward of its own action in that time step, and not the reward of the other actions it could have chosen.

In studying the online learning problems described above, it is usually assumed that the feedback related to each of the agent's actions is provided to the agent immediately after the action has taken place. In many applications, however, the feedback from agent's action is not ready immediately, but rather after some delay. For example, online ads must be shown continuously while the feedback from the earlier ads (i.e., whether the user has clicked on the ad or not) is not received immediately, or is received in an aggregated fashion at the end of each day due to the complex distributed nature of ad-serving systems. Similarly, in parallel computing systems, the results of actions at different nodes are communicated and updated with some delay. In designing medical experiments, while waiting for the result of the treatment of one patient, other patients arrive that must be treated.

This thesis studies online decision making for these types of situations which involve delayed feedback. We design and study online learning algorithms under delayed feedback, and analyze the performance penalty due to delays. We study a special case of the general problem described above, in which the set of actions is $\{1, \dots, K\}$ and only bandit information is available. This is called the Multi-Armed Bandit (MAB) problem. The precise problem definition is given in the coming sections.

1.1 Outline of the thesis

This chapter captures the notations and definitions that will be used through the rest of the thesis. After defining the mathematical notation that we use, we give formal definitions and protocols for the multi-armed bandit settings that we study, as well as performance measures for the algorithms in these settings. Then, in Section 1.4, we formalize and describe our model of delayed feedback.

In the chapters that follow, we first review some of the algorithms for the multi-armed bandit problem, as well as related work in online learning with delayed feedback. Then we will separately treat two variants of the MAB problem, and give two types of algorithms for them. The first type of algorithms receive non-delayed algorithms as a black-box, and utilize their power to address the delayed-feedback problem. The second type of the algorithms are derived by generalizing the non-delayed algorithms so that they handle the delayed case as well. Under practical assumptions on the delays, we analyze the performance of the algorithms provided, and upper bound the penalty that the performance incurs due to the presence of delays. Finally, we test our algorithms in practice using a set of experiments with different delay distributions.

1.2 Notation

Throughout the thesis, we will use the following notation. We use capital letter to denote random quantities. The symbol $\mathbb{I}\{A\}$ denotes the indicator for the event A, which is 1 when A happens and 0 otherwise. $\mathbb{E}[X]$ is the expected value of the random variable X, and $\mathbb{P}\{A\} = \mathbb{E}[\mathbb{I}\{A\}]$ is the probability of the event A. By $\sup_{a \in \mathcal{A}} f(a)$ and $\inf_{a \in \mathcal{A}} f(a)$ we mean the supremum and the infimum, respectively, of the function $f(.)$ over the set $\mathcal{A}$. The set of natural numbers $\{1, 2, \dots\}$ is denoted by $\mathbb{N}$, and $\mathbb{R}$ is the set of real numbers. To denote a sequence, we use the notation $\{X_t\}_{t \in \mathcal{A}}$, which is the sequence of all X_t values when the index t is chosen from the set $\mathcal{A}$. By $\max_{a \in \mathcal{A}} f(a)$ and $\underset{a \in \mathcal{A}}{\operatorname{argmax}} f(a)$ we denote, respectively, the maximum value of function $f(.)$ over the set $\mathcal{A}$ and the value of a parameter $a \in \mathcal{A}$ that achieves that maximum (in case of multiple optimal parameters, we explicitly state how to break the ties). The average of s values $Y_1, Y_2, \dots, Y_s$ is denoted by $\bar{Y}_s = \frac{1}{s} \sum_{t=1}^{s} Y_t$. We use $\log n$ to denote the natural logarithm of real number n. The modulus operator $x \bmod y$ gives the positive remainder of dividing integer x by integer y. We use $O(f(t))$ and $\Omega(f(t))$ to denote the Big-Oh and Omega growth rate families of function f, and

Repeat: At each time step $t = 1, 2, \ldots$:

1. The environment chooses the rewards $x_{i,t} \in [0, 1]$ for $1 \leq i \leq K$.
2. The agent chooses an action I_t, and sends it to the environment.
3. The agent observes the reward $x_{I_t,t}$ that it has enjoyed for choosing action I_t.

Goal: Maximize the cumulative reward $\sum_{t=1}^{n} x_{I_t,t}$ by time step n.

Figure 1.1: The multi-armed bandit game.

$\widetilde{O}(f(t)) = O\left(f(t)\log(f(t))\right)$ for $f(t) > 1$.

1.3 The multi-armed bandit problem

In this section we precisely define the non-delayed MAB problems. As described above, we have an online decision making problem with discrete time steps. There is an agent who sequentially chooses actions from the set of K possible actions $\{1, \ldots, K\}$, and an environment that assigns rewards to the actions and provides the agent with bandit-type feedback.

Figure 1.1 shows the protocol for the MAB game. At each time step $t \in \mathbb{N}$, the environment chooses a vector of K rewards x_t, whose components we denote by $x_{i,t}, 1 \leq i \leq K$. The reward $x_{i,t}$ of each action i is assumed to be in $[0, 1]$. At the same time, using the feedback from its previous interactions with the environment, the agent chooses an action I_t which is *played* in that time step. Note that the choice $I_t \in \{1, \ldots, K\}$ is a random quantity because the agent can use randomization to pick its action. The agent then *observes* the reward of its choice, which is $x_{I_t,t}$, at the end of time step t. The agent, however, does not observe the rewards of the other possible actions $x_{j,t}, j \neq I_t$, as the environment provides only bandit feedback. The goal of the agent is to maximize the cumulative reward of its actions after n time steps, i.e., $\sum_{t=1}^{n} x_{I_t,t}$. The number of time steps n is called the *time horizon*.

Note that in the general form of the MAB game described above, no specific assumptions are made about the rewards, except being bounded in $[0, 1]$. This means that the agent has to achieve a good performance under *any* reward sequence. We will discuss this more formally in Section 1.5, where we define what we mean by good performance. However, additional assumptions about the rewards can be made which lead to interesting variants of the MAB problem. One of these variants is the non-parametric stochastic MAB problem or, in short, stochastic MAB problem, which we define in the next section.

1.3.1 Stochastic multi-armed bandits

The stochastic MAB problem was first studied by Robbins (1952). In this setting, we assume that the reward of each action i at each time step is drawn from a fixed distribution ν_i on $[0, 1]$, independently from all other rewards of that action and from the rewards of other actions. More precisely, for

$1 \leq i \leq K$, let $X_{i,t}$ denote the reward of action i at time step t, which is now a random quantity. The rewards $\{X_{i,t}\}_{t\in\mathbb{N}}$ are assumed to form an i.i.d sequence from the distribution ν_i and, furthermore, the K sequences $\{X_{i,t}\}_{t\in\mathbb{N}}, 1 \leq i \leq K$, are assumed to be independent of each other. For all $1 \leq i \leq K$, the distribution ν_i, whose mean we denote by μ_i, is not known to the agent.

Both the stochastic and the adversarial MAB settings have been studied thoroughly in the literature. In Section 2.1, we summarize some of the existing results about these two settings which we will refer to throughout the thesis. We will also review some prior algorithms that we will use later in solving the delayed problems.

1.4 Multi-armed bandits with delayed feedback

The problem we study in this thesis is the multi-armed bandit problem where there is some delay before the agent can observe the reward of its action. This means that the agent will not always see the reward of its action at the end of the time step in which the action is chosen, but rather some (unknown) number of time steps later.

More precisely, in the delayed-feedback MAB problem, for each action $1 \leq i \leq K$ and each time step $t \in \mathbb{N}$, there is a non-negative delay $\tau_{i,t}$ associated with the reward $x_{i,t}$ of choosing action i at time step t. This means that if action i is chosen at time step t, its reward will be observed not at the end of time step t, but rather at the end of time step $t + \tau_{i,t}$. We allow the delays to be random in the general case. However, we do not make any additional assumptions about their joint distribution. In particular, the delays can all have the same fixed values, or come from the same fixed and potentially unbounded distribution, or can be infinite (in which case the reward will never be observed). As a special case, when all the delays are zero we recover the original non-delayed MAB problem. The exception to this is the stochastic MAB problem where, as we will see in Section 4.1, we require the delays to be independent of the rewards. For the adversarial setting, we require the delays at each time step to be the same for different actions, that is, we have a single sequence $\{\tau_t\}_{t\in\mathbb{N}}$ which is used for all the actions. This, however, does not prevent the delay sequence from having any desired randomness. This is a common situation, e.g., in network-based applications, when the delay depends on the congestion at the current time but not on the actual data that is being transmitted.

Since the delays can have arbitrary values, in general it may happen that the rewards of the actions $I_t = i$ and $I_{t'} = j$ at time steps t and t' have to be observed at the end of the same time step $t + \tau_{i,t} = t' + \tau_{j,t'}$. Therefore, in the delayed MAB problem the agent observes *sets of rewards* for each action (and not only for the action chosen). The set of rewards of action i which are observed at the end of time step t is denoted by $\mathcal{X}_{i,t} = \{(s, x_{i,s}) \mid 1 \leq s \leq t,\ I_s = i,\ s + \tau_{i,s} = t\}$. Note that the agent needs to see the *time stamp* s in addition to the reward value $x_{i,s}$ because the rewards can get reordered due to delays. However, under the assumptions of the stochastic multi-armed bandit problem, this reordering is not important, as we will see in Section 4.1. The detailed protocol for

Repeat: At each time step $t = 1, 2, \ldots$:

1. The environment chooses the rewards $x_{i,t} \in [0, 1]$ for $1 \leq i \leq K$.
2. The agent chooses an action I_t, and sends it to the environment.
3. The environment adds the reward $x_{I_t,t}$ to the set $\mathcal{X}_{I_t,t+\tau_{I_t,t}}$ so that it is observed by the agent at the end of time step $t + \tau_{I_t,t}$.
4. The agent observes the set $\mathcal{X}_{i,t}$ for $1 \leq i \leq K$

Goal: Maximize the cumulative reward $\sum_{t=1}^{n} x_{I_t,t}$ by time step n.

Figure 1.2: The multi-armed bandit game with delayed feedback.

the multi-armed bandit game with delayed feedback is given in Figure 1.2.

1.5 Performance measures

As noted before, the goal of the learning agent is to maximize its accumulated reward. In this section we describe the way we evaluate the performance of the agent in achieving this goal.

In the general MAB problem as described above, we can not just evaluate the performance of the agent by the magnitude of the reward it accumulates. The reason is that the *amount* of the reward that an agent can collect depends on the problem: if the rewards are all zeros, then no algorithm can achieve any reward, and if they are all one, then any algorithm achieves a cumulative reward of n. Therefore, we need to measure the performance in such a way that describes *how effective the agent is in extracting the reward that is in the sequence on which it runs*. In other words, a good agent is one which uses the properties of the reward sequence as best as possible, such that it does not *regret* its choices when the game is finished. This can be achieved by measuring the performance of an agent by comparing its cumulative reward against that of some *reference* strategy, such that the reference makes a good use of the reward content of the sequence of rewards.

The standard strategy that is usually used as the reference chooses the best constant action in hindsight. More precisely, consider an action which if played up to round n, gives the largest cumulative reward. We compare the cumulative reward of the agent after n time steps with the cumulative reward of such an action. This difference is called the *regret* of the agent after n time steps, which is defined as

$$R_n = \max_{1 \leq i \leq K} \left\{ \sum_{t=1}^{n} x_{i,t} - \sum_{t=1}^{n} x_{I_t,t} \right\}. \tag{1.1}$$

Hence, a good agent is one which achieves a small regret. Note that competing with the best action in hindsight is hard enough. In particular, we will not be able to compete with the best strategy that is allowed to change its choice at each time step, because the set of references will become exponentially large (in the number of actions). However, we are still able to compete with strategies that are allowed to change their actions a constant number of times, with algorithms that are in nature the same as the ones used for competing with constant-action strategies. This is called the tracking problem, which we do not consider in this thesis. Analysis of the tracking problem can be found in Section 5.2 of the book of Cesa-Bianchi and Lugosi (2006).

The use of the cumulative reward of a constant action has an even more meaningful intuitive interpretation in the stochastic setting. In this case, since the rewards are random, in the long run the best action in hindsight becomes the action with the largest expected reward $i^* = \underset{1 \le i \le K}{\operatorname{argmax}} \mu_i$ (where in the case of multiple optimal actions, we let i^* be the smallest index among them). Accordingly, in the stochastic setting, as common in the literature we redefine the regret to be

$$R_n = \sum_{t=1}^{n} X_{i^*,t} - \sum_{t=1}^{n} X_{I_t,t}. \tag{1.2}$$

The regret as defined by (1.1) and (1.2) is a random quantity, since the agent (and the environment) can use randomization in their choices. We are therefore often more interested to know the expected value of the regret, which shows the average performance of the algorithm. In fact, in the adversarial setting, randomization is a very useful tool for the agent because if $K \ge 2$ and the agent is deterministic, then there is always a sequence of reward on which the agent suffers a regret as large as $\Omega(n)$. This is because if the agent is deterministic, the environment can always compute the choice of the agent for each possible history. The environment can thus set a reward of 0 for the agents choices and 1 for the other actions, making the regret to be at least $n/2$.

In the adversarial setting, to compare the performance of different algorithms it is more appropriate to measure their *worst-case* regret. Let $\mathcal{P}$ denote the set of all $K \times n$ sequences over $[0, 1]$ for all $n \in \mathbb{N}$, and $\mathcal{A}$ be the set of all MAB algorithms. Denote by $R_n^{A,P}$ the regret of algorithm $A \in \mathcal{A}$ on the sequence $P \in \mathcal{P}$ after n rounds. Then

$$\sup_{P \in \mathcal{P}} R_n^{A,P}$$

and

$$\sup_{P \in \mathcal{P}} \mathbb{E}\left[R_n^{A,P}\right]$$

are the worst-case regret and the worst-case expected regret, respectively, of algorithm A on *any sequence*. A better agent is therefore one which achieves a lower worst-case regret.

To show that the agent controls its regret and therefore has a good performance, we usually give upper bounds on the expected value of the regret. We can also provide an upper bound over the regret that holds with high probability, which is a stronger result in the sense that it holds for all values of the regret, not just the expected value. We are interested in bounds that scale sub-linearly with the time horizon n, so that the average regret per time step, $\frac{1}{n}R_n$, goes to zero as n grows. Note in particular that the growth rate of the regret divided by n is a natural measure of learning rate of the agent.

To see how fast the average per-time-step regret of an agent can approach zero, we usually start by showing lower bounds for the *minimax* regret,

$$\inf_{A \in \mathcal{A}} \sup_{P \in \mathcal{P}} \mathbb{E}\left[R_n^{A,P}\right] \tag{1.3}$$

which is the smallest worst-case expected regret *any agent* can achieve and is an inherent characteristic of the learning problem. If the lower bound on minimax regret matches, up to a constant factor, the upper bound on the worst-case regret of an agent, it means that the agent is learning with a rate similar to that of the best possible agent, and is in this sense *optimal*.

Chapter 2

Related work

2.1 Non-delayed multi-armed bandit algorithms

In this section we study some of the algorithms devised for the *non-delayed* MAB problem, upon which we build our algorithms for the delayed setting in the coming chapters.

The adversarial and stochastic MAB problems have been thoroughly studied in the literature. Auer et al. (2002) study the stochastic MAB problem presented in section 1.3.1, and their work is followed by a sequence of inspired algorithms, such as the works of Audibert et al. (2009) and Garivier and Cappé (2011), which illustrate and improve on different aspects of the problem. The adversarial MAB problem has been studied by Auer et al. (2003) as well as Audibert and Bubeck (2010), amongst others. A recent review of the MAB literature is due to Bubeck and Cesa-Bianchi (2012). In the next sections, we will review some of the algorithms and results of these works which are used throughout the thesis.

2.1.1 Stochastic setting

A large family of algorithms that work in the stochastic MAB setting are based on using *upper confidence bounds* (UCBs). In this thesis we refer to this family as UCB-type algorithms. These algorithms work by following the *optimism in the face of uncertainty principle*: they maintain optimistic estimates of the rewards of each action and in each time step choose an action for which this optimistic estimate is the largest. Being optimistic causes the algorithm to constantly *explore* different actions while trying to *exploit* the current observed optimal choice, thus avoiding early commitment to a choice that may later turn out to be suboptimal.

More precisely, UCB-type algorithms maintain an upper confidence bound $B_{i,s,t}$ for each action i, which depends on the current time step t, the number of observed rewards s, and the observed reward values $Y_{i,1}, Y_{i,2}, \dots, Y_{i,s}$. Each UCB-type algorithm specifies its own form for $B_{i,s,t}$. These bounds are the optimistic estimates that we mentioned and are designed such that the expected reward of the action is below its bound with high probability. At each time step t, these algorithms choose the action given by

$$I_t = \underset{1 \le i \le K}{\operatorname{argmax}}\ B_{i,T_i(t-1),t}$$

where $T_i(t)$ is the number of times action i has been chosen by time t.

Here we review two UCB-type algorithms that we will revisit in the thesis. The first algorithm, called UCB1, is due to Auer et al. (2002). UCB1 uses upper confidence bounds of the form $B_{i,s,t} = \bar{Y}_{i,s} + \sqrt{\frac{2\log t}{s}}$, where $\bar{Y} = \frac{1}{s}\sum_{t=1}^{s} Y_{i,t}$ is the average observed reward. In the original UCB1 paper (Auer et al., 2002), the authors show the following theorem.

Theorem 2.1. *The expected regret of $UCB1$ after n time steps is bounded by*

$$\mathbb{E}\left[R_n\right] \le \sum_{\Delta_i > 0} \left(\frac{8 \log n}{\Delta_i} + \left(1 + \frac{\Pi^2}{3}\right) \Delta_i \right) \tag{2.1}$$

where $\Delta_i = \max_{1 \le j \le K} \mu_j - \mu_i$ is the difference between the expected reward of action i and that of the action with the largest expected reward.

Recently, a new UCB-type algorithm, called KL-UCB, was proposed by Garivier and Cappé (2011) (and independently also by Maillard et al. (2011)), which is asymptotically optimal up to additive sublogarithmic terms and has also been found to work well in simulations. KL-UCB uses upper confidence bounds of the form

$$B_{i,s,t} = \max\left\{ q \in [\bar{Y}_{i,s}, 1] : s\mathbf{d}(\bar{Y}_{i,s}, q) \le \log(t) + 3\log(\log(t)) \right\}$$

where $\mathbf{d}(p,q) = p\log(p/q) + (1-p)\log((1-p)/(1-q))$ is the KL-divergence of two Bernoulli random variables with parameters p and q [1]. The authors show that the KL-UCB algorithm enjoys the following regret guarantee:

Theorem 2.2. *Let μ_i denote the expected reward of action i, and i^* denote an action with the largest expected reward. Then there exists a constant $C_1 \le 7$, as well as functions $C_2(\epsilon) = O(\epsilon^{-2})$ and $0 \le \beta(\epsilon) = O(\epsilon^2)$, such that for any $\epsilon > 0$, the expected regret of the KL-UCB algorithm satisfies*

$$\mathbb{E}\left[R_n\right] \le \sum_{\Delta_i > 0} \Delta_i \left[\frac{\log(n)}{\mathbf{d}(\mu_i, \mu_{i^*})}(1+\epsilon) + C_1 \log(\log n) + \frac{C_2(\epsilon)}{n^{\beta(\epsilon)}} \right]. \tag{2.2}$$

As opposed to UCB1, the computation of the upper confidence bound for KL-UCB must be done numerically, making this algorithm computationally expensive. In Chapter 4, we will extend these two algorithms to work in the delayed setting.

It is noteworthy that the bounds above become vacuous when the gaps Δ_i are very small, e.g., of order $O(1/n)$. However, since the number of time an action is chosen is at most n, we can bound the regret of each action by $n\Delta_i$ as well. We will use this smaller upper bounds in cases with very small gaps.

[1] The Kullback-Leibler(KL) divergence is "a measure of the difficulty of discriminating between" two distributions (Kullback, 1997). Shlens (2007) provides a good tutorial on KL-divergence.

2.1.2 Adversarial setting

The standard algorithm for the adversarial MAB problem is EXP3 by Auer et al. (2003). EXP3 works by maintaining a distribution over actions which is used for picking actions randomly at each time step. The probability of choosing each action is proportional to the exponentiated total reward observed for that action. For the detailed algorithm, please refer to the original paper (Auer et al., 2003).

Recently, a new algorithm called INF (Implicitly Normalized Forecaster) was proposed by Audibert and Bubeck (2010). As opposed to EXP3, this algorithm was shown to be optimal in the sense that it achieves a regret with the same rate of growth as the minimax regret of the adversarial MAB problem. EXP3's regret-bound is larger by a poly-logarithmic factor. Bubeck (2010) shows the following theorem for a special variant of INF.

Theorem 2.3. *In the multi-armed bandit problem with K actions, the expected regret of INF after n time steps satisfies:*

$$\mathbb{E}\left[R_n\right] \leq 12.2\sqrt{nK} \tag{2.3}$$

We will use this theorem in Chapter 3, where we transform the delayed problem to multiple instances of the non-delayed problem, which are then solved by INF.

2.2 Related work on delayed online learning

The problem of online learning with delays has been studied in different contexts and for different types of feedback. In this section we review the related work in this area. A concise summary, together with the contributions of this work, is given in Table 2.1.

2.2.1 Bayesian setting

One of the thoroughly studied settings in the multi-armed bandit literature is the Bayesian setting. In this setting, the problem is to compute an optimal policy (i.e., an optimal action given the history of past observations and actions) so as to maximize the total expected (typically discounted) reward over some finite or infinite horizon assuming that the actions' reward generating distributions are drawn from some known distribution at time $t = 0$, from which time on they are kept fixed. Note that, in contrast to the stochastic MAB problem, this is a purely computational problem. In a seminal paper, Gittins proved that the optimal policy has a special *index*-form, making its computation feasible (Gittins and Jones, 1974).

In the Bayesian MAB setting, the problem of delayed feedback was proposed by Anderson (1964). Suzuki (1966) gives optimal policies for a special case of the problem when the reward distributions are parametric, and Choi and Clark (1970) treat a special case of the problem for determining the probability parameter of a binomial distribution. Recently, Guha et al. (2010) proposed an algorithm that approximates the optimal solution of the problem when the delays are of order

		Stochastic	Adversarial
Full Info	No Side Info	$+\mathbb{E}\left[\tau_t^2\right]$ Agarwal and Duchi (2011)	L $*\tau_{const}$ Agarwal and Duchi (2011) Langford et al. (2009) Weinberger and Ordentlich (2002)
	Side Info	L $+\max G$ Mesterharm (2007)	L $*$ average G Mesterharm (2007)
Bandit Feedback	No Side Info	I [4.2, 4.3, 4.4] $*C_1$ and $+C_2 B\log(B)$ Desautels et al. (2012)	I [3.1] $*\ \tau_{const}$ (Neu et al., 2010)
	Side Info	$+\tau_{const}\sqrt{\log n}$ Dudik et al. (2011)	N/A

Table 2.1: Summary of work on delayed online learning. A $*$ indicates a multiplicative penalty in the regret, while a $+$ shows an additive penalty. L denotes a matching lower bound. G is the abbreviation for *gap*, the number of consecutive time steps the agent does not get any feedback. I denotes the area for which there is an improved result in the thesis, together with the relevant theorem numbers. The term τ_{const} indicates that the results are for constant delays only. For the work of Desautels et al. (2012), C_1 and C_2 are positive constants, with $C_1 > 1$, and B denotes the upper bound on delays. In that case, the regret grows by the multiplicative constant C_1 independent of delays, as well as an additive term depending on delays.

$O(n/\log n)$, where n is the time horizon. Their algorithm considers constant delays δ_i for each action i.

2.2.2 Full-information online learning

The adversarial full information online learning problem is similar to the adversarial MAB problem described in Section 1.3, with the difference that the learner not only observes the reward of its own action, but also the reward of the other actions it could have chosen at that time step. This setting has been deeply studied in the context of prediction with expert advice, e.g., by Cesa-Bianchi and Lugosi (2006).

An early result for learning under delayed feedback in the full information setting is due to Weinberger and Ordentlich (2002). They consider fixed delays and show that a sub-sampling approach gives optimal results in this setting. While they state their problem and results for prediction of individual sequences (which is a less general formulation of the full information feedback), their approach can be used directly for multi-armed bandits; it is, however, limited to constant (or bounded) delays where the delay value or the bound is known a priori. In Section 3.2, we generalize their approach and show that the new version works for unknown, unbounded, and possibly infinite delays. To put our contribution into context, let us now summarize their ideas and algorithm.

The idea of Weinberger and Ordentlich (2002) is that for a fixed delay of τ, once we know the delay, we can *sub-sample*[2] the sequence of reward vectors x_t at a rate $\tau + 1$, and give each of the

[2]By sub-sampling a sequence at a rate d, we mean creating a subsequence of that sequence consisting of members of the

resulting sub-sequences to a non-delayed full-information algorithm. In other words, if we make a prediction at time step t, we will be able to use its reward for decision making at time $t+\tau+1$. Therefore, if we have $\tau+1$ instances of a non-delayed algorithm, and at time step $t \geq 1$ consult algorithm instance $i, 0 \leq i \leq \tau$ such that $t \bmod (\tau+1) = i$, then the algorithm instances will observe no delays and at each time step we know what action to choose. Assuming n divides τ, if R_n^m is the regret of the m-th instance of the non-delayed algorithm on a sequence of length n, I_t is the choice of the algorithm at time step t, $x_{i,t}$ is the reward of choosing action i at time step t, and R_n is the regret of the sub-sampling algorithm under delay τ, then we have:

$$\begin{aligned}
R_n &= \max_{1\leq i\leq K}\left(\sum_{t=1}^{n} x_{i,t} - \sum_{t=1}^{n} x_{I_t,t}\right)\\
&= \max_{1\leq i\leq K}\left(\sum_{m=1}^{\tau+1}\sum_{s=0}^{n/(\tau+1)-1} x_{i,s(\tau+1)+m} - x_{I_{s(\tau+1)+m},s(\tau+1)+m}\right)\\
&\leq \sum_{m=1}^{\tau+1} \max_{1\leq i\leq K} R^m_{n/(\tau+1)}\\
&= \sum_{m=1}^{\tau+1} R^m_{n/(\tau+1)}
\end{aligned}$$

and therefore, the worst-case regret of the sub-sampling algorithm satisfies

$$\sup_{\mathcal{P}} R_n \leq \sum_{m=1}^{\tau+1} \sup_{\mathcal{P}} R^m_{n/(\tau+1)} = (\tau+1)R'_{n/(\tau+1)}$$

where $R'_{n/(\tau+1)}$ is the worst-case regret of the underlying non-delayed algorithm that is used. The authors show a matching lower bound which establishes the optimality of their algorithm for constant delays.

2.2.3 Label prediction

Mesterharm (2005, 2007) investigates another variant of the full information setting, with a different delay model. He considers the label-prediction setting in which the learner observes, at each time step, a side information vector x_t for which it wants to predict a label y_t. The label is given to the agent at the end of the time step (in the non-delayed case), and the agent suffers a loss of 1 if it had made a mistake and a loss of zero if the label is predicted correctly. The sequence of instances and labels can be chosen by an adversary, or randomly from a fixed distribution, and the goal of the learner is to make as few mistakes as possible.

Mesterharm considers an adversarial framework for the delays, in which to a sequence of samples $\{(x_t, y_t)\}_{t\in\mathbb{N}}$ (drawn randomly from a fixed distribution or chosen in an adversarial manner), the adversary assigns a sequence of delays $\{\tau_t\}$ from a multi-set D; the label y_t is observed at time step $t+\tau_t$, i.e., τ_t rounds after side information x_t is observed and prediction is made. The multi-set

sequence that are d indices apart, e.g., the first member, the $(d+1)$-th member, the $(2d+1)$-th member, and so on.

D, used to manage the power of the adversary in choosing the delays, is assumed to have infinitely many elements. Let the number of instances of $i \geq 0$ in D be $d_i \geq 0$. If $d_5 = 10$, for example, the adversary can impose a delay of five rounds on ten of the instances in the sequence of samples.

The worst-case effect of the delays depends on how long the adversary can prevent the agent from observing feedback. This is a property of the multi-set D. Let $F(D, c)$ be the largest total length of c lists of delays, built using members of D, such that the delays in each of the c lists make it impossible to receive any feedback until the end of the list. This means that each delay in each list is at least as large as the remaining length of the list. These lists can then be concatenated together and used as the sequence of delays that the agent observes. This quantity is used in the upper and lower bounds for the label prediction problem.

Mesterharm gives black-box style algorithms which transform non-delayed label prediction algorithms to delayed ones. If the samples are chosen by an adversary, Mesterharm shows that the expected number of mistakes in the delayed case incurs a multiplicative penalty that depends on the average number of rounds at which no feedback is observed. We will see a multiplicative regret bound of the same nature in Section 3.2 for the adversarial MAB problem.

When the samples are drawn from a distribution, the situation is better, and an additive penalty of $F(D, 1)$ on the expected number of mistakes was shown by Mesterharm. Note that $F(D, 1)$ is the maximum number of consecutive time steps when no feedback is received. As we will see in Section 4.2.1, in the stochastic MAB problem as well, we can show a penalty on the regret that is additive in the delays.

Mesterharm (2007) also gives lower bounds for the two cases described above. For the adversarial case a multiplicative lower bound is shown, which uses the same ideas as the one by Weinberger and Ordentlich (2002). We will discuss this idea later in Section 3.3. For the stochastic case, a lower bound of $F(D, 1)/2$ on the additive penalty is given when the labels are binary.

2.2.4 Stochastic optimization

The effect of delayed feedback has also been studied in stochastic optimization (Langford et al., 2009; Agarwal and Duchi, 2011). In stochastic optimization, the aim is to minimize a convex function $f(x)$ over a closed convex set $\mathcal{X}$, using a sequence of random functions $\{f_t\}_{t \in \mathbb{N}}$. The functions $f_t : \mathcal{X} \mapsto \mathbb{R}$ are assumed to be drawn from a distribution P such that for any $x \in \mathcal{X}$, $\mathbb{E}[f_t(x)] = f(x)$. Standard algorithms for this setting (e.g., the algorithm of Nemirovsky and Yudin (1983)) are based on updating a solution candidate x_t at time step t using the gradient of function f_t. The sequence of parameters $\{x_t\}_{t \in \mathbb{N}}$ generated by these algorithms can then be used, e.g., by averaging, to get an estimate of the optimal solution. The performance measure used in this context is *risk*, which is the expected difference $\mathbb{E}[f(\hat{x}(T))] - f(x^*)$ where $\hat{x}(T) = \frac{1}{T}\sum_{t=1}^{T} x_t$ is the average of the updated parameters after T runs and x^* is the optimal solution which minimizes f.

In the delayed-feedback version of stochastic optimization, the gradient g_t of the function f_t

is assumed to be observed after a delay τ_t. In this setting, Langford et al. (2009) show that in the general case, under a constant delay τ the penalty in the performance of the algorithm is bounded from above by a multiplicative factor of order $\sqrt{\tau}$. Agarwal and Duchi (2011) show that using smoothness assumptions on the gradients, it is possible to prove an additive penalty. More precisely, they show that if $\mathbb{E}\left[\tau_t^2\right] \leq B^2$ for some real number B, then the penalty in the risk is additive and is of order $O(B^2)$.

2.2.5 Bandit feedback

Learning under delayed feedback has also been studied for contextual bandits. A contextual bandit game is similar to the usual multi-armed bandit problem defined in Section 1.3, except that at each time step there is some side information $x_t \in \mathcal{X}$ which helps the agent choose its actions and the agent's regret is measured relative to the performance of the best mapping $\pi : \mathcal{X} \mapsto \{1, \ldots, K\}$ from a fixed set Π, which is known to the agent. In detail, the agent interacts with its environment at time step t as follows:

1. The environment chooses the side information x_t and a $K \times 1$ vector r_t of rewards, and presents x_t to the agent.
2. The agent chooses an action I_t from the set of actions $\{1, \ldots, K\}$.
3. The agent observes the reward $r_{I_t,t}$ of its choice.

Similar to the regular MAB problem, the information (x, r) can be chosen in an adversarial manner or can come from a distribution. In the stochastic setting, and when the information is delayed by a fixed delay of τ, Dudik et al. (2011) show that the penalty in the regret due to the delays is an additive term of $O(\sqrt{N \log(Kn^2)})\tau$ where N is the size of the set of policies Π. This gives a direct result for the stochastic MAB problem, as the MAB problem is a special case of the contextual bandit problem when $|\mathcal{X}| = 1$ and $\Pi = \{\pi_1, \ldots, \pi_K\}, \pi_k(x) = k, 1 \leq k \leq K$. Thus, from the results of Dudik et al. (2011), when substituting N with K, we get an additive term of $O\left(\sqrt{Klog(Kn^2)}\right)\tau$ for the stochastic MAB problem. In Theorems 4.2, 4.3 and 4.4, we will show additive penalty terms that, unlike the result of Dudik et al. (2011), do not depend on the time horizon n when the delay is constant. It remains for future work to see whether such results can be extended to the contextual bandit case.

Another work that concerns the bandit setting with delayed feedback is by György et al. (2007). The authors consider the setting in which an agent, observing a side information, at each time step chooses one of the K actions. The agent then has to wait for some time until the reward of this action arrives. The setting, however, is continuous time, that is, there are no discrete time steps: the agent can start with the next action as soon as the reward of the previous action arrives. Due to the continuous time, the goal of the agent is not just to maximize its reward after n actions; it is rather to maximize its reward *per unit of time*. This means choosing an action with a high expected reward but a very long delay is not the best choice, as there may be actions with lower but immediate expected rewards, which result in a higher reward rate. Note that the rewards depend on the current side information, otherwise the problem will be equivalent to the regular MAB problem where the expected rewards are normalized by expected delays. The authors propose a variant of the UCB1 algorithm (Auer et al., 2002) for the aforementioned setting. Their algorithm achieves a regret that grows with a rate arbitrarily close to the optimal rate of $O(\log n)$.

The problem of delayed feedback has also been studied for Gaussian process bandit optimization (Desautels et al., 2012). In this stochastic setting, for delays bounded by a known upper bound B, the authors derive an upper bound on the regret of the form

$$R_n \leq C_1 R'_n + C_2 B \log(B)$$

where R'_n is the regret without delays and C_1 and C_2 are positive constants, with $C_1 > 1$. As a special case, this can be applied to the stochastic MAB problem. In that setting, however, our results provide improved bounds as mentioned before.

A related work for the bandit feedback is due to Neu et al. (2010). In this work, the authors study the problem of online learning in environments with stochastic controlled Markovian dynamics, where the reward function is chosen in an adversarial manner. While not directly tackling the delayed problem, as a byproduct of their analysis, the authors show that the so-called EXP3 algorithm changes its policy slowly over time. Using this fact, the authors are able to bound the regret of EXP3 when its actions are implemented with a delay. The resulting regret bound shows a multiplicative penalty depending on the constant delay. In Theorem 3.1, we show an improved result for the adversarial MAB setting which allows non-constant delays.

2.3 Relation to parallel and distributed computing

The problem of delayed feedback is directly related to parallel and distributed computation. In parallel and distributed systems, there are many computing nodes performing computations to solve a specific problem. These computing nodes can be different cores of a single processor, multiple processors in a computer, or multiple computers connected across a network. Bertsekas and Tsitsiklis (1989) and Leopold (2001) discuss the theory and models of parallel and distributed computing.

The online learning framework described in Chapter 1 is inherently sequential, working on only one piece of data at a time. However, when there is a large amount of data available or when the required time between two predictions is smaller than the time it takes to learn from the data, it is desirable to parallelize the learning task. Due to the inherent latency in communication, excessive synchronization between the computing nodes is not possible in such an architecture. Therefore, at each time some of the computing nodes will be working on data that is not always current. In such cases, having delay-tolerant online learning algorithms facilitates parallelizing the learning task.

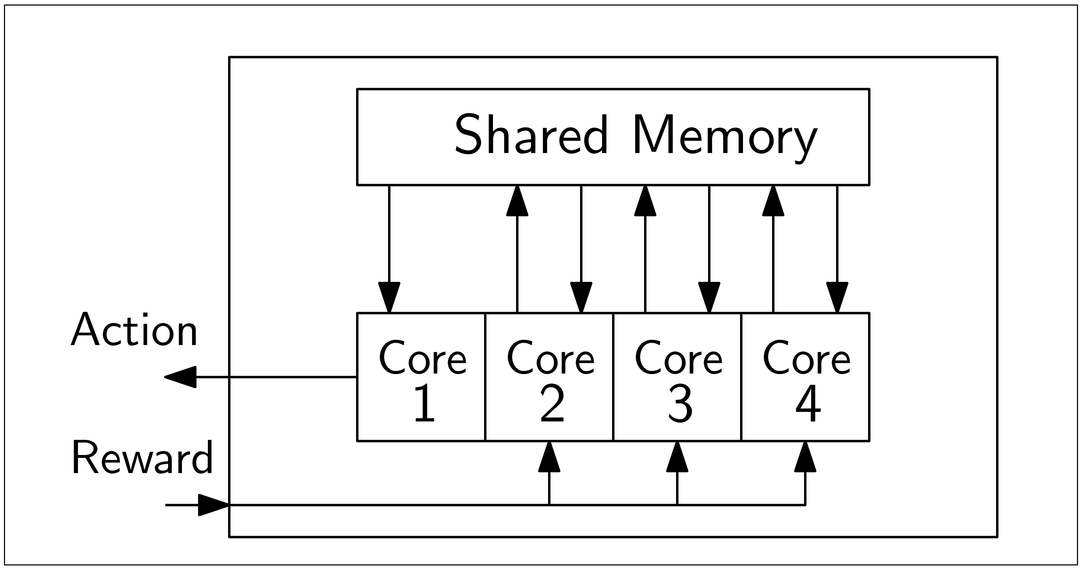


Figure 2.1: An example of parallel online learning. Core 1 makes prediction using the learned parameters in the shared memory, while cores 2 through 4 make updates using the resulting feedback. During the time an update is done, multiple predictions must be made, and thus the predicting core uses parameters that are not updated with the feedback from the most recent prediction.

For example, consider a processor with four cores working on an online learning problem, with a shared set of parameters in the memory which is used for making prediction. Assume that the time required for updating the parameters using the feedback is larger than the time required to make a prediction using that parameters, and therefore one core is dedicated to making predictions while the other three cores update the parameters using the feedback from those predictions. This is shown in Figure 2.1. If an update is nearly three times slower than a prediction, in the time core 1 makes three predictions, I_t, I_{t+1} and I_{t+2}, the parameters are updated using the feedback of the prediction I_{t-1} which had arrived at the end of time step $t-1$ and was processed by, e.g., core 4. Therefore, the predicting core is working under delayed feedback, with parameters that are updated only with the feedback of actions made three time steps ago.

This situation arises in other parallel architectures, as well. Applying delay-tolerant online learning algorithms for some of these different architectures has been studied in the literature on delayed stochastic optimization, such as the studies by Agarwal and Duchi (2011) and Langford et al. (2009).

2.4 Contributions

We extend previous works to the multi-armed bandit setting and improve the previous results. Firstly, unlike most previous work, we allow the delays to be random, and only make mild assumptions on the delays which were discussed in Section 1.4. This is in fact an important improvement over the assumption that the delay is constant, since most of the phenomena which result from delays, e.g., the reordering of the rewards, do not happen when the delay is constant. We also believe that our assumptions are not a major limitation in practice, because many real applications, e.g., parallel and distributed computation, conform to these assumptions. Furthermore, our algorithms do not require knowledge of the delays or any specific bound on them or on the expected delay.

Our first contribution is the generalization of the algorithm of Weinberger and Ordentlich (2002) to the bandit setting with arbitrary delays depending only on time. Furthermore, while our algorithm is analyzed and discussed for the MAB problem, the same idea can be applied to other areas of online learning such as the full-information or the contextual bandit game.

While most of the additive regret penalties under delayed feedback have been shown for the full information setting, in Theorems 4.2, 4.3 and 4.4 we show that an additive penalty can be obtained for stochastic MABs as well, where unlike previous work in the bandit setting, the resulting term will be shown to be independent of the time horizon n when the delays are bounded.

Chapter 3

Adversarial multi-armed bandits

In this chapter we study a black-box approach to solving the adversarial multi-armed bandit problem with delays. We generalize the algorithm of Weinberger and Ordentlich (2002), introduced in Section 2.2.2, to the delayed MAB setting. In the case when the delays depend only on time and not on the actions, we prove that the regret increases by a multiplicative factor that depends on the delays. Note that this condition is naturally satisfied in important practical applications, such as when the delays arise because the information need to be propagated in a network. In the next Chapter we will treat the stochastic MAB problem, a special case of this problem, and show improved theoretical guarantees.

3.1 Interface of a non-delayed MAB algorithm

For our methods to work, we need a traditional multi-armed bandit algorithm that works in non-delayed environments. Such an algorithm has two main functions, GETACTION and FEEDREWARD. The GETACTION function, when called, must return the next action that the algorithm recommends. For *proper* behaviour, it is assumed that the action returned will actually be executed and the resulting reward will be provided to the bandit algorithm, in a call to the FEEDREWARD function. By *proper*, we mean that any deviation from this usage pattern may result in a loss of the regret guarantees that hold when the correct interaction pattern is used.

Another function from the non-delayed MAB algorithm that is useful is the constructor which returns a new instance of the algorithm. We denote a non-delayed MAB algorithm by BASE, and its constructor function by BASE(K, n), and assume that different instances returned by this function use independent internal random sources. Note that along with the number of actions K, we are giving the time horizon n to the algorithm to fulfil the need of those algorithms that need it. This parameter can be ignored if the algorithm does not depend on the time horizon. We note in passing that there exist standard techniques (such as *the doubling trick* (Cesa-Bianchi and Lugosi, 2006, page 38)) that can transform a horizon-dependent algorithm to one that works without knowing the horizon, while essentially maintaining the same theoretical guarantees. In some cases, however, a stochastic non-delayed MAB algorithm can take advantage of knowing the time horizon to achieve performance guarantees that can not be achieved if the time horizon is not known, as shown by Audibert et al. (2009) and Salomon and Audibert (2011).

3.2 Black-box use of non-delayed algorithms

The algorithm of Weinberger and Ordentlich (2002), introduced in Chapter 2, attains an expected regret which suffers a multiplicative penalty due to delays, and is known to be optimal in the full-information setting with constant delays. Although their algorithm works out-of-the-box for the bandit setting as well, it needs to know an upper bound on the delays. Here we extend their algorithm and analysis to the more general delayed adversarial MAB setting as described in Section 1.4, enabling us to handle arbitrary, random, and potentially unbounded delays, without any prior knowledge about the distribution or bounds of the delay.

Recall that the original algorithm of Weinberger and Ordentlich (2002) works in the following way: given an a priori known upper bound τ on the delays, the algorithm creates $(\tau + 1)$ instances of a non-delayed multi-armed bandit algorithm. We call each of these instances a *machine*. The original algorithm uses, at time steps $t = m(\tau + 1) + i,\ m = 0, 1, \ldots,$ the action chosen by machine i. When machine i is consulted the next time, i.e, at time step $(m + 1)(\tau + 1) + i$, the reward of its previous choice has already been observed (since τ time steps have passed), and the machine is therefore working in a non-delayed environment.

To extend this algorithm to work with general delays, we note that what we really need is to have, at each time step, at least one machine that is ready for a call to its GETACTION() function, i.e., that the reward of its most recent choice has arrived. Since a newly created machine can be consulted for its next action immediately, whenever all the machines are waiting for the reward of their most recent action, the algorithm can add a new machine and report its first choice as the choice for that time step. This way, the number of machines grows gradually, depending on the maximum delay upto that point in time. Specifically if the delays are equal to τ, we end up running the same procedure as the original one, with the same number of machines, albeit without the need to know τ upfront. In case of fixed delays, the difference to the original algorithm is that we add new machines as they are needed; hence we will create the $\tau + 1$ machines in the first $\tau + 1$ rounds, as opposed to the first round using the knowledge of τ.

The resulting algorithm, called Adversarial Black-box Bandits with Delays (ABBD), is given in Algorithm 1. ABBD maintains the sets $readyMachines$ and $waitingMachines$ of instances of the non-delayed adversarial multi-armed bandit algorithm BASE. At each time step, the algorithm checks to find a machine that is available, i.e., one that has received the reward for its last action, in the set $readyMachines$. If such a machine exists, it is used to determine the action for the current time step; otherwise, a new machine is created.

Algorithm 1 Adversarial Black-box Bandits with Delays (ABBD)

```
Input: $K$ (number of actions), $n$ (time horizon), BASE (non-delayed MAB algorithm).
Initialization:
$newMachine \leftarrow$ BASE$(K, n)$ {Create a new instance of the base algorithm}
$readyMachines \leftarrow \{newMachine\}$
$waitingMachines \leftarrow \emptyset$
for $t \leftarrow 1$ to $n$ do
    if $readyMachines == \emptyset$ then
        {Create a new machine to consult}
        $newMachine \leftarrow$ BASE$(K, n)$
        $readyMachines \leftarrow readyMachines \cup \{newMachine\}$
    end if
    {Consult a machine that is ready for inquiry}
    Remove machine $m$ from $readyMachines$. Choose the earliest created if there are multiple
    machines available.
    $I \leftarrow m.$GETACTION()
    $waitingMachines \leftarrow waitingMachines \cup \{(t, m)\}$
    Send the action $I$ to the environment
    for $i \leftarrow 1$ to $K$ do
        Get the rewards $\mathcal{X}_{i,t}$ from the environment
        for each $(s, X) \in \mathcal{X}_{i,t}$ do
            Remove $(s, machine)$ from $waitingMachines$
            $machine.$FEEDREWARD$(X)$
            $readyMachines \leftarrow readyMachines \cup \{machine\}$
        end for
    end for
end for
```

3.2.1 Regret analysis: the case of time-dependent delays

Let M_n be the number of machines created by the ABBD Algorithm by the end of time step n. Recall that $\mathcal{X}_{i,t} = \{(s, x_{i,s}) | 1 \leq s \leq t, I_s = i, s + \tau_{i,s} = t\}$ denotes the sequence of rewards belonging to action i and observed at the end of time step t. In particular, $\tau_{i,s}$ is the number of time steps that the reward of action i is delayed if the action is chosen at time step s. We will use the following lemma in the rest of our analysis.

Lemma 3.1. *At any time step* $t \geq 1$*, we have:*

$$M_t \leq \min\{t,\ \tau_t^* + 1\}$$

where

$$\tau_t^* = \max_{\substack{1 \leq i \leq K \\ 1 \leq s \leq t}} \tau_{i,s}.$$

Proof. We have $M_t \leq t$ because we do not create more than one machine in any time step. First, suppose t is a time step at which we have created a new machine. At that time step we needed to consult a machine, but all the machines were awaiting their rewards. This means that if t_j is the last time machine j had been chosen, for any $1 \leq j \leq M_{t-1}$ we have $t_j + \tau_{I_{t_j}, t_j} \geq t$. Therefore, $\tau_t^* \geq t - \min_{1 \leq j \leq M_{t-1}} t_j \geq t - (t - M_{t-1}) = M_{t-1}$ where the last inequality follows because at least one of the machines must have been last used at or before time step $t - M_{t-1}$, otherwise we have consulted two machines at the same time. Thus, by adding a new machine, we will have $M_t = M_{t-1} + 1 \leq \tau_t^* + 1$.

Now, consider the other case when we have not added a new machine at time t. Let $t' < t$ be the most recent time step when a new machine was added. We have $t > 1$ and the time step t' always exists because we are adding a new machine at time step 1. By the argument above, we have $M_t = M_{t'} \leq \tau_{t'}^* + 1 \leq \tau_t^* + 1$, which finishes the proof. □

We can now translate the expected regret guarantee of the non-delayed base algorithm to a guarantee in the delayed setting, at least in the special case when the delays only depend on the time step, but not on the action that is chosen at that time step. We will assume that the expected regret of the non-delayed base algorithm is bounded by a convex function f. This assumption holds for adversarial MAB problems, as well as other online learning contexts, which all have a regret upper bound of the form $\widetilde{O}(n^\alpha)$ for some $0 \leq \alpha < 1$, typically $\alpha = 1/2$.

Theorem 3.1. *Suppose that the non-delayed algorithm* BASE*, used in ABBD, has the guarantee that its expected regret after* n *time steps is bounded by* $f(n)$ *for some function* $f : [0, \infty) \mapsto [0, \infty)$ *that is concave and differentiable and satisfies* $f(0) = 0$*. Further assume that the delays of all actions at the same time step are the same, that is,* $\tau_{i,t} = \tau_t$ *for all* $1 \leq i \leq K, t \in \mathbb{N}$*. Then the expected regret of ABBD after* n *time steps satisfies*

$$\mathbb{E}\left[R_n\right] \;\leq\; \mathbb{E}\left[U_n f\left(\frac{n}{U_n}\right)\right] \tag{3.1}$$

where $U_n = \min\{n, \tau_n^* + 1\}$.

Proof. We prove the statement for deterministic delays. The extension to random delays follows by first conditioning on the delay sequence.

Let M_n be the number of machines created by time step n and $L_j,\ (1 \leq j \leq M_n)$ be the list of time steps in which ABBD has used the action chosen by machine j. Since at each time exactly one machine is used, the sets L_j together partition the set of time steps $\{1, \ldots, n\}$. Since the delays at each time step are the same for different actions, the available and waiting machines do not depend on the actions chosen. Therefore, given the list of delays $\tau_1, \tau_2, \ldots, \tau_n$, the number of machines M_n and the sets L_j are determined no matter what the rewards or the choices of the machines are, and so is the machine that will be used in each time step. This means that given the delays, the sequence of rewards $x_{i,t}, t \in L_j, 1 \leq j \leq K$, that each machine j will operate on is determined. Therefore, each machine j is in fact running in an oblivious adversarial MAB environment given the delays. Let n_j be the size of the list L_j, which is determined by the delays as L_j is. Define $R_{n_j}^j$ to be the regret of machine j in this environment, that is

$$R_{n_j}^j = \max_{1 \leq i \leq K} \sum_{t \in L_j} x_{i,t} - \sum_{t \in L_j} x_{I_t,t}$$

where I_t is the action chosen by ABBD (and machine j) at time step t. Then:

$$\begin{aligned}
R_n &= \max_{1\le i\le K} \sum_{t=1}^{n} x_{i,t} - \sum_{t=1}^{n} x_{I_t,t} \\
&= \max_{1\le i\le K} \sum_{j=1}^{M_n} \sum_{t\in L_j} x_{i,t} - \sum_{j=1}^{M_n} \sum_{t\in L_j} x_{I_t,t} \\
&\le \sum_{j=1}^{M_n} \left(\max_{1\le i\le K} \sum_{t\in L_j} x_{i,t} - \sum_{t\in L_j} x_{I_t,t} \right) \\
&= \sum_{j=1}^{M_n} R^j_{n_j}. \qquad (3.2)
\end{aligned}$$

By the assumption of the theorem, in a non-delayed environment, for any $n \in \mathbb{N}$ the expected regret of the base algorithm over a sequence of n rewards is bounded by $f(n)$. Therefore, using the fact that given the delays, each machine j is running in a non-delayed oblivious MAB environment, we get

$$\mathbb{E}\left[R^j_{n_j}\right] \le f(n_j),$$

for each $1 \le j \le M_n$. Taking the expectation of (3.2) we get:

$$\begin{aligned}
\mathbb{E}\left[R_n\right] &\le \mathbb{E}\left[\sum_{j=1}^{M_n} R^j_{n_j}\right] \\
&= \sum_{j=1}^{M_n} \mathbb{E}\left[R^j_{n_j}\right] \\
&\le \sum_{j=1}^{M_n} f(n_j) = M_n \sum_{j=1}^{M_n} \frac{1}{M_n} f(n_j) \\
&\le M_n f\left(\sum_{j=1}^{M_n} \frac{1}{M_n} n_j\right) = M_n f\left(\frac{n}{M_n}\right) \qquad (3.3)
\end{aligned}$$

where the last line follows from Jensen's inequality (Lemma A.2). By Lemma 3.1, $n/M_n \ge n/U_n$ where $U_n = \min\{n, \tau^*_n + 1\}$. Since f is differentiable and concave, and because $f(0)$ is zero, $sf(1/s)$ is non-decreasing in s according to Lemma A.3. By setting $s = M_n/n$, from (3.3) we get:

$$\mathbb{E}\left[R_n\right] \le n\frac{M_n}{n} f\left(\frac{n}{M_n}\right) \le n\frac{U_n}{n} f\left(\frac{n}{U_n}\right) = U_n f\left(\frac{n}{U_n}\right).$$

This concludes the proof. □

Note that our analysis is not limited to the bandit case, and applies to the full-information case as well. Therefore, using a full-information base algorithm, we recover the performance guarantee of Weinberger and Ordentlich (2002) in the case of bounded delays, without the need to know the upper bound τ, while we still get a practical algorithm and guarantee even if the delays are not bounded.

We illustrate the use of Theorem 3.1 using an example. Consider the INF algorithm of Audibert and Bubeck (2009). Recall from Chapter 2 that INF enjoys an expected regret bounded as

$$\mathbb{E}\left[R_n^{\text{INF}}\right] \leq 12.2\sqrt{nK}.$$

If ABBD is run with INF as the base algorithm, then according to Theorem 3.1, we get:

$$\mathbb{E}\left[R_n\right] \leq \mathbb{E}\left[U_n\left(12.2\sqrt{nK/U_n}\right)\right] = 12.2\sqrt{nK}\,\mathbb{E}\left[\sqrt{U_n}\right] \leq 12.2\sqrt{(\mathbb{E}\left[\tau_n^*\right]+1)nk} \qquad (3.4)$$

In the case of delays bounded in $[0, \tau]$, we have $\mathbb{E}\left[\tau_n^*\right] \leq \tau$, and therefore (3.4) gives $\mathbb{E}\left[R_n\right] \leq 12.2\sqrt{(\tau+1)nK}$, which means the penalty we suffer due to delays is bounded by a multiplicative factor of $\sqrt{\tau+1}$. It remains an open problem to extend the analysis to the case when the delays can be action dependent.

3.3 A note on optimality

In this chapter, we showed an expected regret of type $(\tau+1)\mathbb{E}\left[R^{\text{Base}}(n/(\tau+1))\right]$, with $R^{\text{Base}}(n)$ being the regret of a non-delayed MAB agent on a reward sequence of length n. The question is whether such a multiplicative penalty is the best we can get in the delayed setting.

In Chapter 2, we mentioned that Weinberger and Ordentlich (2002) gave a lower bound of this type for the full-information setting. The basic idea of their argument is that if we have sequences of rewards in which each reward is repeated $\tau+1$ times, then the agent does not receive any feedback on a reward before having to act on all the $\tau+1$ of those rewards, and once the information is received, the rewards have probably changed. Therefore, the agent is in fact working on a non-delayed sequence of length $n/(\tau+1)$ (when the repetitions are removed), in which each of its losses are multiplied by $\tau+1$.

While it may seem that this lower bound extends to the bandit setting, in fact it does not. The reason is that in the full-information setting, in the $\tau+1$ rounds when a reward is repeated, the agent cannot act in such a way that at the end of the $\tau+1$ round, it gets more information than it is possible to get in one round in the non-delayed setting. In the bandit setting, however, this is not the case. In other words, the MAB agent can use this period for $\tau+1$ explorations, and therefore get more information after the $\tau+1$ rounds than what was possible to get in one round in the non-delayed MAB setting. Therefore, we cannot simply multiply the regret in the non-delayed setting by $\tau+1$. At the moment, we are not aware of a method to extend this lower bound to the bandit setting.

3.4 Time and memory usage of ABBD

In this section we discuss the time and space complexity of the algorithm introduced in the previous section. Assume that creating a new instance of the base algorithm needs constant time, and insertion to the set $waitingMachines$ takes logarithmic time [1]. Before sending the action to the environment

[1]For the $readyMachines$ we can use a linked list, since we do not need look up but only $O(1)$ insertion and deletion. The logarithmic insertion and search time is satisfied for data structures such as Red-Black trees (Cormen et al., 2009).

in time step t, the time that the ABBD algorithm spends is in the same order as the time of the call GETACTION to the base machine plus $O(\log M_t)$.

In the update phase, since the number of calls to the FEEDREWARD function is at most as much as the number of rewards and each of those calls is preceded by a search in the set of waiting machines, over n time steps the algorithm will need a total time of at most $O(n(\log(M_n) + t_{\text{Base}}))$, where t_{Base} is the running time of a call to FEEDREWARD. This means that on average, the time it takes for an update in each time step is as much as a call to the FEEDREWARD function plus $O(\log M_n)$.

The space complexity of the algorithm is determined by the number of non-delayed instances that it is storing. Since by Lemma 3.1 the number of machines created by ABBD at time step t is bounded by τ_t^*, where τ_t^* is the largest delay by that time step, the space complexity of the algorithm is at most of the order τ_t^* times that of a non-delayed MAB algorithm. If the base algorithm has a memory requirement of $O(K)$, then the memory requirement of ABBD by time step t is upper bounded by $O(K\tau_t^*)$.

3.4.1 Why black-box?

According to the discussion above, the memory requirement of the black box approach is proportional to the number of instances created of the non-delayed algorithms. One may then wonder why a black box approach makes sense at all. The benefit of using a black-box approach is both theoretical and practical. Theoretically, using a black box approach we automatically enjoy the results of any further development of the non-delayed algorithms, directly transferring them to the delayed case. Furthermore, practically the black-box approach can be used in environments with established and tested implementations of non-delayed algorithms, so as to reuse them reliably without the extra cost of developing and testing a brand-new algorithm for the delayed setting.

In cases when memory is a constrained resource, we can use the approach of Neu et al. (2010), which we referred to in Section 2.2.5. That is, we can run the EXP3 algorithm introduced in Section 2.1 with only the observed rewards and, under constant delays, still get a theoretical guarantee similar to what we showed in Theorem 3.1. Furthermore, in Chapter 4, we will see examples of these algorithms for the stochastic MAB problem with a memory requirement of $O(K)$, as is the case for the usual non-delayed stochastic MAB algorithms.

Chapter 4

Stochastic multi-armed bandits

In this chapter we study the delayed-feedback multi-armed bandit problem with the additional stochastic assumptions defined in Section 1.3.1. We study the theoretical properties of this formulation which enable us to derive better regret guarantees than the general case, and develop two types of algorithms that give us such guarantees.

Before starting with the algorithms, we study the properties of the stochastic MAB problem under delayed feedback. These properties are used in deriving and analysing our algorithms in the next sections. In Section 4.2, we give a black-box style algorithm which reduces the stochastic delayed MAB problem to the stochastic non-delayed MAB problem with the same reward distributions, and show that the penalty in the regret for this transformation is bounded by an additive term depending on the delays. In Section 4.3, we show another approach for solving the delayed MAB problem which uses adapted versions of UCB-style non-delayed MAB algorithms introduced in Section 2.1. We give a method to generally extend the theoretical guarantees of this class of algorithms to the delayed setting, again resulting in an additive penalty.

4.1 Properties of the stochastic delayed MAB problem

In this section we study the implications of delays on the assumptions of the stochastic MAB problem. As noted before, the delays can change the order in which the rewards arrive. First, we want to show that if the delays are independent of the rewards, then the sequence of *observed* rewards for each action is still an i.i.d. sequence with the same distribution as the original reward sequence.

More precisely, for each action $1 \leq i \leq K$ define the sequence $\{Y_{i,s}\}_{s\in\mathbb{N}}$ to be the sequence of rewards that is observed by the agent. That means $Y_{i,1}$ is a random variable taking the value of the first reward that the algorithm observes for action i, $Y_{i,2}$ is the second one, and so on. We had denoted the sequence of rewards of action i at each time step t by $\{X_{i,t}\}_{t\in\mathbb{N}}$. Note that in the sequence $\{Y_{i,t}\}$ we are considering only the rewards of the choices that the agent has actually made, which is only a reordered subsequence of the rewards $X_{i,t}$. The way this sequence is reordered can depend on the choices of the agent and therefore it is not clear whether this imposes a dependence between the observed rewards. In the following, we show that this dependence is actually not imposed.

We start by showing that the i.i.d. property is preserved when the sequence $X_{i,t}$ is reordered. We have the following lemma.

Lemma 4.1. *Let $\{X_t\}_{t\in\mathbb{N}}$, be a sequence of independent, identically distributed random variables. If we reorder this sequence according to an independent random permutation, then the resulting sequence is i.i.d. with the same distribution as $\{X_t\}_{t\in\mathbb{N}}$.*

Proof. Let the reordered sequence be denoted by $\{Z_t\}_{t\in\mathbb{N}}$. We need to show that for all $n \in \mathbb{N}$, for all $y_1, y_2, \ldots, y_n$, we have

$$\mathbb{P}\left\{Z_1 \leq y_1, Z_2 \leq y_2, \ldots, Z_n \leq y_n\right\} = \mathbb{P}\left\{X_1 \leq y_1, X_2 \leq y_2, \ldots, X_n \leq y_n\right\}.$$

Since $\{X_t\}_{t\in\mathbb{N}}$ is i.i.d., for any fixed permutation the equation above holds as both sides are equal to $\Pi_{t=1}^{n}\mathbb{P}\left\{X_t \leq y_t\right\}$. Since the permutations are independent of the sequence $\{X_t\}_{t\in\mathbb{N}}$, using the law of total probability this extends to the general case as well. □

We also need the following lemma (Doob, 1953, Page 145, Chapter III, Theorem 5.2).

Lemma 4.2. *Let $\{X_{i,t}\}_{t\in\mathbb{N}}$ be a sequence of i.i.d. random variables, and $\{X'_{i,t}\}_{t\in\mathbb{N}}$ be its subsequence such that the decision whether to include $X_{i,t}$ in the subsequence is independent of future values in the sequence, i.e., $X_{i,s}$ for $s \geq t$. Then the sequence $\{X'_{i,t}\}_{t\in\mathbb{N}}$ is an i.i.d. sequence with the same distribution as $\{X_{i,t}\}_{t\in\mathbb{N}}$.*

We can now use the two previous lemmas to prove that the i.i.d. property is preserved on the sequence of *observed* rewards. This is captured in the following theorem.

Theorem 4.1. *Consider a delayed stochastic multi-armed bandit game, with any multi-armed bandit agent, and with delays independent of the rewards. For any action i, for any $s \in \mathbb{N}$ let $Y_{i,s}$ denote the s^{th} reward the algorithm observes for choosing action i. Then the sequence $\{Y_{i,s}\}_{s\in\mathbb{N}}$ is an i.i.d. sequence with the same distribution as the sequence of rewards $\{X_{i,t}\}_{t\in\mathbb{N}}$.*

Proof. Define $\{Z_{i,t}\}_{t\in\mathbb{N}}$ to be the sequence resulting from sorting the variables $X_{i,t}$ by their *possible* observation times $t+\tau_{i,t}$ (that is, $Z_{i,1}$ is the earliest reward that can be obtained if action i is chosen at the appropriate time, and so on). Since delays are independent of the rewards, they define an independent reordering on the sequence of rewards. Hence, by Lemma 4.1, $\{Z_{i,t}\}_{t\in\mathbb{N}}$ is an i.i.d. sequence with the same distribution as $\{X_{i,t}\}_{t\in\mathbb{N}}$. Note that $\{Y_{i,s}\}_{s\in\mathbb{N}}$, the sequence of rewards of the choices of action i by the agent sorted by their observation times, is a subsequence of $\{Z_{i,t}\}_{t\in\mathbb{N}}$ where the decision whether to include each $Z_{i,t}$ in the subsequence cannot depend on future rewards $Z_{i,s}, s \geq t$. Also, the rewards of other actions that are used in this decision are independent of $Z_{i,t}$. Hence, by Lemma 4.2, $\{Y_{i,t}\}_{t\in\mathbb{N}}$ is an i.i.d. sequence with the same distribution as $Z_{i,t}$, which in turn has the same distribution as $X_{i,t}$. □

Before continuing to the algorithms, we give some additional notation and properties of the stochastic setting. Define $T_i(t) = \sum_{s=1}^{t} \mathbb{I}\{I_s = i\}$ to be the number of times action i has been chosen by the agent up to (and including) time step t. Define $C_i(s)$ to be the "inverse" of $T_i(t)$, showing the time step at which the agent has chosen action i for the s^{th} time. Also, define $S_i(t)$ to be the number of rewards that are observed by the end of time step t. Let $i^* = \underset{1 \le i \le K}{\operatorname{argmax}} \mu_i$ denote the index of an optimal action (break ties by choosing the smallest index). For each action i, let $\Delta_i = \mu_{i^*} - \mu_i$ denote the gap between the expected reward of an optimal action and that of action i. Let $\tau^*_{i,n} = \max_{1 \le t \le n} \tau_{i,t}$ denote the maximum delay for action i over n time steps. We will use this quantity in our regret upper bounds.

In the stochastic setting, from (1.2) we can get a more tractable expression about the expected regret. In particular we have

$$\mathbb{E}[R_n] = n\mu_{i^*} - \sum_{i=1}^{K} \mu_i \mathbb{E}[T_i(n)] = \sum_{i=1}^{K} \Delta_i \mathbb{E}[T_i(n)]. \tag{4.1}$$

This essentially means that in order to bound the expected regret, it suffices to bound the expected number of times each action is chosen. We will use this fact together with the notation above in the next sections.

4.2 Black-box use of non-delayed stochastic MAB algorithms

In this section we give a black-box style algorithm for the stochastic MAB problem under delayed feedback. Algorithm 2 shows the way our method works. The algorithm, called Stochastic Black-box Bandit with Delays (SBBD), builds on a non-delayed algorithm BASE which is consulted when making actions in the environment. We use the same model of a non-delayed MAB algorithm as described in Section 3.1. To manage the delayed observation of rewards and simulate a non-delayed environment for this base algorithm, SBBD maintains a First-In-First-Out (FIFO) buffer $Q[i]$ for each action $1 \le i \le K$. This buffer is empty at the beginning and is used to store the rewards observed for action i so that they can be used later when the base algorithm wants them.

Figure 4.1 shows the block diagram for the algorithm. At each time step, an SBBD agent tries to feed into the base algorithm all the information available, and then choose its action based on the next choice of the base algorithm for which there are no buffered rewards remaining. More precisely, to make a decision at each time step t, SBBD first runs the base algorithm in a loop and for each action the base algorithm chooses, gives it the next reward from the buffer of rewards of that action, until the base algorithm chooses an action i for which the buffer is empty. SBBD then reports this action as its choice at time step t. After reporting its choice, at the end of the time step, SBBD observes new rewards for each of the actions and puts them into the queue for the respective action, so that they can be used later when the base algorithm asks for them. If there is no reward for action I_t, in the next time step SBBD will not be able to feed the base algorithm with a reward, so

Algorithm 2 Stochastic Black-box Bandit with Delays (SBBD)

```
Input: $K$ (number of actions), $n$ (time horizon, optional), BASE (non-delayed stochastic MAB
algorithm)
Initialization:
BASEAGENT $\leftarrow$ BASE$(K, n)$ {Create a new instance of the base algorithm}
For each action $1 \leq i \leq K$, $Q[i] \leftarrow$ new FIFO_QUEUE()

$I \leftarrow$ BASEAGENT.GETACTION()
for $t \leftarrow 1$ to $n$ do
    while $Q[I] \neq \emptyset$ do
        {Feed the base algorithm with a reward from the buffer of the selected action}
        $r \leftarrow Q[I]$.POP()
        BASEAGENT.FEEDREWARD$(r)$
        {Get the next action from the base algorithm}
        $I \leftarrow$ BASEAGENT.GETACTION()
    end while
    {There has been no buffered reward for action $I$, so send it to the environment}
    Send action $I$ to the environment
    Get the sets of rewards $\mathcal{X}_{i,t}$ for each action $i$ from the environment
    For each action $i$, for each $(s, r) \in \mathcal{X}_{i,t}$, $Q[i]$.PUSH$(r)$
end for
```

it will again pick the same action as in the previous time step. As a result, once the base algorithm picks an action for which the reward queue is empty, it keeps picking the same action until a reward for that action is observed.

Consider the case when all the rewards are observed after a constant delay τ. The first time the base algorithm chooses an action, SBBD keeps choosing that action for τ times consecutively, until it receives the first reward for that action. On the next $\tau - 1$ rounds, the rest of the rewards of that τ choices arrive, and are stored in the buffer for that action. When the base algorithm chooses that action at a later time, it is immediately given the reward from the buffer, without executing the action in the environment as long as there is some reward buffered for that action. Once the base algorithm chooses that action for τ more times, the buffer gets empty and SBBD issues another τ rounds of consecutive plays of the action, filling the buffer again. This way the algorithm goes through periods of τ time steps in which it chooses the same action, buffering the rewards and feeding them to the base algorithm later. This is in fact similar to concentrating the exploration that the base algorithm does when there are no delays, so that when the base algorithm needs the rewards, they are available with no delays. At the same time, by choosing only the action for which the buffer is empty, the SBBD makes sure that the base algorithm is fed with as much available information as it wants before exploring for more information.

4.2.1 Regret analysis

To investigate the performance of the algorithm, first we show that the penalty due to delays on the expected regret of the algorithm is bounded by an additive term that depends on the delays.

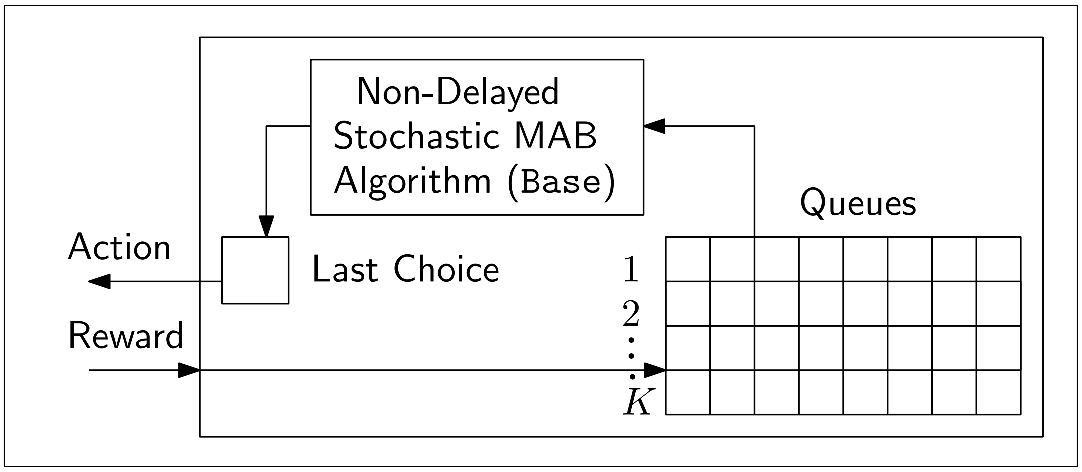


Figure 4.1: The SBBD algorithm. The algorithm uses a non-delayed stochastic MAB algorithm (BASE) as a black-box, and simulates a non-delayed environment for it. When the rewards of an action are delayed, the SBBD algorithm keeps choosing the last action chosen by the black-box algorithm. The rewards that arrive later are buffered in the queues, and given to the black-box algorithm BASE when it chooses that action again.

After that, we show how we can employ the properties of the non-delayed base algorithm to give high-probability upper bounds on the regret for the delayed-feedback problem.

We will use the following notation throughout the analysis. Assume that SBBD queries the base algorithm for s times (i.e., the GETACTION() function of the base algorithm is called s times). Let $T'_i(s)$ denote the number of times action i has been chosen by the base algorithm during these s queries. Recall that $T_i(n)$ is the number of time steps that the agent (SBBD) has chosen action i in the first n time steps. Also, recall that $\tau^*_{i,n} = \max_{t=1,\dots,n} \tau_{i,t}$ denotes the maximum delay for action i.

The following lemma gives us the relationship between the delays and the number of rewards not fed into the base algorithm, and forms the basis of the expected regret bound.

Lemma 4.3. *Assume SBBD is run for $n \geq 1$ time steps, and has queried the base algorithm for n' times. Then*

$$n' \leq n$$

and

$$0 \leq T_i(n) - T'_i(n') \leq \max_{1 \leq s \leq T'_i(n)} \tau_{i,C_i(s)} \leq \tau^*_{i,n}. \tag{4.2}$$

Proof. Since the total number of calls made to GETACTION() function of the base algorithm before the end of time step n is at most one more than the number of rewards observed by then, and the number of rewards observed is at most $n-1$, we have $n' \leq n$. Whenever the base algorithm chooses action i, either SBBD chooses that action as well (if the buffer $Q[i]$ is empty), or the reward buffer is not empty which means SBBD had already made an extra choice of i. Therefore we have $T_i(n) - T'_i(n') \geq 0$.

Recall that $C_i(s)$ was the time at which the agent (SBBD) had chosen action i for the s^{th} time. Let $t'_n = C_i(T'_i(n'))$ be the time step at which SBBD has picked action i for the $T'_i(n')$-th time. Let $t_n = C_i(T_i(n))$ be the time step at which SBBD has made its last choice of action i by time step n. Then $t'_n \leq t_n$ and the number of times SBBD has chosen action i between time steps t'_n and t_n, i.e., $T_i(n) - T'_i(n')$, can be written as

$$T_i(n) - T'_i(n') = \sum_{t=t'_n+1}^{t_n} \mathbb{I}\{I_t = i\} \leq t_n - t'_n.$$

If $t_n = t'_n$, the lemma holds trivially because the delays are non-negative, so assume that $t_n > t'_n$. This means that after the first $T'_i(n')$ choices of action i, SBBD has chosen action i at least once more at time step t_n. But our algorithm only picks action i when the base algorithm requests it and there are no rewards buffered in $Q[i]$. Therefore, if all of the rewards of the first $T'_i(n')$ choices of action i were observed before time step t_n, SBBD would not pick this action at time step t_n because the base algorithm would not need another reward for action i. Therefore, for some $1 \leq s^* \leq T'_i(n')$, the reward of the s^*-th choice of action i has been observed at the time step $C_i(s^*) + \tau_{i,C_i(s^*)} \geq t_n$. Since $C_i(s*) \leq C_i(T'_i(n')) = t'_n$, we have:

$$t_n - t'_n \leq (C_i(s^*) + \tau_{i,C_i(s^*)}) - C_i(s^*) = \tau_{i,C_i(s^*)} \leq \max_{1 \leq s \leq T'_i(n)} \tau_{i,C_i(s)} \leq \tau^*_{i,n}.$$

which finishes the proof. □

We can now analyse the expected regret of Algorithm 2 using Lemma 4.3 and Equation (4.1). We will have the following theorem.

Theorem 4.2. *When run with a base algorithm* BASE *in a delayed stochastic MAB environment, the expected regret of SBBD is upper-bounded by*

$$\mathbb{E}[R_n] \leq \mathbb{E}\left[R_n^{\text{BASE}}\right] + \sum_{i=1}^{K} \Delta_i \mathbb{E}\left[\tau^*_{i,n}\right] \tag{4.3}$$

where $\mathbb{E}\left[R_n^{\text{BASE}}\right]$ is the expected regret of the base algorithm when run in a non-delayed environment with the same reward distributions as the delayed problem.

Proof. Taking expectation of (4.2) we get:

$$\mathbb{E}[T_i(n)] \leq \mathbb{E}[T'_i(n')] + \mathbb{E}\left[\tau^*_{i,n}\right].$$

Assume that the game is run so long that the base algorithm is queried for n times (i.e., it is queried $n-n'$ more times). Then, since $n' \leq n$, the number of times action i is chosen by the base algorithm, namely $T'_i(n)$, can only increase, that is, $T'_i(n') \leq T'_i(n)$. Therefore,

$$\mathbb{E}[T_i(n)] \leq \mathbb{E}[T'_i(n)] + \mathbb{E}\left[\tau^*_{i,n}\right]$$

Multiplying both sides by Δ_i and using (4.1), we get:

$$\sum_{i=1}^{K} \Delta_i \mathbb{E}\left[T_i(n)\right] \leq \sum_{i=1}^{K} \Delta_i \mathbb{E}\left[T_i'(n)\right] + \sum_{i=1}^{K} \Delta_i \mathbb{E}\left[\tau_{i,n}^*\right]. \tag{4.4}$$

As shown in Theorem 4.1, the observed rewards $Y_{i,1}, Y_{i,2}, \ldots, Y_{i,T_i'(n')}, \ldots Y_{i,T_i(n)}$ are i.i.d. with the same distribution as the rewards $\{X_{i,t}\}_{t\in\mathbb{N}}$. The base algorithm has worked on the first $T_i'(n')$ of these rewards. Therefore, the base algorithm has operated in a simulated non-delayed environment with rewards that are i.i.d. with the same distribution of the rewards in the delayed environment. $\mathbb{E}\left[T_i'(n)\right]$ is therefore the expected number of times that the base algorithm would choose action i in an environment without delays but with the same reward distributions as the original delayed problem. Thus, the first summation in the right hand side of (4.4) is in fact $\mathbb{E}\left[R_n^{\text{Base}}\right]$, the expected regret of the base algorithm. This concludes the proof. □

4.2.2 Time and memory usage of SBBD

In this section we discuss the time and space complexity of SBBD. In the action selection phase, the total number (over n time steps) of accesses to the buffers cannot exceed the number of observed rewards, which is at most as large as the number of time steps n. Therefore, the average number of calls to the queues per time step is $O(1)$. Also, the total number of calls in n time steps to the GETACTION function of the base algorithm is at most as large as the number of rewards (since each of them is followed by a FEEDREWARD call). The average number of such calls per time step is thus $O(1)$. Therefore, the average time consumed per time step for getting an action from SBBD is of the same order as a regular non-delayed stochastic MAB algorithm.

In the update phase, in the total n rounds SBBD processes at most n rewards, each of them resulting in a call to FEEDREWARD(). Therefore, on average there is one call to FEEDREWARD() per time step, and therefore the time overhead during an update is similar to a non-delayed stochastic MAB algorithm.

The buffers in SBBD are the main elements consuming memory. As shown in Lemma 4.3, the size of each of these buffers at each time step is at most as large as the maximum delay by that time step. Therefore, at each time step t, the memory requirement of the algorithm is of order $O(\sum_{i=1}^{K} \tau_{i,t}^*)$ which is in turn of order $O(K\tau_t^*)$, in addition to the memory usage of the base algorithm.

Here again, the space complexity is the bottleneck of our black-box style algorithm. In the next section we will explore another approach that alleviates this problem.

4.3 Modification of non-delayed algorithms

Here we introduce another method to address the problem of stochastic MABs with delayed feedback. This approach is based on modifying non-delayed algorithms to work with any reward that is observed, as opposed to requiring one feedback per time step. In Section 4.3.1 we will show that standard stochastic bandit algorithms based on upper confidence bounds (UCBs) can be used in the delayed setting while retaining their theoretical guarantees. The general method to transform theoretical guarantees to the delayed setting is shown in Section 4.3.1, and is applied to the standard UCB1 algorithm of Auer et al. (2002) in Section 4.3.2. In Section 4.3.3, we show another example using the recently proposed KL-UCB algorithm (Garivier and Cappé, 2011), for which applying this general scheme is more involved.

4.3.1 UCB-type algorithms in delayed environments

The UCB class of algorithms was introduced in Chapter 2. Recall that a UCB algorithm maintains, for each action $i \in \mathcal{K}$, an upper confidence bound $B_{i,s,t}$, and at each time step chooses an action with largest bound. In a non-delayed environment, the number of observed rewards for action i before time step t is $T_i(t-1)$, and therefore the action to choose at time step t is given by

$$I_t = \operatorname*{argmax}_{i \in \mathcal{K}} B_{i,T_i(t-1),t}.$$

In case of delays, however, not all the rewards of previous actions are observed. In general, the number $S_i(t)$ of rewards observed for action i by the end of time step t can be smaller than the number of choices $T_i(t)$. It may seem like a good idea to use the same upper confidence bounds in this setting as well, with all rewards that have been observed. This means choosing

$$I_t = \operatorname*{argmax}_{i \in \mathcal{K}} B_{i,S_i(t-1),t}$$

at each time step t. The question is whether this approach performs well in the delayed setting, and how the resulting regret behaves.

In what follows, we want to answer this question for UCB-type algorithms in general, and show that the same method of analysis that is used for them in the non-delayed setting can be exploited to give performance guarantees in the delayed case. In the following lemma (which is similar to Lemma 4.3 but holds for any algorithm) we bound the number of missing rewards. Then we go on to show how it can be exploited to re-use the machinery used to analyze the original non-delayed algorithms.

Lemma 4.4. *For any algorithm in the stochastic delayed MAB game, at any time step $t \in \mathbb{N}$, we have*

$$T_i(t) - S_i(t) \le \max_{1 \le s \le S_i(t)} \tau_{i,s} \le \tau^*_{i,t}. \tag{4.5}$$

Proof. If $S_i(t) = T_i(t)$, then the lemma holds trivially, so consider the case when $T_i(t) \ge S_i(t)+1$. Then from the first $S_i(t)+1 \le T_i(t)$ times the agent chooses action i, the reward of at least one of them, for example that of the s_1^{th} choice, is not observed by time step t, otherwise we would have $S_i(t) = S_i(t)+1$ which is not possible. That means there exists $1 \le s_1 \le S_i(t)+1$ such that

$C_i(s_1) + \tau_{i,C_i(s_1)} \geq t+1$. Let $t_1 = C_i(S_i(t)+1)$ be the time step of the $(S_i(t)+1)$-th choice of action i. Note that $C_i(s_1) \leq t_1 \leq t$ by definition. Therefore we can count the number of choices of action i between the time steps t_1 and t as

$$\begin{aligned} T_i(t) - S_i(t) &= \sum_{s=t_1}^{t} \mathbb{I}\{I_s = i\} \\ &\leq (t+1) - t_1 \\ &\leq C_i(s_1) + \tau_{i,C_i(s_1)} - C_i(s_1) \\ &= \tau_{i,C_i(s_1)} \leq \max_{1\leq s\leq S_i(n)} \tau_{i,C_i(s)} \leq \max_{1\leq s\leq t} \tau_{i,s} \end{aligned}$$

which was the claim of the lemma. □

We will now use this lemma to provide a theoretical analysis of UCB-type algorithms in the delayed setting. This will be done by showing how to adapt the proof machinery of a typical UCB algorithm to the delayed setting. We start by characterizing the way UCB-type algorithms are shown to perform well.

A usual analysis of the performance of a UCB algorithm works by giving an upper bound for the number of trials of any suboptimal action, and showing that it is unlikely that the algorithm makes more choices of that action. More precisely, the general analysis starts by the fact that for any $\ell > 1$,

$$T_i(n) \leq \ell + \sum_{t=1}^{n} \mathbb{I}\{I_t = i,\ T_i(t) > \ell\},$$

and then shows that the events in the summation have very low probability for sufficiently large ℓ. This is because the event $\mathbb{I}\{I_t = i\}$ implies that the upper confidence bound $B_{i,T_i(t-1),t}$ is greater than the upper confidence bound of an optimal arm, $B_{i^*,\ T_{i^*}(t-1),t}$, that is,

$$T_i(n) \leq \ell + \sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, B_{i,T_i(t-1),t} \geq B_{i^*,T_{i^*}(t-1),t},\ T_i(t-1) \geq \ell\right\} \tag{4.6}$$

The events in the summation are very improbable if ℓ is large enough, since the number of samples, $T_i(t-1)$, from the reward distribution of the suboptimal action i become large enough to give a good enough estimate of its mean and avoid choosing it over a better arm. To show this, concentration inequalities suitable for the specific form of the upper-confidence bound are applied. Examples of such concentration inequalities include Hoeffding's inequality (Lemma A.1) as well as Theorem 10 of Garivier and Cappé (2011) (Theorem 4.5 in this chapter), which are used for UCB1 and KL-UCB, respectively.

Trying to use the same method of analysis in the delayed setting, we can write

$$T_i(n) \leq \ell + \sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t},\ T_i(t-1) \geq \ell\right\}.$$

In this case, however, the number of samples used in the UCB is $S_i(t-1)$, while we know $T_i(t-1) \geq \ell$. However, Lemma 4.4 bounds how much smaller $S_i(t-1)$ can be than $T_i(t-1)$. Therefore, setting

$\ell = \ell' + \tau^*_{i,n}$ we get:

$$T_i(n) \leq \ell' + \tau^*_{i,n} + \sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t},\ T_i(t-1) \geq \ell' + \tau^*_{i,n}\right\},$$

and since by Lemma 4.4, $T_i(t-1) - S_i(t-1) \leq \tau^*_{i,t-1} \leq \tau^*_{i,n}$,

$$T_i(n) \leq \ell' + \tau^*_{i,n} + \sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t},\ S_i(t-1) \geq \ell'\right\}. \tag{4.7}$$

The probability of the event in the summation above can now be bounded using the same concentration inequalities as the ones used to bound (4.6) in the original analysis for the non-delayed setting, with the same value for ℓ' as was previously used for ℓ. Thus, we can re-use the proof machinery for the regret analysis of UCB-type algorithms in a delayed environment, using the same UCBs but only on the rewards that are observed, with only an additive penalty in the regret compared to the non-delayed case. We will illustrate this using two examples. The first example uses the well-known UCB1 algorithm, while the second one uses the recent KL-UCB algorithm.

4.3.2 UCB1 under delayed feedback

As an example of applying the general framework described above, we show the process for the UCB1 algorithm (Auer et al., 2002). In the delayed setting, at time step t we will use upper confidence bounds $B_{i,S_i(t-1),t}$, where $B_{i,s,t} = \bar{Y}_{i,s} + \sqrt{\frac{2\log(t)}{s}}$ and $\bar{Y}_{i,s} = \frac{1}{s}\sum_{t=1}^{s} Y_{i,t}$ is the average over the first s observed rewards. We call this algorithm Delayed-UCB1 to distinguish it from the original UCB1 for non-delayed environments. Note that in case there are no delays, Delayed-UCB1 reduces to UCB1.

Following the same lines as the proof of Auer et al. (2002), we get the following theorem, showing an additive penalty compared to Theorem 2.1.

Theorem 4.3. *For any $n \geq 1$, the expected regret of the Delayed-UCB1 algorithm is bounded as*

$$\mathbb{E}\left[R_n\right] \leq 8\left[\sum_{i:\mu_i<\mu_{i^*}} \frac{\log n}{\Delta_i}\right] + \left(\sum_{i=1}^{K} \Delta_i \mathbb{E}\left[\tau^*_{i,n}\right]\right) + 3.5\sum_{i=1}^{K}\Delta_i. \tag{4.8}$$

Proof. Following the outline of the previous section, we can bound the summation in (4.7) using the same analysis as in the original UCB1 paper (Auer et al., 2002). In particular, for any action i we can write

$$\begin{aligned}\sum_{t=1}^{n} \mathbb{I}\left\{B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t},\ S_i(t-1) \geq \ell'\right\} \leq \\ \sum_{t=1}^{n} \mathbb{I}\left\{B_{i^*,S_{i^*}(t-1),t} \leq \mu_{i^*},\ S_i(t-1) \geq \ell'\right\} + \\ \sum_{t=1}^{n} \mathbb{I}\left\{B_{i,S_i(t-1),t} \geq \mu_{i^*},\ S_i(t-1) \geq \ell'\right\}.\end{aligned} \tag{4.9}$$

The event in the first summation implies that either $\bar{Y}_{i^*,S_{i^*}(t-1)}+\sqrt{\frac{2\log(t)}{S_{i^*}(t-1)}} \le \mu_{i^*}$ or $\bar{Y}_{i,S_i(t-1)}-\sqrt{\frac{2\log(t)}{S_i(t-1)}} \ge \mu_i$. Hence,

$$\begin{aligned}(4.9) \le \; & \sum_{t=1}^{n} \mathbb{I}\left\{\bar{Y}_{i^*,S_{i^*}(t-1)} + \sqrt{\frac{2\log(t)}{S_{i^*}(t-1)}} \le \mu_{i^*},\ S_i(t-1) \ge \ell'\right\} + \\ & \sum_{t=1}^{n} \mathbb{I}\left\{\bar{Y}_{i,S_i(t-1)} - \sqrt{\frac{2\log(t)}{S_i(t-1)}} \ge \mu_i,\ S_i(t-1) \ge \ell'\right\} + \\ & \sum_{t=1}^{n} \mathbb{I}\left\{\mu_i + 2\sqrt{\frac{2\log(t)}{S_i(t-1)}} > \mu_{i^*},\ S_i(t-1) \ge \ell'\right\}. \end{aligned} \tag{4.10}$$

By removing the condition $S_i(t-1) \ge \ell'$, the summations can only grow. Therefore,

$$\begin{aligned}(4.10) \le \; & \sum_{t=1}^{n}\sum_{s=1}^{t-1} \mathbb{I}\left\{\bar{Y}_{i^*,s} + \sqrt{\frac{2\log(t)}{s}} \le \mu_{i^*}\right\} + \\ & \sum_{t=1}^{n}\sum_{s=\ell'}^{t-1} \mathbb{I}\left\{\bar{Y}_{i,s} - \sqrt{\frac{2\log(t)}{s}} \ge \mu_i\right\} + \\ & \sum_{t=1}^{n} \mathbb{I}\left\{2\sqrt{\frac{2\log(t)}{S_i(t-1)}} > \Delta_i,\ S_i(t-1) \ge \ell'\right\}. \end{aligned}$$

Choosing $\ell' = \left\lceil \frac{8\log(n)}{\Delta_i^2} \right\rceil$ makes the events in the last summation above impossible, because $S_i(t-1) \ge \ell' \ge \frac{8\log(n)}{\Delta_i^2}$ which implies $2\sqrt{\frac{2\log(t)}{S_i(t-1)}} \le 2\sqrt{\frac{2\log(n)}{\ell'}} \le \Delta_i$. Therefore, combining with (4.7), we can write

$$\begin{aligned} T_i(n) \;\le\; & \left\lceil \frac{8\log(n)}{\Delta_i^2} \right\rceil + \tau^*_{i,n} + \\ & \sum_{t=1}^{n}\sum_{s=1}^{t}\left(\mathbb{I}\left\{\bar{Y}_{i^*,s} \; + \; \sqrt{\frac{2\log(t)}{s}} \le \mu_{i^*}\right\} + \mathbb{I}\left\{\bar{Y}_{i,s} - \sqrt{\frac{2\log(t)}{s}} \ge \mu_i\right\}\right). \end{aligned}$$

Taking expectation gives

$$\begin{aligned} \mathbb{E}\left[T_i(n)\right] \;\le\; & \left\lceil \frac{8\log(t)}{\Delta_i^2} \right\rceil + \mathbb{E}\left[\tau^*_{i,n}\right] + \\ & \sum_{t=1}^{n}\sum_{s=1}^{t}\left(\mathbb{P}\left\{\bar{Y}_{i^*,s} \; + \; \sqrt{\frac{2\log(t)}{s}} \le \mu_{i^*}\right\} + \mathbb{P}\left\{\bar{Y}_{i,s} - \sqrt{\frac{2\log(t)}{s}} \ge \mu_i\right\}\right). \end{aligned}$$

We can use the concentration inequality used in the original analysis, namely Hoeffding's inequality (Lemma A.1), to bound each of the probabilities in the summation:

$$\begin{aligned} & \mathbb{P}\left\{\bar{Y}_{i^*,s} \; + \; \sqrt{\frac{2\log(t)}{s}} \le \mu_{i^*}\right\} \le e^{-4\log(t)} = t^{-4}, \\ & \mathbb{P}\left\{\bar{Y}_{i,s} - \sqrt{\frac{2\log(t)}{s}} \ge \mu_i\right\} \le e^{-4\log(t)} = t^{-4}. \end{aligned}$$

Therefore, we have

$$\begin{aligned}\mathbb{E}\left[T_i(n)\right] &\leq \left\lceil \frac{8\log(t)}{\Delta_i^2} \right\rceil + \mathbb{E}\left[\tau_{i,n}^*\right] + \sum_{t=1}^{\infty} 2t^{-3} \\ &\leq \frac{8\log(n)}{\Delta_i^2} + 1 + \mathbb{E}\left[\tau_{i,n}^*\right] + 2\zeta(3),\end{aligned}$$

where $\zeta(3) < 1.21$ is the Riemann Zeta function[1]. Combining with (4.1) proves the theorem. □

4.3.3 KL-UCB under delayed feedback

In this section, we show an example when the scheme above may need more work to apply, and produce slightly weaker results. The problem here is that it is not always a trivial task to reuse the exact same analysis of the original, non-delayed algorithm for the delayed setting. Nevertheless, it is still possible to get meaningful guarantees in the case of upper confidence bounds of the KL-UCB algorithm (Garivier and Cappé, 2011) under delayed feedback.

Recall from Chapter 2 that for each action i, the upper confidence bound used by KL-UCB at time t is $B_{i,T_i(t-1),t}$ where

$$B_{i,s,t} = \max\left\{q \in [\bar{Y}_{i,s}, 1] : s\,\mathbf{d}(\bar{Y}_{i,s}, q) \leq \log(t) + 3\log(\log(t))\right\} \tag{4.11}$$

with $\mathbf{d}(p,q) = p\log(p/q) + (1-p)\log((1-p)/(1-q))$ being the KL-divergence of two Bernoulli random variables with parameters p and q. According to the method of Section 4.3.1, in the delayed setting we use upper confidence bounds $B_{i,S_i(t-1),t}$, and call this algorithm the Delayed-KL-UCB algorithm. We can then prove the following theorem using the general scheme of Section 4.3.1. Here again, the penalty in the regret is additive compared to the non-delayed setting of Theorem 2.2.

Theorem 4.4. *For any $\epsilon > 0$, the expected regret of the Delayed-KL-UCB algorithm after n time steps is bounded by*

$$\begin{aligned}\mathbb{E}\left[R_n\right] \leq \sum_{i:\Delta_i>0} \Delta_i \left(\frac{\log(n)}{\mathbf{d}(\mu_i, \mu_{i^*})}(1+\epsilon)C_1 \log(\log(n)) \right) + \\ + \sum_{i=1}^{K} \Delta_i \left(\frac{C_2(\epsilon)}{n^{\beta(\epsilon)}} \mathbb{E}\left[\tau_{i,n}^*\right] + \mathbb{E}\left[\tau_{i,n}^*\right] + 1 \right),\end{aligned}$$

where C_1, C_2, and β are the same as those in Theorem 2.2, and $\tau_{i,n}^* = \max_{1\leq t\leq n} \tau_{i,t}$.

In this case, it is not as easy to repeat the same analysis as in the case of UCB1. The problem is that the original, non-delayed analysis, when bounding the probability of the event in the right hand side of (4.7), makes use of not only the number of samples, but also the fact that the number of samples equals the number of trials, $T_i(t-1)$. This makes it possible to bound one of the summations in the analysis by observing that $T_i(t)$ increases after each play of action i. In the delayed-case, however, the number of samples at time step t is $S_i(t-1)$, which does not grow at

[1] For properties and theory of the Riemann Zeta function, see the book of Titchmarsh (1987)

each trial but may rather stay the same for as many as $\tau^*_{i,n}$ trials. This can introduce a multiplicative factor of $\tau^*_{i,n}$ into the bound. Here we avoid that effect by factoring out the logarithmic term of the bound earlier in the proof, so that the delay term multiplies only the constant term of the original bound. The other ideas of the analysis are the same as those in the original proof.

To prove the theorem, we need some auxiliary results. Define, as in the original analysis

$$\mathbf{d}^+(x,y) = \mathbf{d}(x,y)\mathbb{I}\left\{x < y\right\}.$$

Then we will have the following lemma, which is an adapted version of Lemma 7 of Garivier and Cappé (2011). The proof follows the same idea as the original, except for having $S_i(t-1) \geq \ell'$ which reduces the probability of the event.

Lemma 4.5. *For any* $n \geq 1$,

$$\sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, \mu_{i^*} \leq B_{i^*,S_{i^*}(t-1),t},\ S_i(t-1) \geq \ell'\right\} \leq$$
$$\tau^*_{i,n} \sum_{s=\ell'}^{n} \mathbb{I}\left\{s\mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) < \log(n) + 3\log(\log(n))\right\}.$$

Proof. We start in the same way as the original proof. Note that $\mathbf{d}^+(p,q)$ is non-decreasing in its second parameter, and that $I_t = i$ and $\mu_{i^*} \leq B_{i^*,S_{i^*}(t-1),t}$ together imply $B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t} \geq \mu_{i^*}$, which in turn gives

$$\mathbf{d}^+(\bar{Y}_{i,S_i(t-1)}, \mu_{i^*}) \leq \mathbf{d}(\bar{Y}_{i,S_i(t-1)}, B_{i,S_i(t-1),t}) \leq \frac{\log(t) + 3\log(\log(t))}{S_i(t-1)}.$$

Therefore, we have

$$\begin{aligned}
&\sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, \mu_{i^*} \leq B_{i^*,S_{i^*}(t-1),t},\ t > S_i(t-1) \geq \ell'\right\} \\
&\quad\leq \sum_{t=\ell'}^{n} \mathbb{I}\left\{I_t = i, S_i(t-1)\mathbf{d}^+(\bar{Y}_{i,S_i(t-1)}, \mu_{i^*}) \leq \log(t) + 3\log(\log(t)),\ S_i(t-1) \geq \ell'\right\} \\
&\quad\leq \sum_{t=\ell'}^{n} \mathbb{I}\left\{I_t = i, S_i(t-1)\mathbf{d}^+(\bar{Y}_{i,S_i(t-1)}, \mu_{i^*}) \leq \log(n) + 3\log(\log(n)),\ S_i(t-1) \geq \ell'\right\} \\
&\quad\leq \sum_{t=\ell'}^{n} \sum_{s=\ell'}^{t} \mathbb{I}\left\{I_t = i, S_i(t-1) = s\right\} \times \mathbb{I}\left\{s\mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) \leq \log(n) + 3\log(\log(n))\right\} \\
&\quad= \sum_{s=\ell'}^{n} \sum_{t=s}^{n} \mathbb{I}\left\{I_t = i, S_i(t-1) = s\right\} \times \mathbb{I}\left\{s\mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) \leq \log(n) + 3\log(\log(n))\right\} \\
&\quad= \sum_{s=\ell'}^{n} \mathbb{I}\left\{s\mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) \leq \log(n) + 3\log(\log(n))\right\} \times \left(\sum_{t=s}^{n} \mathbb{I}\left\{I_t = i, S_i(t-1) = s\right\}\right).
\end{aligned}$$

But note that the second summation is bounded by $\tau^*_{i,n}$, because for each s, there cannot be more than $\tau^*_{i,n}$ time steps at which action i is played while $S_i(t) = s$ stays constant, otherwise we would have $T_i(t') - S_i(t') > \tau^*_{i,n}$ for some $t' \in \{s, \ldots, n\}$, which is not possible due to Lemma 4.4. Substituting this bound in the last expression proves the lemma. □

We also need the following two results from the original paper by Garivier and Cappé (2011).

Theorem 4.5 (Theorem 10 of Garivier and Cappé (2011))**.** *Let* $\{Y_t\}, t \geq 1$ *be a sequence of independent random variables bounded in* $[0,1]$*, with common expectation* $\mu = \mathbb{E}[Y_t]$*. Consider a sequence* $\{\epsilon_t\}, t \geq 1$ *of Bernoulli variables such that for all* $t > 0$*,* ϵ_t *is a random function of* $Y_1, \ldots, Y_{t-1}$ [2]*, and is independent of* $Y_s, s \geq t$*. Let* $\delta > 0$ *and for every* $1 \leq t \leq n$*, let*

$$S_t = \sum_{s=1}^{t} \epsilon_s \quad \textit{and} \quad \bar{Y}_t = \frac{\sum_{s=1}^{t} \epsilon_s Y_s}{S_t},$$

with $\bar{Y}_t = 0$ *when* $S_t = 0$*, and*

$$B_n = \operatorname{argmax}\left\{q > \bar{Y}_n : S_n \mathbf{d}(\bar{Y}_n, q) \leq \delta\right\}.$$

Then

$$\mathbb{P}\{B_n < \mu\} \leq e\lceil \delta \log(n) \rceil e^{-\delta}.$$

Lemma 4.6 (Lemma 8 of Garivier and Cappé (2011))**.** *For a suboptimal action* i*, for every* $\epsilon > 0$*, let* $K_n = \left\lfloor \frac{1+\epsilon}{\mathbf{d}(\mu_i, \mu_{i^*})} \left(\log(n) + 3\log(\log(n)) \right) \right\rfloor$*. Then there exist* $C_2(\epsilon) > 0$ *and* $\beta(\epsilon) > 0$ *such that*

$$\sum_{s=K_n+1}^{\infty} \mathbb{P}\left\{ \mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) < \frac{\mathbf{d}(\mu_i, \mu_{i^*})}{1+\epsilon} \right\} \leq \frac{C_2(\epsilon)}{n^{\beta(\epsilon)}}.$$

Now, we are ready to prove Theorem 4.4.

Proof of Theorem 4.4. For an action i, bounding the terms in (4.7) gives

$$\begin{aligned}
&\sum_{t=1}^{n} \mathbb{I}\left\{I_t = i, B_{i,S_i(t-1),t} \geq B_{i^*,S_{i^*}(t-1),t},\ S_i(t-1) \geq \ell'\right\} \\
&\quad \leq \sum_{t=1}^{n} \mathbb{I}\left\{B_{i^*,S_{i^*}(t-1),t} < \mu_{i^*}\right\} \\
&\qquad + \sum_{t=1}^{n} \mathbb{I}\left\{I_t = i,\ \mu_{i^*} \leq B_{i^*,S_{i^*}(t-1),t},\ S_i(t-1) \geq \ell'\right\} \\
&\quad \leq \sum_{t=1}^{n} \mathbb{I}\left\{B_{i^*,S_{i^*}(t-1),t} < \mu_{i^*}\right\} \\
&\qquad + \tau_{i,n}^* \sum_{s=\ell'}^{n} \mathbb{I}\left\{s\mathbf{d}^+(\bar{Y}_{i,s}, \mu_{i^*}) < \log(n) + 3\log(\log(n))\right\}, \qquad (4.12)
\end{aligned}$$

where the last inequality follows from Lemma 4.5. Let

$$K_n = \left\lfloor \frac{1+\epsilon}{\mathbf{d}(\mu_i, \mu_{i^*})} \left(\log(n) + 3\log(\log(n)) \right) \right\rfloor \qquad (4.13)$$

and

$$\ell' = 1 + K_n. \qquad (4.14)$$

[2]That is, a function of $Y_1, \ldots, Y_{t-1}$ together with possibly an extra, independent randomization.

Then we have:

$$\begin{aligned}
&\sum_{s=\ell'}^{n} \mathbb{I}\left\{s\mathbf{d}^{+}(\bar{Y}_{i,s},\mu_{i^*}) \leq \log(n) + 3\log(\log(n))\right\} \\
&\qquad \leq \sum_{s=K_n+1}^{\infty} \mathbb{I}\left\{(K_n+1)\mathbf{d}^{+}(\bar{Y}_{i,s},\mu_{i^*}) \leq \log(n) + 3\log(\log(n))\right\} \\
&\qquad \leq \sum_{s=K_n+1}^{\infty} \mathbb{I}\left\{\mathbf{d}^{+}(\bar{Y}_{i,s},\mu_{i^*}) < \frac{\mathbf{d}(\mu_i,\mu_{i^*})}{1+\epsilon}\right\}. \qquad (4.15)
\end{aligned}$$

Putting inequalities (4.13) trough (4.15) back into inequality (4.12), from (4.7) we get:

$$\begin{aligned}
\mathbb{E}\left[T_i(n)\right] \leq\ & \frac{1+\epsilon}{\mathbf{d}(\mu_i,\mu_{i^*})}\Bigg(\log(n) + 3\log(\log(n))\Bigg) + \mathbb{E}\left[\tau^*_{i,n}\right] + 1 + \\
& \sum_{t=1}^{n} \mathbb{P}\left\{B_{i^*,S_{i^*}(t-1),t} < \mu_{i^*}\right\} + \mathbb{E}\left[\tau^*_{i,n}\right] \sum_{s=K_n+1}^{\infty} \mathbb{P}\left\{\mathbf{d}^{+}(\bar{Y}_{i,s},\mu_{i^*}) < \frac{\mathbf{d}(\mu_i,\mu_{i^*})}{1+\epsilon}\right\}.
\end{aligned}$$

where the last term is a result of the delays being independent of the rewards. The first summation can be bounded using Theorem 4.5, for which it suffices to set $\epsilon_t = 1, 1 \leq t \leq n$, and use the sequence of observed rewards $\{Y_{i,t}\}$ for the arm under consideration as the sequence $\{Y_t\}$ in the theorem. The second summation can be bounded by Lemma 4.6. Therefore, the expectation of the number of trials of the suboptimal action is bounded by:

$$\begin{aligned}
\mathbb{E}\left[T_i(n)\right] \leq & \frac{1+\epsilon}{\mathbf{d}(\mu_i,\mu_{i^*})}\Bigg(\log(n) + 3\log(\log(n))\Bigg) + C_1\log(\log(n)) + \\
& \frac{C_2(\epsilon)}{n^{\beta(\epsilon)}}\mathbb{E}\left[\tau^*_{i,n}\right] + \mathbb{E}\left[\tau^*_{i,n}\right] + 1.
\end{aligned}$$

Combining the above inequality with (4.1) finishes the proof. □

4.4 Lower bounds for fixed delays

To investigate how close our solutions in this chapter are to optimal, we would like to derive lower bounds that show that the minimum penalty possible should be additive in the delays. However, this is not a clear goal. The problem is that the asymptotic sense in which lower bounds and upper bounds *match* (as we are familiar with in the non-delayed setting) does not make much sense for an *additive* lower bound. In other words, the lower and upper bounds for the MAB problem, as we know them, have the same growth rate but do not match in constant factors. In a non-delayed setting, this is not much of a problem, because we are interested in showing the *rate* with which the regret goes to zero, which does not require knowledge of the specific constants. However, if we want to show the same matching growth rates for the delayed setting, and at the same time get an additive effect of delays, the constants will become important: we can show any additive bound for large enough n, simply by decreasing the constant of the first term (the term that grows with n), and adding as much delay term as we want.

To give a concrete example, suppose that we know of a lower bound of $c\sqrt{Kn}$ for the minimax regret of the non-delayed stochastic MAB problem. It would then be a lower bound for the delayed setting as well, since we cannot gain anything by observing late (or less) information. Now suppose that, under a constant delay τ, the time horizon n is large enough to have $\frac{c}{2}\sqrt{Kn} > K\tau$, that is, $n > \left(\frac{2}{c}\sqrt{K}\tau\right)^2$. We would then have $\frac{c}{2}\sqrt{Kn} + K\tau \leq c\sqrt{Kn}$, so for large enough n we would have an additive lower bound on the regret in the delayed setting! The bound, however, is obviously vacuous in the sense that it does not show anything about the additive *nature* of the effect of the delays.

Another approach to tackle this problem might be to show a relation between the expected regret of the delayed algorithm, $\mathbb{E}\left[R_n\right]$, and the exact value of the expected regret in the non-delayed problem, $\mathbb{E}\left[R'_n\right]$. This means trying to derive a mapping that, for every delayed MAB algorithm, gives a non-delayed MAB algorithm such that any run of the first algorithm can be associated with a hypothetical run of the second one, in such a way that the regret in the delayed setting can be written as the regret of the non-delayed algorithm in the corresponding non-delayed environment, plus a term that depends on the delays. This way one may be able to reason for general random delays as well, not only for fixed delays. This method, however, is not straightforward to use, and investigating it is left for future work.

Chapter 5

Experiments

In this chapter we empirically test the algorithms introduced in the previous sections, and verify the results therein. First we study the effect of the delays on the regret and choose a magnitude for the delay to run the rest of the experiments with. Then we test each of our four algorithms, namely UCB1, KL-UCB and our black-box style algorithm SBBD run with non-delayed UCB1 and non-delayed KL-UCB, respectively. We run each of the algorithms under four different delay distributions: constant, uniformly distributed, exponential, and Gaussian (truncated at zero). In these experiments our aim is to investigate the penalty in the regret compared to the non-delayed setting, and to see how tight our bounds for the algorithms in Chapter 4 are. We then compare the black-box approach to the white-box one, in terms of the difference in their regrets as well as the memory usage of the black-box algorithm.

5.1 General setup

All the experiments in this chapter were performed using the following settings. We use a two-armed stochastic multi armed bandit environment, with Bernoulli rewards of expectations 0.8 and 0.9 respectively. The reason to choose this instantiation of the MAB problem was that the reward structure was simple enough to let us distinguish the effects of the delay from those of the learning problem itself. One other interesting question may be the way the penalty in the regret scales with the number of arms. This study is postponed to future work.

Except for the first experiment in Section 5.2, each of the experiments in this chapter are run for a time horizon of $n = 1000$ time steps, and all the results are averaged over 1000 independent runs. Because of running time constraints, we were not able to use significantly larger time horizons. The delay configuration varies from experiment to experiment, which is described in its corresponding section.

We implemented the UCB1 and KL-UCB algorithms exactly as described by their corresponding authors. To break the ties between equal upper confidence bounds, we used an independent uniform randomization. This helps to distribute evenly the choices that occur at the beginning of runs when there are no rewards, instead of just choosing the action with the smallest index.

5.2 Regret penalty due to delays

First we run an experiment to help us choose a suitable magnitude for the delays which we want to use in the rest of the experiments. For this purpose, we run Delayed-UCB1 for $n = 10,000$ time steps under increasing constant delays. Figure 5.1 shows the results on a logarithmic time scale. The results are averaged over $2,000$ independent runs.

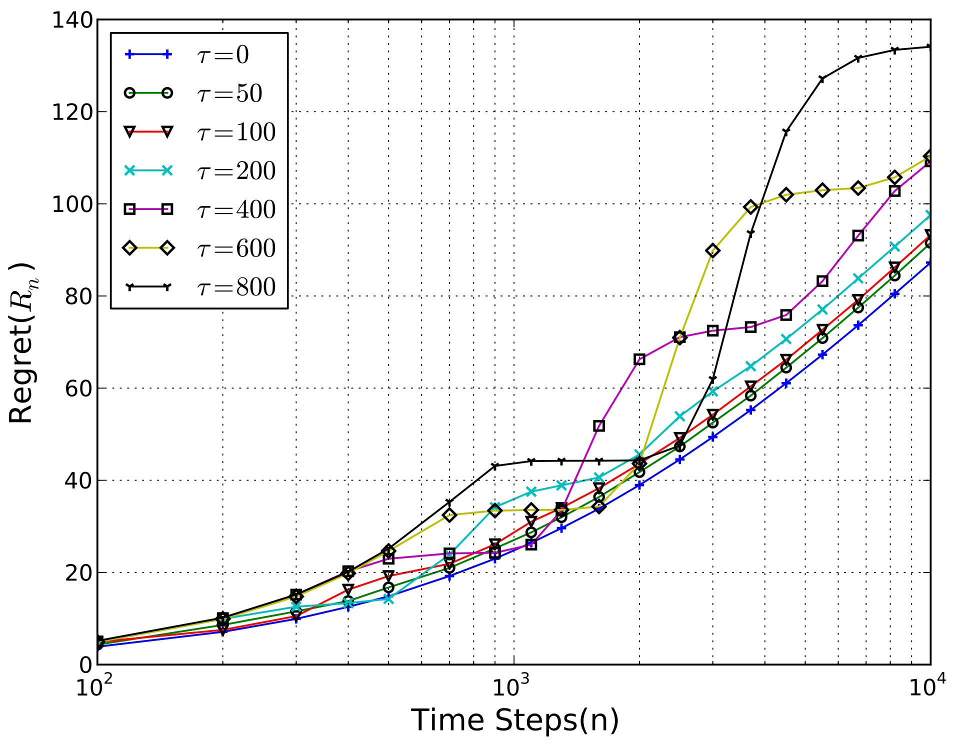


Figure 5.1: Regret of Delayed-UCB1 for different constant delays.

As the figure shows, the regret in the delayed setting passes some unstable phase before being stabilized and showing the real effect of the delay. In fact, these fluctuations were to be expected. These fluctuations are only a scaled version of the fluctuations over the first few rounds of non-delayed UCB1. In other words, for a constant delay τ, Delayed-UCB1 can not do better than random exploration in the first τ rounds. This causes the algorithm to compensate by doing more exploitation in the next rounds, thereby causing a sudden increase over τ time steps and then another τ rounds of very slowly growing regret, hence the delayed regret getting less than the non-delayed regret at some points (in fact, the idea of compacting the explorations to the first rounds of the game to get a small final regret is used to provide optimal MAB algorithms such as MOSS (Audibert and Bubeck, 2009)). The fluctuations take a while to decay completely, as the information resulting from the

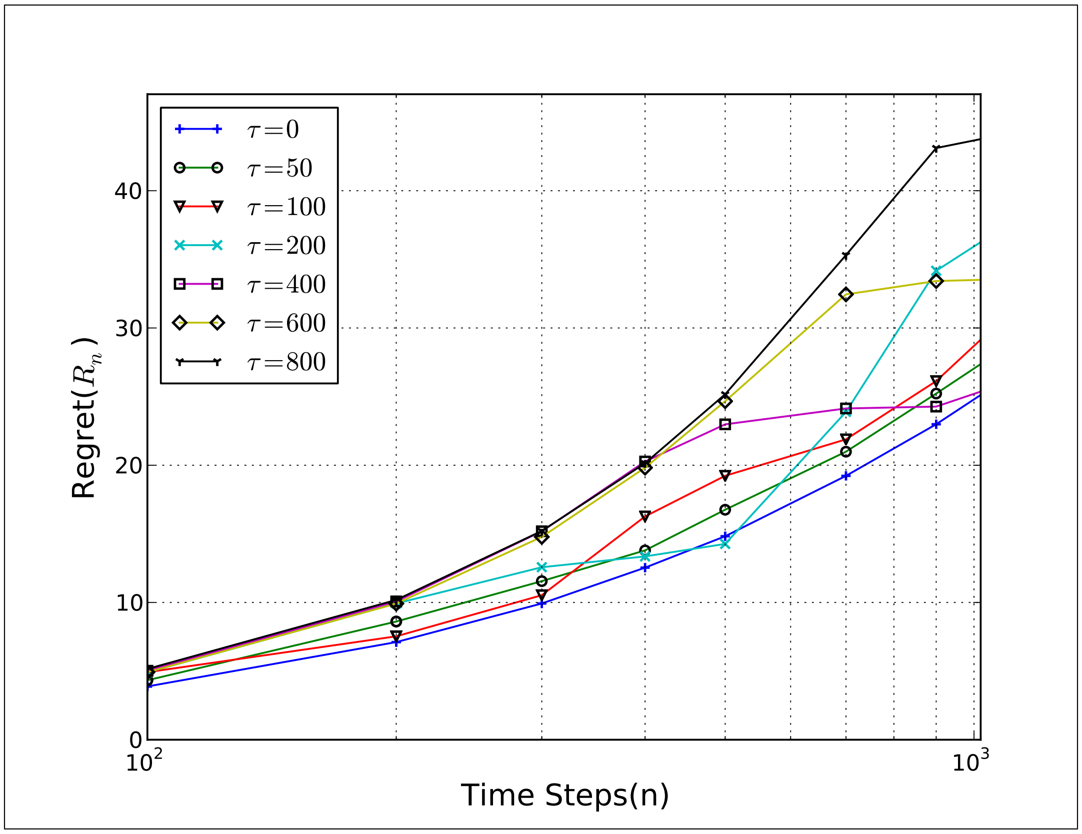


Figure 5.2: Regret of Delayed-UCB1 for different constant delays. The regret has stabilized at time $n = 1000$ for delays between 50 and 100.

actions the agent chooses influence his actions only after τ time steps.

While other experiments in this chapter use a time horizon of $n = 1000$, in this experiment we used a larger time horizon to better see whether the regret has stabilized by time step 1000 for a specific delay value. Figure 5.2 shows the same results by time step 1000. For the rest of our experiments, while the delay should be small enough to let us reach the stable time steps, it has to be large enough to let its effect be manifested in the regret. Therefore, according to the figures, a delay between 50 and 100 would be suitable for a time horizon of $n = 1000$. We did not simulate the effect for delays larger than 500, as by the explanation above as well as the observed behaviour of smaller delays, the effect we could see was not stabilized. In the rest of the simulations, we have used a delay of 75 time steps.

5.3 Effect of delay distributions

Having the average magnitude of the delay fixed, we were interested in the effect of different delay distributions on the regret. We run each of our four algorithms in four different delay conditions: constant delays, uniformly distributed delays, exponential delays, and delays with a Gaussian distribution. Figures 5.3 through 5.6 show the results for each of the four algorithms. Symbols

$U[0, 150]$, $Exp(1/\lambda = 75)$ and $N_+(75, 15)$ denote, respectively, the uniform distribution, the exponential distribution, and the truncated Gaussian distribution. The standard deviation of the Gaussian distribution before truncation was 15.

As seen in the figures, the average penalty of a fixed delay is usually an upper bound for that of other distributions, with the normal distribution being almost the same as the fixed delay. This is contrary to what is expected. For the uniform, exponential and Gaussian distributions, the penalty term $\mathbb{E}\left[\max_{1 \leq t \leq n} \tau_t\right]$ shown in Theorems 4.2, 4.3 and 4.4 is expected to grow as the time horizon n grows and to take larger values than its expectation $\mathbb{E}\left[\tau_t\right] = 75$[1]. The fact that this has not happened in the simulations suggests that the penalty term shown in the aforementioned theorems may be an overestimation and a better way to bound the penalty may exist.

The difference of UCB1 and KL-UCB is also noteworthy in these experiments. Figure 5.7 compares the regret of these two algorithms at time step $n = 1000$. The penalty in regret that UCB1 has suffered here seems to be smaller than KL-UCB. We are not sure whether the extra term in the analysis of KL-UCB in Theorem 4.4 has any effect here. This, however, does not necessarily mean that UCB1 is more robust to delays, or more suitable to use in a delayed environment, since the regret of KL-UCB under delays is still lower than that of UCB1. In other words, not only the difference in the delayed and non-delayed regrets should be taken into account, we also need to check the total regret.

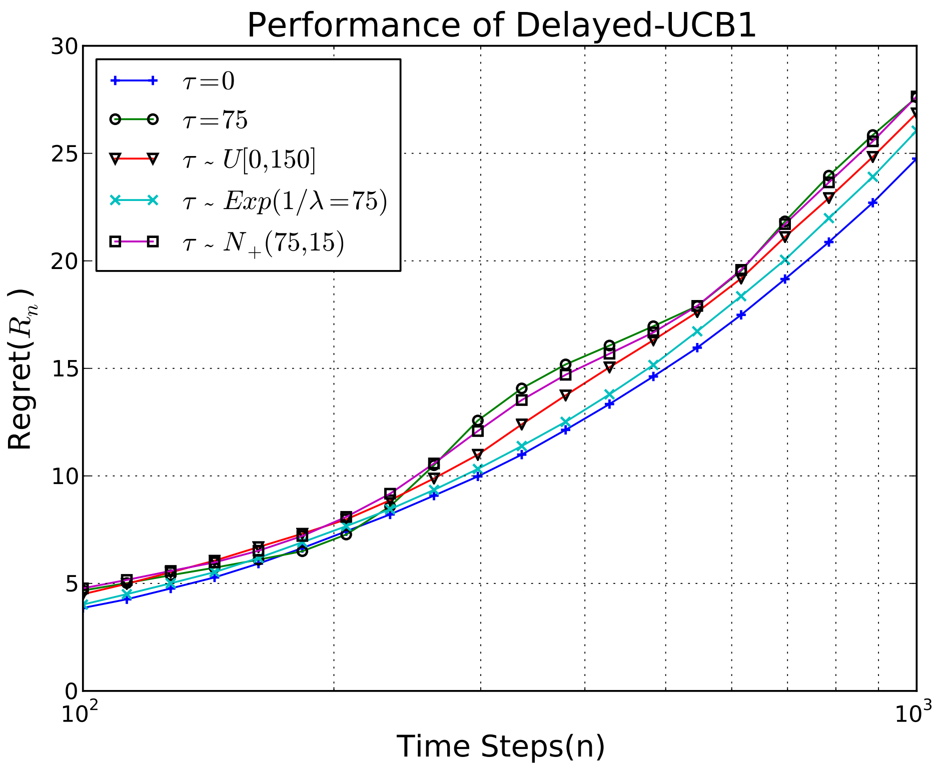
5.4 Black-box vs. white-box

The question that we want to investigate in this section is how different the performances of the delayed UCB algorithms and their black-box versions are. To this end, we run and compare Delayed-UCB with the SBBD algorithm using non-delayed UCB as its base algorithms, as well as Delayed-KL-UCB with SBBD run with non-delayed KL-UCB. Figures 5.8 and 5.9 show the results for UCB1, and Figures 5.10 and 5.11 and KL-UCB, respectively.

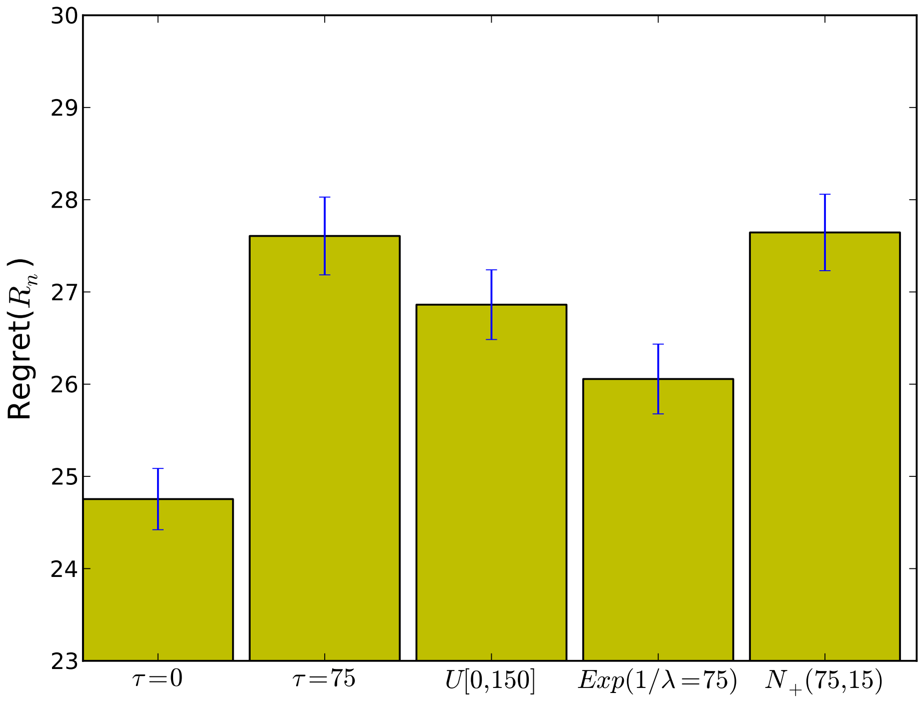
For UCB1, the regrets of the two algorithms are very similar. For KL-UCB, while the situation is similar (note the different scales of the y axis), the difference vanishes more slowly. The more interesting case is that the black-box versions seems to have worked better than the modified version. The difference, however, is insignificant for all of the simulations, as a paired t-test was not able to reject the null-hypothesis of equal means with 95% confidence.

The question that remains is how large the extra memory requirement of the black-box algorithm is. This is shown in Figure 5.12. Each plot shows the total number of items in the queues of the algorithm at the end of each time step. The experiment shows that the average number of items in the queue is much smaller than τ_n^* (except in the case of fixed delays, for which $\tau_n^* = \mathbb{E}\left[\tau\right] = \min_{1 \leq t \leq b} \tau_t$) and therefore the black-box algorithm is still a feasible solution.

[1]The expectation of the truncated Gaussian is slightly larger than 75, because of the one-sided truncation.

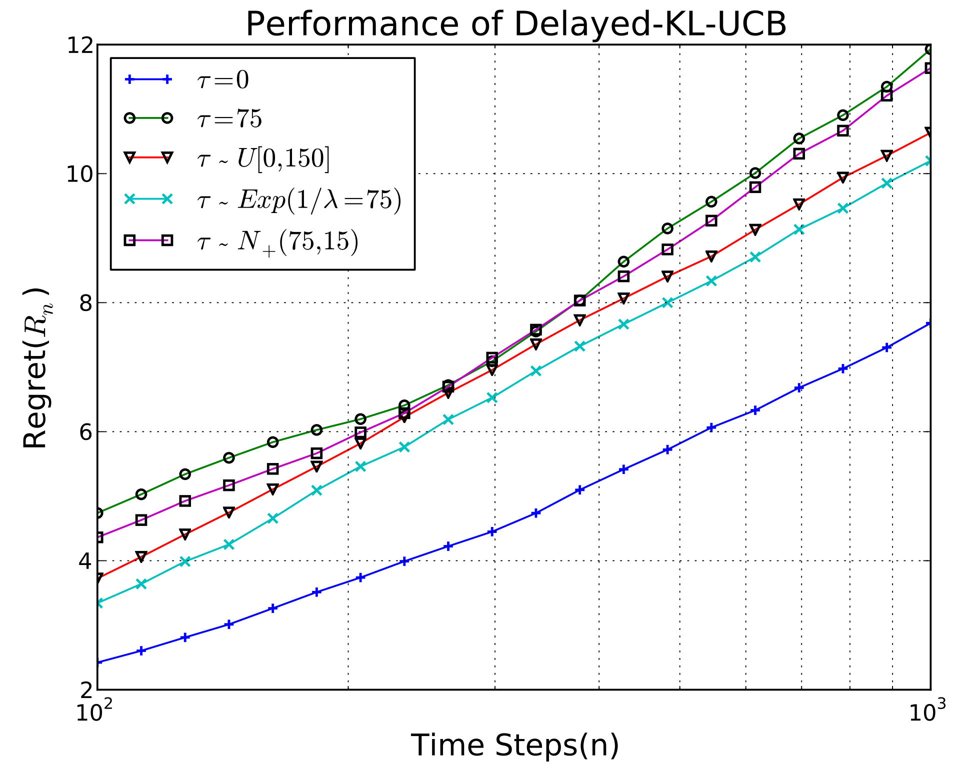


(a)

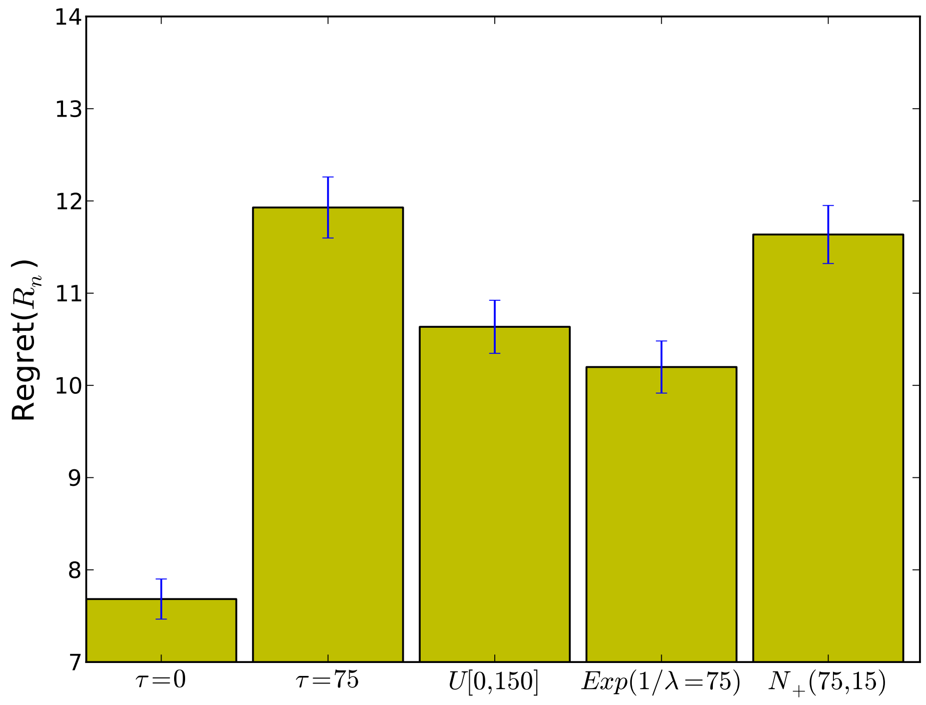


(b)

Figure 5.3: Regret of Delayed-UCB1 on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean.

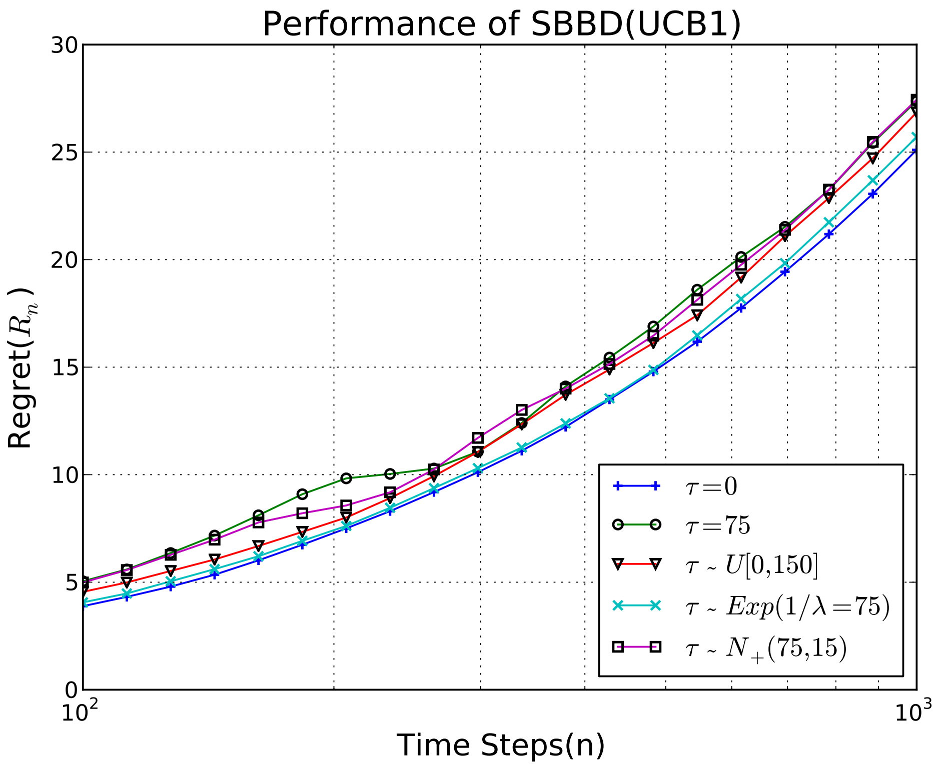


(a)

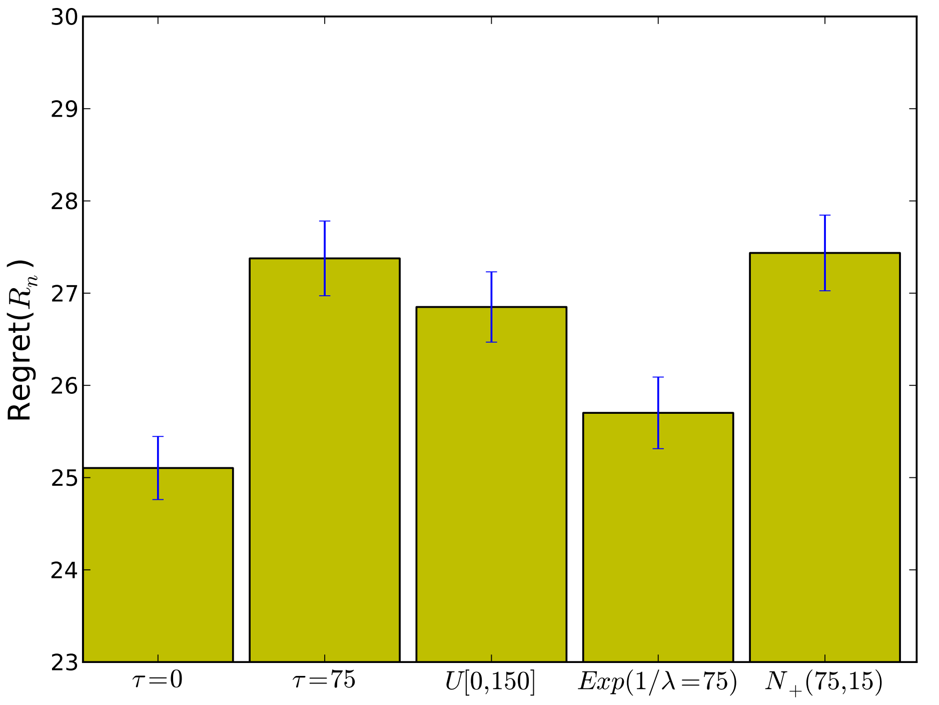


(b)

Figure 5.4: Regret of Delayed-KL-UCB on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean.

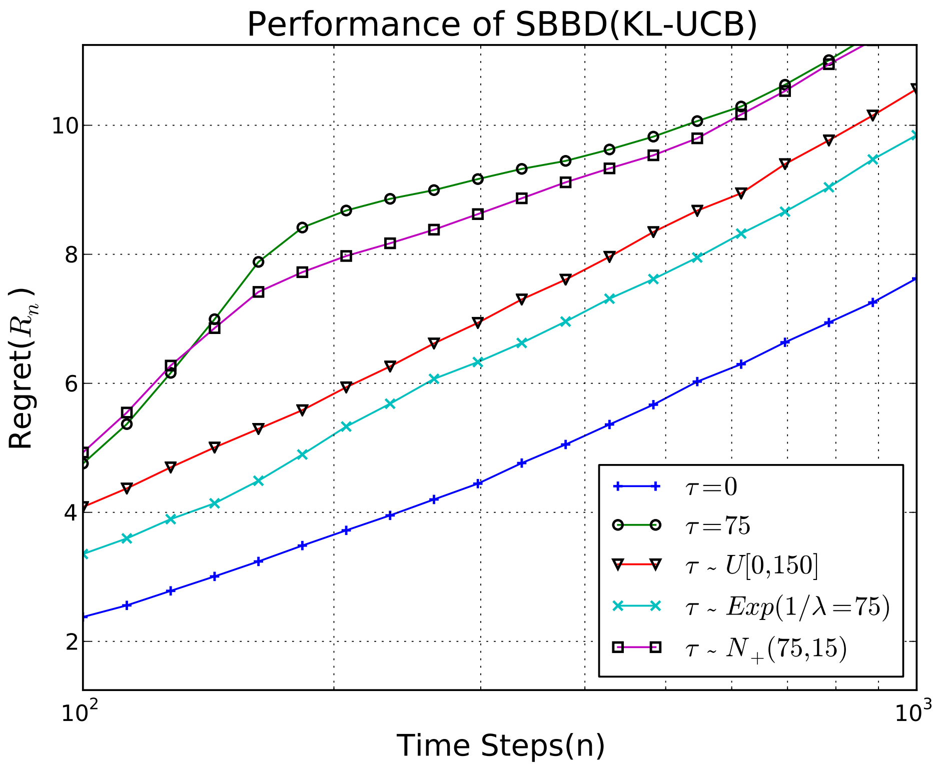


(a)

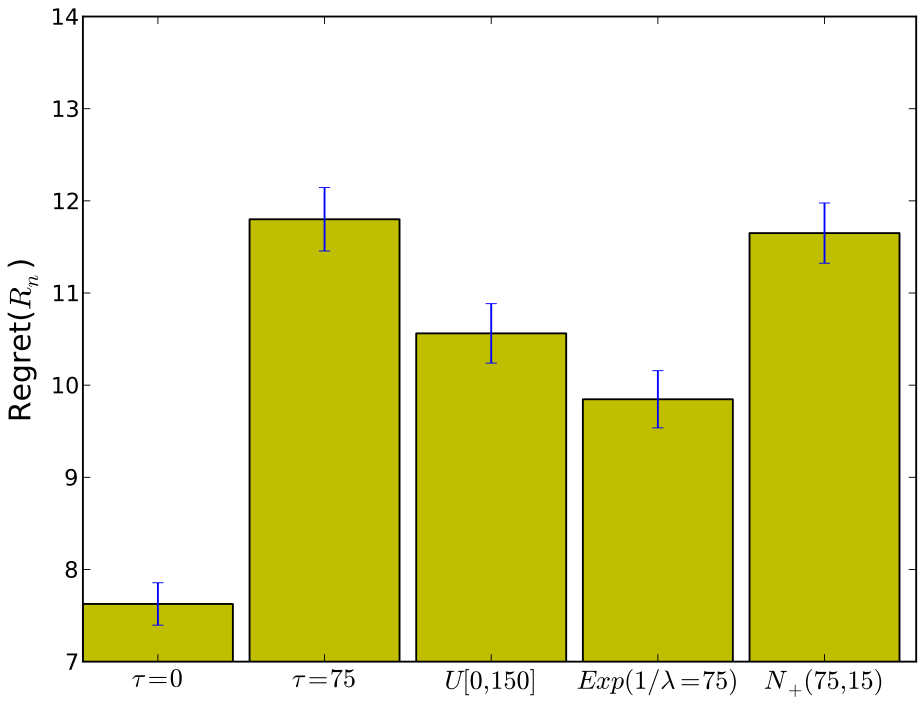


(b)

Figure 5.5: Regret of SBBD, with UCB1 as base algorithm, on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean.



(a)



(b)

Figure 5.6: Regret of SBBD, with KL-UCB as base algorithm, on the four different delay distributions compared with the non-delayed setting. (a) Regret as a function of time. (b) Regret at time step $n = 1000$. Error bars are two times the standard error in mean.

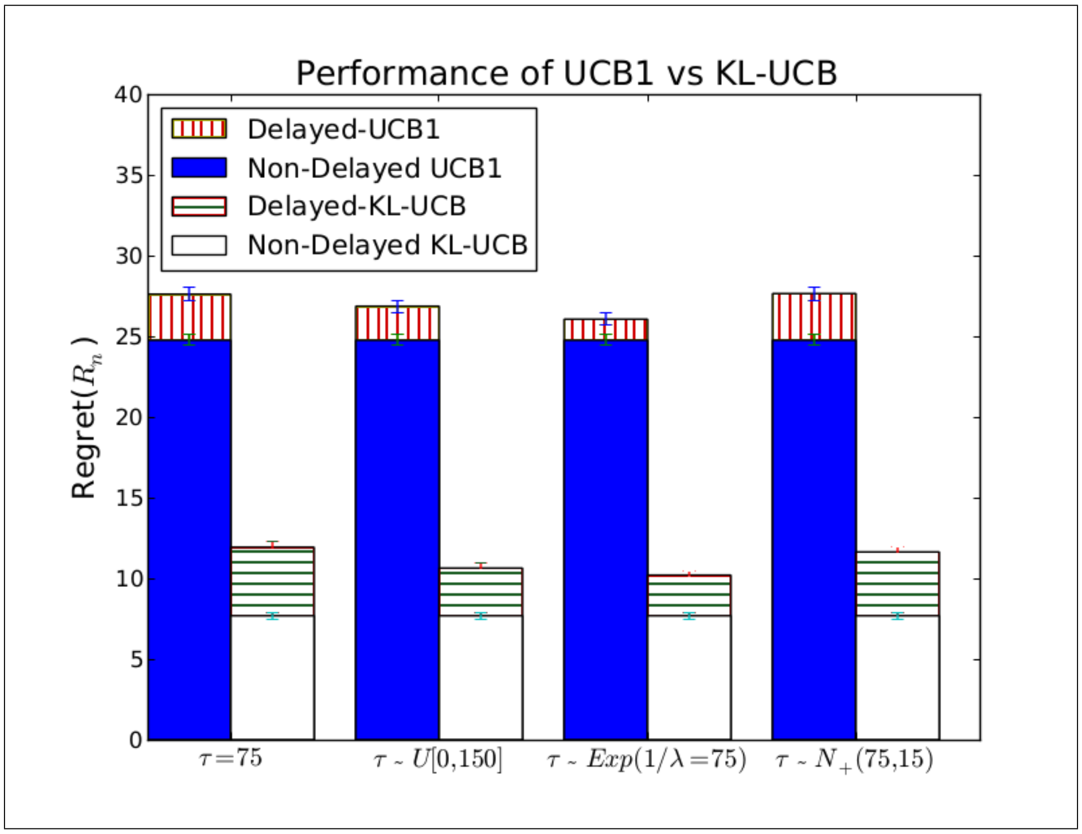


Figure 5.7: The growth in the regret of UCB1 vs KL-UCB at time step $n = 1000$. Error bars are two times the standard error in mean. KL-UCB seems to be more sensitive to delays, though its total regret is always significantly below that of UCB1.

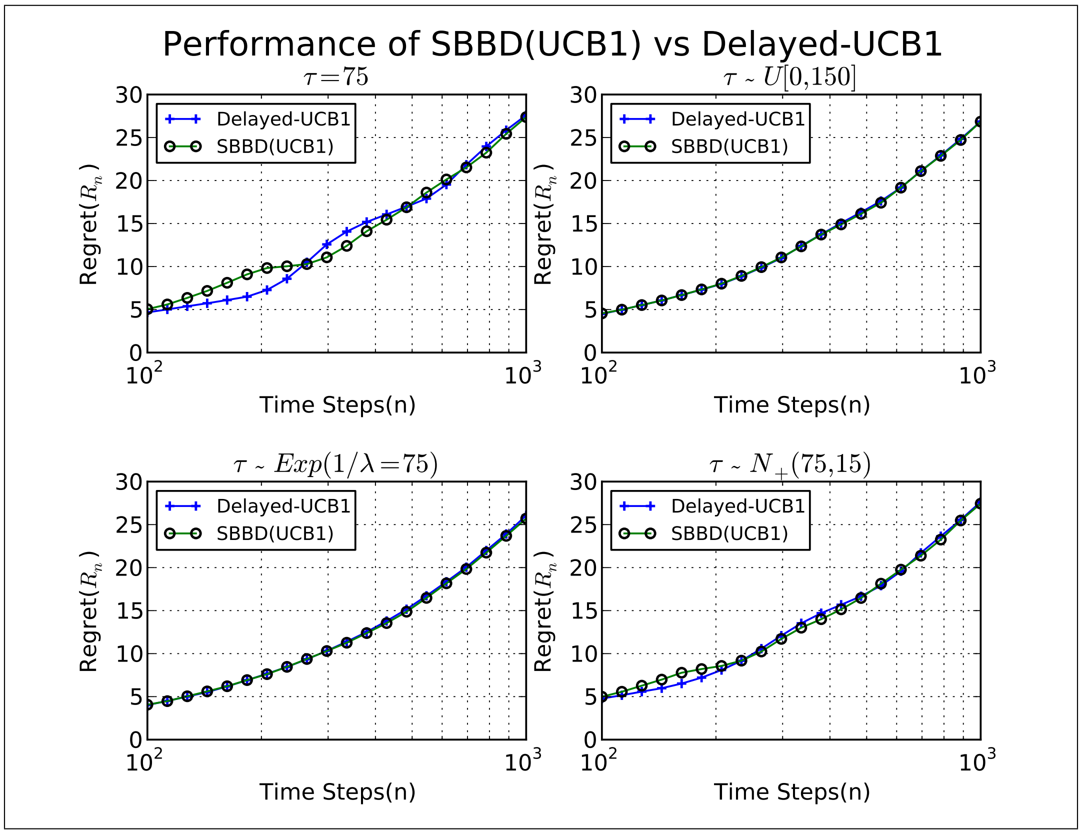


Figure 5.8: Regret of Delayed-UCB1 vs SBBD, with UCB1 as base algorithm, as a function of time. The regret curves of the two algorithms are closely matching each othe except for the case of constant delays, when the regret of SBBD(UCB1) is first larger than that of UCB1, but then the ordering swaps and eventually the curves converge.

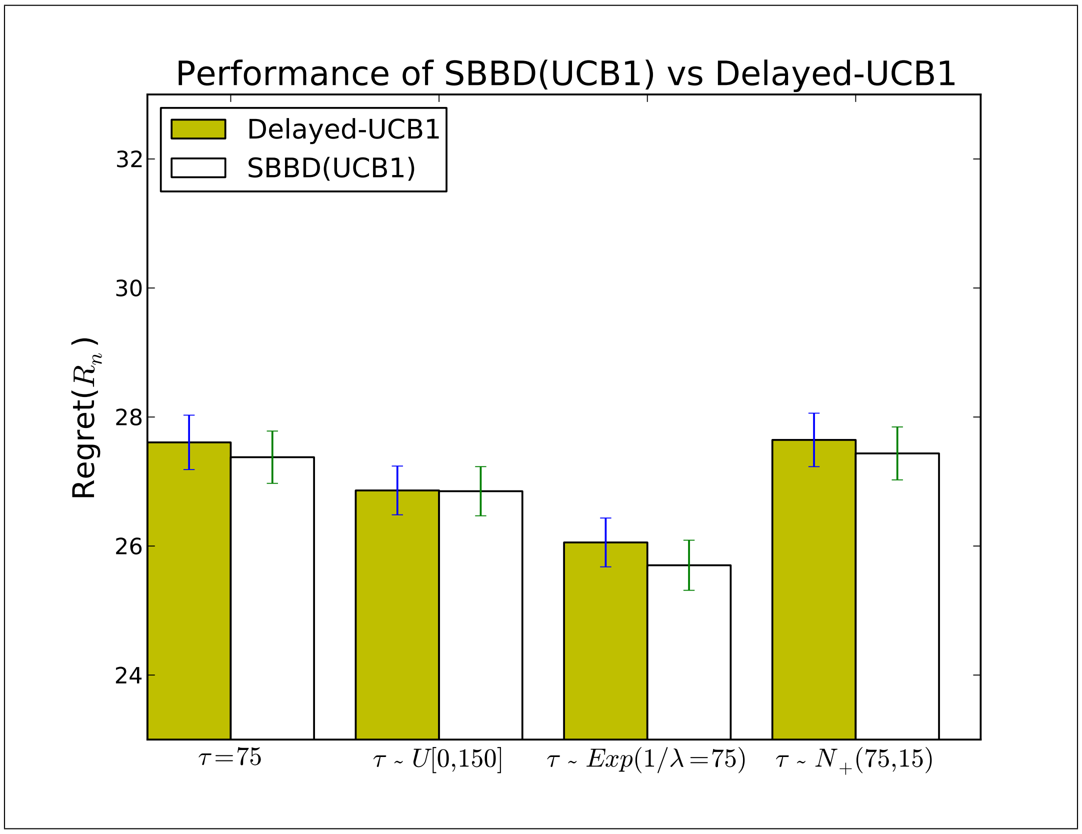


Figure 5.9: Regret of Delayed-UCB1 vs SBBD, with UCB1 as base algorithm, at time step $n = 1000$. Error bars are two times the standard error in mean. The regret of the black-box SBBD(UCB1) algorithm is slightly smaller, but this is not statistically significant.

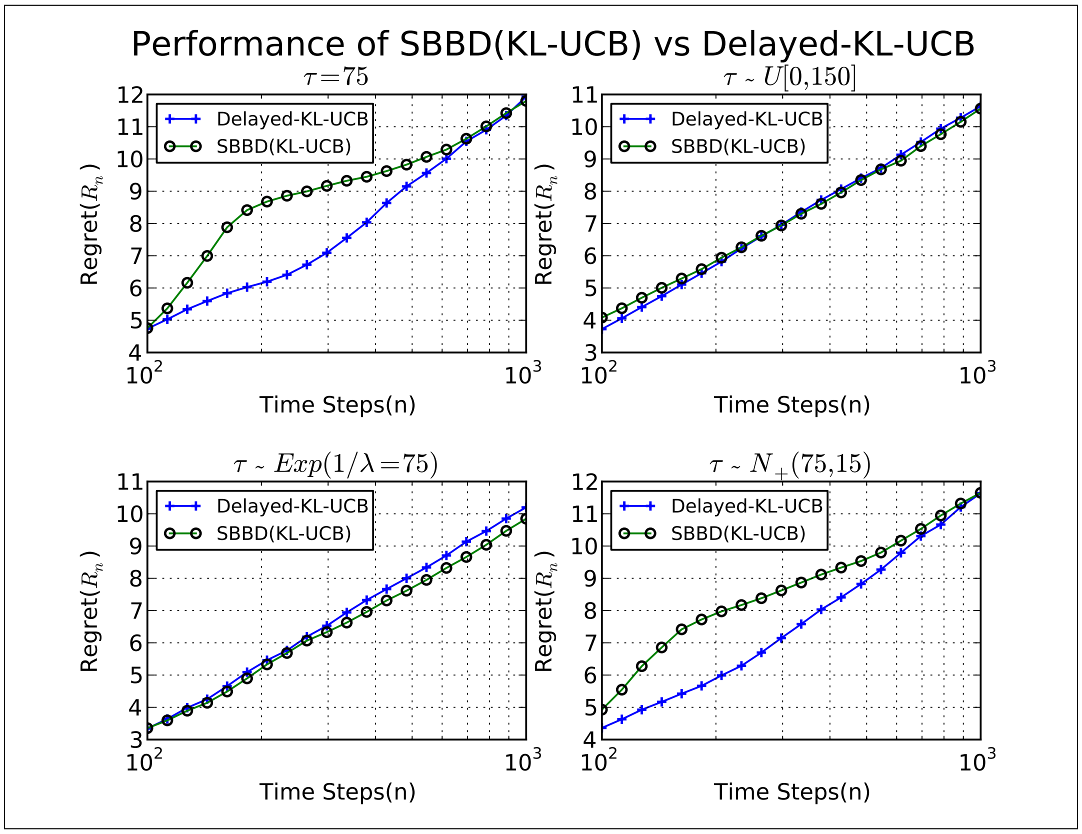


Figure 5.10: Regret of Delayed-KL-UCB vs SBBD, with KL-UCB as base algorithm, as a function of time. As opposed to UCB1, the regret of the black-box method, SBBD(KL-UCB) is at times significantly different with the regret of Delayed-KL-UCB.

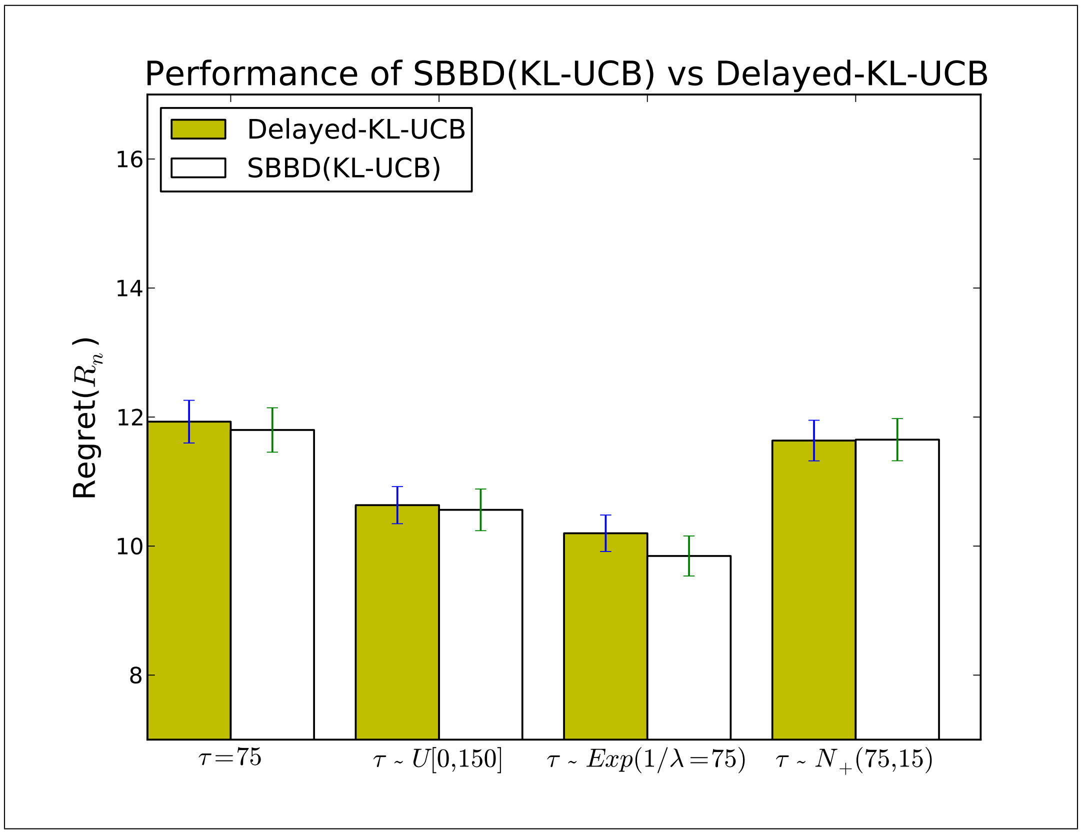


Figure 5.11: Regret of Delayed-KL-UCB vs SBBD, with KL-UCB as base algorithm, at time step $n = 1000$. Error bars are two times the standard error in mean. As in the case of UCB1, the regret of the black-box algorithm, SBBD(KL-UCB) is slightly better than the regret of Delayed-KL-UCB, but again the difference is not statistically significant.

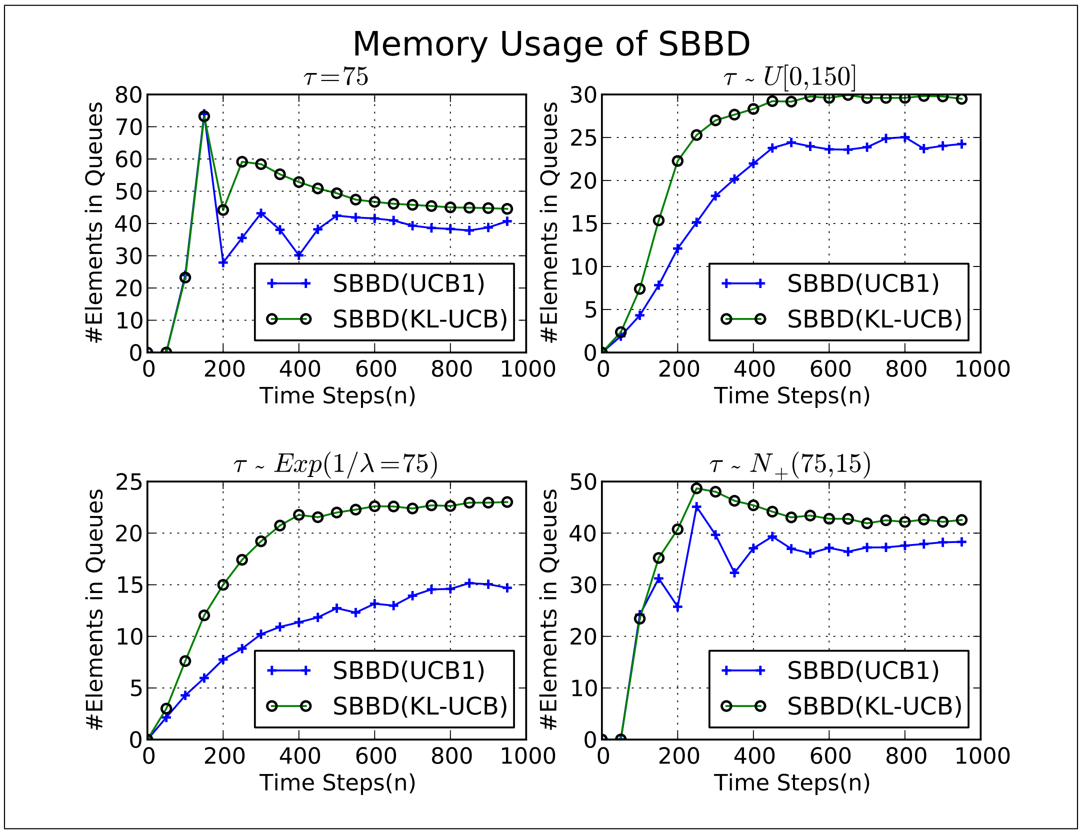


Figure 5.12: Memory usage of the buffers of SBBD under four different delay distributions, run separately with KL-UCB or UCB1 as the base algorithm. After an initial transient phase, the memory usage stabilizes.

Chapter 6

Conclusion

In this thesis, we studied the multi-armed bandit problem under delayed feedback, and provided algorithms for both the adversarial and the stochastic reward settings. We analyzed the regret penalty arising from delays for both of these settings, and showed a multiplicative penalty in the general adversarial case and an additive penalty in the stochastic setting.

Our proposed algorithms fall into two categories: black-box algorithms and white-box algorithms. Our two black-box style algorithms, SBBD and ABBD, target the stochastic and adversarial MAB problems, respectively. These algorithms work by using a non-delayed base MAB algorithm and transforming the delayed problem to one or more instances of the non-delayed problem. Apart from the memory requirements, for which we only have a large upper bound at the moment, the black-box algorithms have many useful features, such as the ability to reuse already implemented and tested non-delayed MAB algorithms for the delayed problem, as well as easily extending any improvement in the non-delayed setting to the delayed setting. The white-box algorithms proposed use upper-confidence bounds similar to the ones used in the non-delayed setting, but operate only on the observed rewards. We show how one can reuse the mechanics of the analysis of UCB-type algorithms in the non-delayed setting in order to obtain theoretical guarantees in the delayed setting.

We used a general formulation for delays which, while allowing random delays, does not make non-realistic assumptions about the delays. For the stochastic MAB problem, we require the delays to be independent of the rewards of the actions (and the randomization of the algorithm if the algorithm is randomized). For the adversarial setting, our proof of the multiplicative upper bound of our ABBD algorithm assumes that, at each time step, the delays of different actions are the same. This is not less general than the previous work, since previous work mostly considers either constant delays or the full information setting in which the delays of all the actions are the same at each time step. Our results do not require the delays to be bounded; infinite delays are allowed, as well.

In our experiments, we found a gap between the regret penalty observed in practice and our theoretical upper bound on the penalty. Furthermore, we tested the memory requirements of our black-box algorithms, which seems to be much smaller than our initial conservative upper bounds.

6.1 Future work

The exact way that the regret under delayed feedback depends on the delays is still not clear. An important required result to make this dependence clear is the development of a tight lower bound. Furthermore, as shown in our experiments, our additive delay terms in upper bounds on the regret is not the best we can get. Perhaps a better experiment to verify this conjecture is to run simulations with fixed delay mean and increasing delay variance. This can be achieved by using long-tail distributions such as the Inverse Gamma distribution. Using the two parameters of this distribution, one can keep the mean constant while increasing the variance. An interesting quantity that can be checked in these experiments is the distribution of *gaps*, i.e., the distribution of the number of consecutive time steps when the agent receives no feedback.

An interesting open question is to tighten the upper bound on the regret penalty in both the adversarial and the stochastic cases. This results in a better upper bound for the memory usage of our black-box algorithms, as well. Apart from the asymptotic behaviour of the regret with the time horizon n, the dependence on the number of actions K is also interesting to investigate.

Our algorithms can in principle be extended to other areas of online learning. Our ABBD algorithm, for example, works independent of the feedback or information structure of the game. SBBD can also be extended to other stochastic environments such as stochastic partial monitoring games (Bartók et al., 2010). Investigating the extension of SBBD to stochastic settings with side information is another open problem.

An interesting extension to the ABBD algorithm is to feed the black-box machines with all the rewards that arrive. While this is not straightforward for MAB agents without white-box modifications, it is in principle possible to do in a black-box fashion for *semi-bandit* algorithms (Mannor and Shamir, 2011).

In addition to the stochastic setting, it is interesting to show that we can reuse the proof mechanism for the adversarial setting. This has already been done by Neu et al. (2010) for constant delays and the EXP3 algorithm of Auer et al. (2003), but it remains to be generalized to arbitrary delays and to other algorithms. The FTRL algorithm (Shalev-Shwartz and Singer, 2007) should provide a suitable starting point, as EXP3 and INF (Audibert and Bubeck, 2009) are instances of FTRL. The results of Neu et al. (2010) on slowly changing policies may be useful here as well, and the exact analysis is the subject of future work.

Finally, removing our assumptions on the delays in the adversarial MAB problem, i.e., that the delays depend only on time but not on actions, remains to be studied. This is especially interesting because the statistical dependence structure of the problem becomes very complicated without that assumption.

Bibliography

A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011.

D. Agarwal, B.-C. Chen, and P. Elango, "Explore/exploit schemes for web content optimization," in *Data Mining, 2009. ICDM '09. Ninth IEEE International Conference on*, dec. 2009, pp. 1 –10.

T. W. Anderson, "Sequential analysis with delayed observations," *Journal of the American Statistical Association*, pp. 1006–1015, 1964.

J. Audibert and S. Bubeck, "Minimax policies for adversarial and stochastic bandits," in *Proceedings of the 22nd Annual Conference on Learning Theory (COLT)*, 2009.

——, "Regret bounds and minimax policies under partial monitoring," *The Journal of Machine Learning Research*, vol. 11, pp. 2785–2836, 2010.

J. Audibert, R. Munos, and C. Szepesvári, "Exploration-exploitation tradeoff using variance estimates in multi-armed bandits," *Theoretical Computer Science*, vol. 410, no. 19, pp. 1876 – 1902, 2009, (Algorithmic Learning Theory).

P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2003.

G. Bartók, D. Pál, and C. Szepesvári, "Toward a classification of finite partial-monitoring games," in *Lecture Notes in Artificial Intelligence (Algorithmic Learning Theory)*. Springer-Verlag, 2010, pp. 224–238.

D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Englewood Clifffs, New Jersey: Printice-Hall, 1989.

S. Bubeck, "Bandit games and clustering foundations," Ph.D. dissertation, Université Lille 1, 2010.

S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," 2012, submitted to Foundations and Trends in Machine Learning.

N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006.

S. C. Choi and V. A. Clark, "Sequential decision for a binomial parameter with delayed observations," *Biometrics*, pp. 411–420, 1970.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.

T. Desautels, A. Krause, and J. Burdick, "Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization," in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

J. Doob, *Stochastic processes*. Wiley, 1953.

M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," in *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, 2011.

A. Garivier and O. Cappé, “The KL-UCB algorithm for bounded stochastic bandits and beyond,” in *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*, 2011.

J. Gittins and D. Jones, “A dynamic allocation index for the sequential design of experiments,” in *Progress in Statistics (European Meeting of Statisticians, 1972)*, J. Gani, K. Sarkadi, and I. Vincze, Eds., North-Hollan, Amsterdam (The Netherlands), 1974, pp. 241–266.

S. Guha, K. Munagala, and M. Pal, “Multiarmed bandit problems with delayed feedback,” *CoRR*, vol. abs/1011.1161, 2010.

A. György, L. Kocsis, I. Szabó, and C. Szepesvári, “Continuous time associative bandit problems,” in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

W. Hoeffding, “Probability inequalities for sums of bounded random variables,” *Journal of the American Statistical Association*, vol. 58, no. 301, pp. pp. 13–30, 1963.

A. Jeffrey, *Handbook of Mathematical Formulas and Integrals*. Academic Press, 2004.

S. Kullback, *Information theory and statistics*. Dover Publications, 1997.

J. Langford, A. Smola, and M. Zinkevich, “Slow learners are fast,” *arXiv:0911.0491*, Nov. 2009.

C. Leopold, “Parallel and distributed computing: a survey of models, paradigms and approaches,” *Recherche*, vol. 67, p. 02, 2001.

L. Li, W. Chu, J. Langford, and R. E. Schapire, “A contextual-bandit approach to personalized news article recommendation,” in *Proceedings of the 19th international conference on World wide web*, ser. WWW ’10. New York, NY, USA: ACM, 2010, pp. 661–670.

O. A. Maillard, R. Munos, and G. Stoltz, “Finite-time analysis of multi-armed bandits problems with Kullback-Leibler divergences,” in *Proceedings of the 24th Annual Conference on Learning Theory (COLT)*, 2011.

S. Mannor and O. Shamir, “From bandits to experts: On the value of side-observations,” in *Advances in Neural Information Processing Systems 24*, 2011.

C. J. Mesterharm, “On-line learning with delayed label feedback,” in *Algorithmic Learning Theory*, ser. Lecture Notes in Computer Science, S. Jain, H. Simon, and E. Tomita, Eds. Springer Berlin / Heidelberg, 2005, vol. 3734, pp. 399–413.

——, “Improving on-line learning,” Ph.D. dissertation, Rutgers University, New Brunswick, NJ, 2007.

A. Nemirovsky and D. Yudin, *Problem complexity and method efficiency in optimization.* Chichester, New York: Wiley, 1983.

G. Neu, A. György, A. Antos, and C. Szepesvári, “Online markov decision processes under bandit feedback,” 2010, extended version of a paper submitted to NIPS-2010.

H. Robbins, “Some aspects of the sequential design of experiments,” *Bulletin of the American Mathematical Society*, vol. 58, no. 5, pp. 527–535, 1952.

A. Salomon and J.-Y. Audibert, “Deviations of stochastic bandit regret,” in *Lecture Notes in Computer Science (Algorithmic Learning Theory)*, vol. 6925. Springer-Verlag, 2011, pp. 159–173.

S. Shalev-Shwartz and Y. Singer, “A primal-dual perspective of online learning algorithms,” *Machine Learning Journal*, vol. 69, no. 2/3, pp. 115–142, 2007.

J. Shlens, *Notes on Kullback-Leibler Divergence and Likelihood*, 1st ed., Systems Neurobiology Laboratory, Salk Insitute for Biological Studies, La Jolla, CA 92037, 2007.

Y. Suzuki, “On sequential decision problems with delayed observations,” *Annals of the Institute of Statistical Mathematics*, vol. 18, no. 1, pp. 229–267, 1966.

E. Titchmarsh, *The theory of the Riemann zeta-function*. Oxford University Press, USA, 1987.

M. K. Warmuth, W. M. Koolen, and D. P. Helmbold, “Combining initial segments of lists,” in *Lecture Notes in Computer Science (Algorithmic Learning Theory)*, vol. 6925, 2011, pp. 219–233.

M. Weinberger and E. Ordentlich, “On delayed prediction of individual sequences,” *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 1959–1976, July 2002.

Appendix A

Some mathematical results

The following lemma is a deviation result due to Hoeffding (1963), which is used in the analysis of UCB-1 (Auer et al., 2002).

Lemma A.1. (Hoeffding's Inequality) *Let $Y_1, ..., Y_n$ be random variables with range $[0, 1]$ and $\mathbb{E}[Y_t|Y_{t-1}, .., Y_1] = \mu$. Let $\bar{Y}_s = \frac{1}{s}\sum_{i=1}^{s} Y_i$. Then for all $a \geq 0$:*

$$\mathbb{P}\left\{\bar{Y}_s \geq \mu + a\right\} \leq e^{-2a^2 s} \quad \textit{and} \quad \mathbb{P}\left\{\bar{Y}_s \leq \mu - a\right\} \leq e^{-2a^2 s}.$$

The next lemma is Jensen's inequality (Jeffrey, 2004, Page 29). We have stated it here for concave functions.

Lemma A.2. (Jensen's Inequality) *Let $f(x)$ be a concave function on an interval $I \subseteq \mathbb{R}$. Then for any integer n, for any non-negative real numbers $w_1, w_2, \ldots, w_n$ such that $\sum_{t=1}^{n} w_t = 1$, and for any $x_1, x_2, \ldots, x_n \in I$,*

$$\sum_{t=1}^{n} w_t f(x_t) \leq f\left(\sum_{t=1}^{n} w_t x_t\right).$$

The following lemma is a general result for concave functions, used in the proof of Theorem 3.1.

Lemma A.3. *Suppose that the function $f : [0, \infty) \mapsto [0, \infty)$ is differentiable and concave and that $f(0) = 0$. Then the function $g(s) = sf(1/s)$, defined over $(0, \infty)$ is non-decreasing in s.*

Proof. Since f is differentiable, by the definition of concavity we have $f(0) \leq f(t) + f'(t)(0 - t)$ for any $t \in (0, \infty)$. Together with $f(0) = 0$, this implies

$$\frac{1}{s} f'(1/s) \leq f(1/s) \tag{A.1}$$

for $s = 1/t$. Taking the derivative of $g(s)$, we get $g'(s) = f(1/s) - \frac{1}{s}f'(1/s) \geq 0$ by (A.1). Therefore, $g(s)$ is non-decreasing in s. □