Sequential Decision Making in Non-stationary Environments

Jia Yuan Yu

Doctor of Philosophy

Department of Electrical and Computer Engineering

McGill University

Montreal, Quebec

February 23, 2010

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Doctor of Philosophy

ACKNOWLEDGEMENTS

In retrospect, I have made an optimal decision even before starting work on the theory behind decision making. This decision was to embark on the journey to a Ph.D. As the journey comes to an end, comes the difficult task of giving credit to others. With the help a simple threshold rule—and apologies to those that I thank not in ink but only in spirit, I will begin by thanking my thesis supervisor Shie Mannor for welcoming me into "the family." I thank Peter Caines and Ioannis Psaromiligkos for their invaluable help and inspiration at many stages of my studies. I thank my co-authors Branislav Kveton, Nahum Shimkin, Georgios Theocharous, and John Tsitsiklis for their contributions to the work of this thesis. I thank my friends for the true blessing of their friendship. I thank my parents for their love and always being there for me.

CONTRIBUTIONS OF AUTHORS

This thesis contains work published in journals or proceedings. For the works of chapters 2, 3, 6, and 7, I have made the majority of the contribution. For the work of chapter 4, I contributed about one third of the work. For the work of chapter 5, I contributed about half of the work.

ABSTRACT

We consider the problem of sequential decision making in non-stationary environments. In order to avoid solutions that are too conservative, we capture the degree of non-stationarity in real-world problems through different mathematical models for the environment. In the first model, we add to the environment a state that follows Markovian dynamics subject to limited levels of non-stationary uncertainty. In the second model, we add non-stationary constraints to the environment. In the third model, we limit the frequency of non-stationary changes. In each of these models, we provide efficient learning algorithms and prove corresponding performance guarantees that depend critically on the degree of non-stationarity.

ABRÉGÉ

Nous étudions le problème de décisions séquentielles dans des environnements non-stationnaires. Pour éviter des solutions trop conservatrices, nous modelisons le degré de non-stationnarité à travers differents modèles mathématiques de l'environnement. Dans le premier modèle, nous ajoutons à l'environnement un état qui suit une dynamique Markovienne, sujet à des niveaux limités d'incertitude non-stationnaire. Dans le second modèle, nous ajoutons des contraintes non-stationnaires à l'environnement. Dans le troisième modèle, nous limitons la fréquence des changements non-stationnaires. Pour chaque modèle, nous présentons des algorithmes d'apprentissage efficients et prouvons des guaranties de performance qui dépendent du degré de non-stationnarité.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

This thesis deals with the theory of learning for machines. The goal is to develop algorithms for making good decisions on the basis of past observations. The models that we study are generalized versions of sequential decision-making problems, the simplest of which consists of predicting the next element in a sequence that is generated by an unknown source. The classical solution of statistical learning assumes that this source is stationary and uses past observations to predict all the subsequent elements of the sequence. This approach works as long as the source is stationary, but many problem in practice are not stationary, e.g., stock markets, and power distribution and data networks. To address this shortfall, so-called *online learning* methods have been developed in order to deal with these non-stationary (or unpredictable) settings. These methods, however, have short-comings of their own. Due to their simplistic representation of the non-stationary source as an arbitrary individual sequence, they produce solutions that are excessively conservative or prohibitively inefficient. This thesis introduces a set of flexible models that capture the notion of varying degrees of non-stationarity, and hence allow us to design better suited solutions.

The motivation for online learning methods comes not only from the field of machine learning. A wide range of problems can be modelled as sequential decision making in non-stationary environments. These problems come from different parts of science and engineering. One example is power management in wireless networks [100], where the objective is to trade-off energy

consumption and performance when service requests are unpredictable. Another example is routing packets of data in computer networks with unknown traffic and demand [13, 14]. In information theory, the important problem of compressing individual sequences [134] is also an instance of online learning. In computer science, examples include the problem of caching data to minimize memory access in the worst-case scenario [25, 61], and the construction of data structures to minimize the worst-case access cost [26]. Online learning methods can also be combined with other methods of machine learning in order to select important features of high-dimensional data points and querying for labelled training data points (active learning) [34].

The methods of online learning also have important applications in other fields. Indeed, some of the early theoretical developments occurred in the design of adaptive strategies for repeated games in game theory [64] and the sequential design of experiments in statistics [85, 18]. These methods also have a strong connection with strategies for playing adaptive games [55, 56, 52, 65]. Furthermore, these methods have led to applications in the design of clinical trials in medicine [16], the pricing of commodities in economics [114], the sequential management of financial portfolios [38, 68], and the exploration for mineral and petroleum resources [17]. Recent applications include the design of online auction mechanisms [27, 76], the prediction of diseases [132], the detection of malicious Internet content [89], and the development of competitive algorithms for playing complex board games (*e.g.*, game of go [125]).

This thesis is written from a machine learning perspective, but the same general problem appears in the fields of control theory, game theory, operations research, and economics. The methods and results are therefore presented through abstract models, instead of specific applications, so that they may easily be employed in the wide variety of applications just mentioned.

The two common threads throughout the thesis are problems of sequential decision and non-stationary settings. In Sections 1.1 and 1.2, we explain the notion of non-stationary settings and the goal of sequential decision problems in such settings. In Section 1.3, we illustrate these ideas with examples of practical problems.

1.1 Non-stationary Environments

In statistical learning problems, the rewards to the decision maker, or *agent*, form a stationary random process. In this thesis, we consider *non-stationary* environments, *i.e.*, where the sequence of observations from the environment is non-stationary. This thesis considers a number of different models for the environment—Markov decision processes and stochastic games, sequential constrained optimization, and multi-armed bandits, each of which has a notion of non-stationarity that is specific to the environment. In this section, we present the common concepts and ideas, but we leave the specific formalism to the respective chapters.

Statistical learning theory has brought significant improvements in important applications such as routing, classification, and speech recognition. For these applications, the success of existing techniques, such as dynamic programming [15], boosting [53], support vector machines [123] and LASSO regularization [122], relies on the fundamental assumption that the observations are generated from a *stationary* source, which is fixed and does not change over time. This simplifying assumption, however, does not accurately reflect sources that change in an unpredictable fashion at unpredictable time instants. In this thesis, our goal is to represent these sources with various models of non-stationary environments.

In general, a non-stationary environment can be modelled as a source that generates a deterministic sequence of observations. The elements of this

sequence take arbitrary values from a given set and these values are a priori unknown to the agent. We call these sequences *arbitrary* or *individual* sequences, as in [134].

Non-stationary environments can also be represented from the game theoretic perspective. In this case, the agent interacts (as in a repeated game) with a *non-rational* or *arbitrary* opponent that may not have a well-defined preference, *i.e.*, the sequence of actions of this opponent is an arbitrary sequence. This opponent may represent an aggregate of other agents or Nature. This useful view gives us access to tools from game theory and will be adopted in Chapters 3 and 4.

A major advantage of modelling non-stationary environments with arbitrary sequences is simplicity: there is no need to make assumptions on the source of observations and estimate its parameters. A possible pitfall, however, is that the model may be too general, the assumptions too weak. By making no assumption, we must consider the worst case in our solution, and hence, the solution may be too conservative. This thesis addresses this pitfall by introducing models of non-stationary environments where different degrees of non-stationarity can be distinguished, and hence more suitable algorithms can be designed.

We can capture the notion of *degrees of non-stationarity* in a number of different ways. For example, we can limit the degree of non-stationarity by bounding the frequency of unpredictable events, restricting the magnitude of unpredictable changes, or making one component of the source stationary. We can also increase the degree of non-stationary by extending the source with additional components such as non-stationary constraint sequences, and non-stationary state-transition dynamics. Specifically, in this thesis, we distinguish degrees of non-stationarity in the following models:

- Markov decision processes where the state-transition dynamic is stationary but the rewards are non-stationary and may possibly adapt to the actions of the agent (Chapter 2);
- Markov decision processes where both rewards and state-transitions are non-stationary, but where the magnitude of changes is bounded (Chapter 3), which may also be interpreted as stochastic games against an arbitrary opponent;
- Repeated games with side constraints against an arbitrary opponent (cf. Chapter 4), which can also be interpreted as sequential optimization problems where both rewards and constraints are non-stationary (Chapter 5);
- Multi-armed bandit problems where the rewards are non-stationary, but the frequency of changes is bounded (Chapters 6).

We present motivations for these problems in Section 1.3. In the next section, we define our objective in these problems.

1.2 Objective

The goal in non-stationary sequential decision problems differs in a significant way from the goal of learning problems in stationary environments. For instance, since the reward process is stationary in the case of reinforcement learning, the challenge arises from building a representative model and estimating the unknown parameters of the model. Moreover, the agent may hope to perform (asymptotically) as well as *every* alternative policy. In contrast, for the case of online learning, the challenge is to deal simultaneously with all possible instances of a non-stationary environment. Consequently, we seek a policy that performs well against every individual sequence of reward functions. A solution of such generality can only be fairly evaluated against a *restricted* set of alternatives—the set of stationary (time-invariant) policies,

for instance. We also refer to sequential decision making in a non-stationary environment as *online learning*, and refer to the decision maker as the *agent*.

1.2.1 Notion of Regret

Throughout this thesis, we compare the actual performance of the agent with the performance of the best alternative policy in retrospect. The difference between these two competing quantities is the *regret*. For example, the agent's actual performance is the reward accumulated by following a learning algorithm. The baseline is the maximum reward that can be accumulated by a policy from a limited set of alternative policies and it is evaluated in retrospect against the same instance of the environment as observed by the agent. Intuitively, these two competing quantities incarnate the notions of *adaptability* and *foresight*. Our results show that it is asymptotically possible to trade-off one for the other.

Although the objective is slightly different in each chapter of the thesis, the general goal is the same and the performance is evaluated in the same fashion. We illustrate this with a simple sequential decision problem. A decision maker has a set of action choices A at each time instant $t = 1, 2, \ldots$. There is an arbitrary sequence of bounded reward functions $r_t : A \to \mathbb{R}$ for $t = 1, 2, \ldots$ that are initially unknown to the agent. At each time step t, the decision maker takes an action a_t, and then observes the reward function r_t and receives a reward $r_t(a_t)$. An algorithm in this problem is a rule that takes as input the history of observations available to the decision maker and outputs an action. We will also consider randomized algorithms, which take also as input a random variable. Suppose that the decision maker follows an algorithm that dictates the sequence of actions $a_1, a_2, \ldots$. The average regret

of this algorithm is

$$\max_{a \in A} \frac{1}{T} \sum_{t=1}^{T} r_t(a) - \frac{1}{T} \sum_{t=1}^{T} r_t(a_t). \tag{1.2.1}$$

The agent's accumulated reward is averaged over the time steps $1, 2, \ldots, T$ and compared to a baseline. We take as baseline the maximum average reward accumulated by alternative policies that always picks a single action $a \in A$. The actual sequence of reward functions $r_1, r_2, \ldots$ observed by the agent is also used to evaluate the baseline in retrospect.

Our main goal will be to describe the long-term behaviour of the regret. We say that a policy *minimizes regret*, or is *no-regret*, if the average regret vanishes almost surely as $T \to \infty$ for every possible sequence of reward functions $r_1, r_2, \ldots$. Such algorithms are also said to be *Hannan consistent* [64] and *universally consistent* [56]. The above regret notion arises naturally from game theoretic considerations because a natural equilibrium is achieved if each player in a repeated game plays a regret-minimizing strategy [34].

Just as we can apply regret minimization to different settings, different notions of regret are suitable to different models: *e.g.*, with different classes of comparison, finite time-horizon or discounted rewards. For instance, when the environment has a state and the accumulated reward depends on the agent's action-history as in Chapter 2, it is useful to consider the steady-state regret. By introducing side-constraints to the online learning model (Chapter 4), it is necessary to relax the baseline performance to its convex hull. By restricting the frequency of non-stationary events in multi-armed bandit problem (Chapters 6 and 7), it is more meaningful to extend the set of comparison policies. In each chapter, we adapt the classical notion of regret of (1.2.1) to the respective setting.

1.2.2 Uniform and Asymptotic Guarantees

When dealing with non-stationary components of the environment—for which no model exists, we must account for the worst case. Therefore, the objective is to guarantee a certain performance *uniformly* against *all* possible instances of the non-stationary environment. In this thesis, we seek solutions that adapt to the non-stationary environments and are *robust* to the degrees of non-stationarity outlined in Section 1.1. Hence, the results that we present hold uniformly over all the individual sequences of observations from the environment. From the game theoretic perspective, we seek strategies with performance guarantees that hold regardless of the opponent's motivation or behaviour.

1.3 Examples

By virtue of robustness to non-stationary environment changes, the online learning approach is suitable for real-world problems. More examples relevant to the specific model of each chapter will be presented in the respective chapter. In this section, we present intuitive examples that do not require delving into the details of mathematical models.

The following example is a Markov decision process with an arbitrary sequence of reward functions.

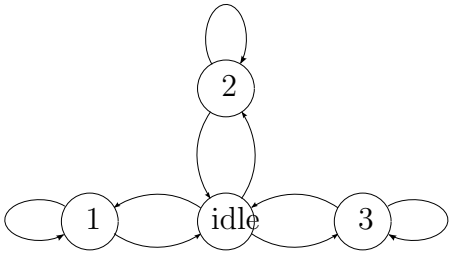


Figure 1–1: Example 1.1: Multi-armed bandit with three arms and an idle state.

Example 1.1 (Multi-armed Bandit with Restrictions)**.** Consider a non-stochastic multi-armed bandit problem (*e.g.*, [10]) with k arms and a special restriction

(see Figure 1–1). The sequence of reward functions is arbitrary. When choosing an arm, the agent may repeat the previous arm or switch to a different arm. If it switches to a different arm, it must go through an additional intermediate state before completing the switch. We have one state for each of the k arms, each with two actions ("repeat" and "switch"), and one "idle" state with k actions leading to the corresponding arms. An instance of such a model is an electrical power distribution network where generators cannot turn on or off instantaneously due to inertia, but instead must ramp up and down gradually [36]. In this case, the arbitrary reward can model the revenue to the operator, which depends on complex and hard-to-predict factors such as power outages, the quantity of power supplied by competing operators, and a non-stationary demand. In Chapter 2, we will show that, under reasonable assumptions, it is possible to achieve asymptotically a reward as high as that of the best arm, regardless of extra delays and transition costs.

A Markov decision process model may also add other features to the classical multi-armed bandit problem, such as costs to transition between arms, forbidden transitions between some arms. Since the MDP model contains multiple states and a reward process that depends on states and actions, it can capture real dynamics of the investment process, such as delays in completing transactions and commissions. Therefore, our results on MDPs with arbitrary reward processes (Chapter 2) provide a solution to an open problem of sequential investment [38].

Stochastic games [116] (*e.g.*, [108]) arise as natural models for problems where an agent interacts with an opponent in a Markovian environment, where the state evolution depends on the actions of both the agent and the opponent. The reward to the agent likewise depends on both players' actions. In the following example, the opponent is not rational and plays an arbitrary

sequence of actions. In this case, an online learning approach can be applied to exploit the opponent that acts in a sub-optimal fashion.

Example 1.2 (Queuing System)**.** As an example of a stochastic game, let us consider a queueing system. At each time instant, the agent sends a job to be serviced at one of several queues. A job incurs a delay in each queue proportional to its congestion. Besides our agent, there are other agents sending jobs to the same queues. If the other agents do not behave in a cohesive manner, they can be aggregated into an arbitrary opponent. The congestion pattern on all queues may be represented as a state that evolves according to all the agents' action and the movement of jobs through the queueing system. Similar problems are encountered in networks, *e.g.*, admission control, routing, and transmission rate control [6]. An additional application to communication over arbitrarily varying channels is described in [86].

1.4 Literature Overview

In this section, we survey works in the literature on sequential decision-making in non-stationary environments. Besides the distinction between stationary and non-stationary environments, in this section, we further divide online learning problems into two types: expert problems—in which the entire reward function is revealed after each action of the agent—and bandit problems, in which only the reward corresponding to the agent's action is revealed. Since each of the following chapters considers a different model and employs different methods, each chapter will contain a more specific literature survey, where we put the results of that chapter in perspective with other works that share similarities.

1.4.1 Non-stationary Settings

The basic version of the online learning problem, described in Section 1.2, has been a topic of great interest for over five decades and across various fields

of research. Hannan presents the first regret-minimizing algorithm based on perturbed fictitious play in his seminal paper [64]. He proposes a randomized policy whose regret is of the order of $O(|A|\sqrt{T})$ after T plays, where $|A|$ is the number of actions available to the agent. The exponentially weighted average forecaster algorithm, introduced in [87] and [124], reduces the regret to the order of $O(\sqrt{T \log |A|})$. A number of other no-regret algorithms have been introduced, using methods such as Blackwell's approachability-based scheme [23], smooth fictitious play [56], calibrated forecasting [50], multiplicative weights [54], and online gradient ascent [133].

The basic online learning problem has an equivalent *repeated game* formulation, which assumes a reward function of the form $r_t(a) = R(a, b_t)$, where b_t is the action chosen by an opponent, and R is a known payoff function. In this case, observing the opponent's action b_t is equivalent to observing the reward vector r_t. Online learning in the two-player repeated game framework, *without any state dynamics*, has been extensively studied in [64, 55, 56, 52, 65].

The online learning problem is also equivalent to the universal coding problem in information theory, where the notion of regret is known as *redundancy*. Algorithms for universal[1] prediction [96] and universal coding [111] are regret-minimizing algorithms. For the problem of universal data compression, where the source is unknown a priori, [134] presents a regret-minimizing algorithm that encodes every individual sequence of this source as well as all finite-state encoders in the long run. Similarly to our objective (cf. Section 1.2.2), this result holds even as the number of finite-state machines in the comparison class is allowed to increase as the length of the sequence increases.

[1] This concept of universality refers to the fact that the regret bounds hold uniformly over all sequences generated by a source.

Depending on the form of the agent's reward function and the extent of the comparison class within some hierarchy, different bounds on regret are obtained [96]. Regret minimization also appears in the frameworks of sequential gambling [49] and sequential investment [38].

There are two important variants of the basic online learning problem: *expert*-type and *bandit*-type problems. Expert-type problems, such as the formulation of Section 1.2, where the entire reward function is observed, are related to the problem of prediction with expert advice [87]. In bandit-type problems, only the component $r_t(a_t)$ of the reward function, corresponding to the taken action a_t, is observed at each time step. This corresponds to the *nonstochastic* or *adversarial* multi-armed bandit problem [10], Another model of partial observation is studied in [35].

Another important distinction exists between an *oblivious* opponent (or environment) and a *non-oblivious* or adaptive one [34]. In the former case, the reward vector sequence is assumed fixed in advance but unknown. In the latter case, it may depend on the actions previously taken by the agent. This distinction is crucial to [41] and Chapter 2.

An exact but computationally prohibitive solution to the online learning problem exists when the time horizon T of the problem is known in advance. This solution is based on the dynamic programming approach [15, 19], but quickly becomes intractable as T increases. In this thesis, we seek efficient solutions with asymptotic bounds on the regret that do not require prior knowledge of the time horizon T. The notion of discounted regret is treated in [34].

1.4.2 Stationary Settings

The problem of regret minimization is also studied in the case of stationary environments. In this case, the regret is defined similarly to Section 1.2,

except that the baseline is actually the optimal expected reward, instead of the reward in retrospect of the best alternative policy. The classical instance of this problem is the stochastic multi-armed bandit, which is a fundamental problem to statistics [112] and exemplifies the trade-off between exploitation and exploration in reinforcement learning [121]. In the basic bandit problem, there is a finite set of arms, where each arm generates an independent sequence of random rewards according to an unknown probability distribution. The objective is to pull the arms sequentially so as to maximize the total reward. Lai and Robbins showed that the regret is at least of the order of $\Omega(\log T)$ and provided a policy based on upper-confidence indices that achieves the lower bound exactly in the limit [85]. This solution is an instance of the general method of Gittins indices [59]. The UCB algorithm, with a uniform finite-time regret bound of $O(\log T)$, is presented in [9].

1.5 Contributions

This thesis deals with different degrees of non-stationarity in three settings of sequential decision-making:

- Markov decision processes and stochastic games,
- sequential constrained optimization problems and constrained repeated games,
- and multi-armed bandits.

In each of these settings, we present new algorithms that adapt to non-stationary changes, along with performance guarantees in the form of finite-time expectation bounds or probabilistic bounds on the performance loss. Besides these theoretical properties, our algorithms are also computationally efficient. We validate their performance through experiments in Chapters 4, 5, and 6.

In Chapter 2[2] , only one of the two components of the environment is non-stationary. The decision-maker interacts with a version of Markov decision processes, where the state transitions occur according to stationary dynamics (as in standard Markov processes), but where the reward function may change in a non-stationary fashion over time. We present a method to perform as well—in retrospect—as every stationary policy against every possible sequence of reward functions. Our approach uses an efficient online algorithm—in the spirit of reinforcement learning—that ensures that the average performance loss vanishes over time, provided that the environment is oblivious to the decision-maker's actions. This generalizes the classical no-regret result for repeated games. Moreover, it is possible to modify the basic algorithm to cope with instances where the decision-maker has only limited observations of the rewards. We also present approximation techniques to reduce the computational cost at the expense of performance, and we extend our baseline policies to include a subset of non-stationary policies.

In Chapter 3[3] , we generalize the MDP model of Chapter 2 to a much harder model, where *both* the rewards and the state-transitions may change in a non-stationary fashion. This setting is a generalization of ordinary online learning problems and corresponds to a one-sided stochastic game against a non-rational opponent. In order to obtain tractable solutions, we restrain

[2] This work is published in the journal *Mathematics of Operations Research* [131].

[3] Parts of this work are published in the *Proceedings of the International Conference on Game Theory for Networks (GameNets)* [129] and the *Proceedings IEEE Conference on Decision and Control* [128].

the degree of non-stationarity by the magnitude of the changes in the state-transition probabilities. As in Chapter 2, we propose online learning algorithms and provide guarantees on their performance evaluated in retrospect against alternative policies. We first use an approach based on robust dynamic programming, and then present a computationally efficient simulation-based algorithm (in the style of Q-learning) that requires neither prior knowledge nor estimation of the transition probabilities. The case of limited observation of the rewards can also be handled with a modification. Unlike existing results though, our guarantees depend critically on the magnitude of non-stationary changes in the transition probabilities.

In Chapter 4[4] , we consider a sequential constrained optimization problem, where the objective is to maximize the average reward subject to constraints on the sample path. The challenge is that both the sequence of objective functions and the sequence of constraints change in a non-stationary fashion. Unlike the case where the sequence of constraints is stationary, it is not possible to attain the average reward of the best stationary solution. The degree of non-stationarity in this setting is such that it is necessary to relax the baseline of our notion of regret. However, we identify the set of attainable average rewards and present an algorithm using calibrated forecasting that attains this set. We evaluate the performance of computationally efficient methods based on non-calibrated forecasters in experiments.

[4] This work is published in the *Journal of Machine Learning Research* [94].

In Chapter 5[5] , we model a real-life power management problem as a sequential optimization problem with non-stationary rewards and constraints. The model is similar to that of Chapter 4, but the constraints and objective are defined differently. For this setting, we provide a solution using prediction with expert advice. In addition to theoretical guarantees, we validate its performance through experiments.

In Chapter 6[6] , we consider a random source that is stationary over long intervals, but may change distribution abruptly at unknown time instants. We quantify the degree of non-stationarity by the frequency of changes from one stationary distribution to another. This is the piecewise-stationary generalization of the classical multi-armed bandit problem. We evaluate the regret with respect to the sequence of instantaneously optimal arms—which is more competitive than the notion of regret for typical non-stationary settings, but assume that side observations are revealed to the agent. We present an efficient solution that detects shifts in the mean of the reward function over different windows. We show that the expected regret of this algorithm has upper and lower bounds that match up to a constant factor.

In Chapter 7, we consider a different version of piecewise-stationary multi-armed bandits than Chapter 6, where the set of arms is infinite—*i.e.*, an interval of the real line, but the arm rewards are related in an unimodal fashion. In this setting, we provide an efficient algorithm that iteratively eliminates subsets of arms based on samples of a small number of arms. We show that

[5] Parts of this work are published in the *Proceedings of the AAAI Conference on Artificial Intelligence* [81] and the *Proceedings of the International Symposium on Artificial Intelligence and Mathematics* [80].

[6] This work is published in the *Proceedings of the International Conference of Machine Learning (ICML)* [130].

this algorithm minimizes the regret without requiring any side observation in contrast to Chapter 6. Moreover, in the case where there are no change-points, the regret bound of our algorithm matches existing lower bounds for similar continuum-armed bandit problems.

Throughout this thesis, for simplicity of notation, we sometimes reuse the same symbol (*e.g.*, ϵ, δ, k, n) to denote different quantities across different chapters. For instance, r_t represents a deterministic reward function in one chapter; but in another chapter, r_t denotes a stochastic reward vector. Within each chapter, however, each symbol consistently refers to the same notion.

CHAPTER 2
Markov Decision Processes with Arbitrary Reward Processes

2.1 Introduction

In this chapter, we consider a learning problem where the decision maker interacts with a standard Markov decision process, with the exception that the reward functions vary arbitrarily over time. We show that, against every possible realization of the reward process, the agent can perform as well—in hindsight—as every stationary policy. This generalizes the classical no-regret result for repeated games. Specifically, we present an efficient online algorithm—in the spirit of reinforcement learning—that ensures that the agent's average performance loss vanishes over time, provided that the environment is oblivious to the agent's actions. Moreover, it is possible to modify the basic algorithm to cope with instances where reward observations are limited to the agent's trajectory. We present further modifications that reduce the computational cost by using function approximation and that track the optimal policy through infrequent changes.

A common theme in the majority of existing works is that the decision maker faces an *identical* decision problem at each stage. This falls short of addressing realistic decision problems that often take place in a dynamic and changing environment. Such an environment is commonly captured by a state variable, which evolves as a controlled Markov chain. The model thus obtained is that of a Markov decision process (MDP), augmented by arbitrarily varying rewards and (possibly) transitions. Furthermore, by modelling the arbitrary elements as the actions of an opponent (actual or virtual), the model takes the form of a two-person *stochastic game* [116], played between the decision

maker and an arbitrary opponent. In this chapter, we consider MDPs where only rewards change arbitrarily. Such a model arises as a simple extension to a standard online decision problem, as illustrated by the examples of Section 1.3.

2.1.1 Related Works

Various versions of MDPs have been studied in a large number of works. A well-known setting is that of reinforcement learning, where the rewards are i.i.d. random variables generated by a fixed, but unknown, source. For example, [48] considers MDPs where the state and action spaces and the transition function are *fixed* but not known to the decision maker.

Regret minimization in such dynamic environments has been the topic of only a handful of papers so far. This may seem surprising, given the proliferation of interest in no-regret algorithms, on the one hand, and the extensive literature on MDPs and stochastic games, on the other hand. In [91], the problem has been considered within the general stochastic game model, where both the transition probabilities and the rewards are affected by the actions of both players, the opponent is adaptive and the opponent's actions are observed at traversed states only. A central observation of that paper is that no-regret strategies do not exist for the general model (where regret is defined relative to the best *stationary* policy of the decision maker). An exception is the case where the transition probabilities are controlled by the opponent only, which can be treated by applying a no-regret algorithm at each state separately and independently of other states. For the general model, a relaxed goal was set and shown to be attainable by using approachability arguments. We note that similar conclusions hold true for the (essentially simpler) model of repeated games with varying stage durations, as reported in [93]. Merhav *et al.* [97] have considered sequential decision problems where the loss functions

have memory, which correspond to special MDPs, where every state is reachable from every other via a single action. They presented an algorithm using piecewise-constant policies and provided regret-minimizing guarantees similar to ours.

The paper [44], whose model is closest to the present one, focuses on MDPs with arbitrarily varying rewards. Specifically, it assumes that (1) The state dynamics is known, namely, the state transition probabilities are determined by the decision maker alone; (2) Oblivious opponent: The reward functions, while unknown to the decision maker, are fixed in advance; (3) Observed reward functions: The entire reward function r_t (for every state and action) is observed after each stage t. As mentioned in [44], a simple-minded approach to the problem could start by associating each deterministic stationary policy with a separate expert, and applying existing experts algorithms in that setting. However, as the number of such policies is prohibitive for all but the smallest problems, this approach is computationally infeasible and slow to converge. Thus, more efficient algorithms must be devised. Under the above assumptions, Even-Dar *et al.* propose an elegant no-regret algorithm, and provide finite-time bounds on the expected regret. The suggested algorithm places an independent experts algorithm at each state; however, the feedback to each algorithm depends on the aggregate policy determined by the action choices of all the individual algorithms and by the value function which is computed for the aggregate policy.

Our chapter also relates to problems outside the regret minimizing framework. Optimal control in MDPs with unknown, but stationary, reward processes can be solved using reinforcement learning, *e.g.*, model-based and Q-learning algorithms [126]. In contrast to an ordinary stochastic game, the

opponent in our model is not necessarily rational or self-optimizing. Our emphasis is providing the agent with policies that perform well against *every* possible opponent. A max-min solution to a zero-sum stochastic game, such as one produced by the R-max algorithm of [30], may well be too conservative when the opponent is not adversarial. It may be in the agent's interest to exploit the non-adversarial characteristic of the opponent. Our model corresponds to a stochastic game where an arbitrary opponent picks the reward functions, but does not affect state transitions.

The basic model that we consider here is similar to [44]. We start by examining the above-mentioned assumptions, and show that the oblivious opponent requirement is necessary for the existence of no-regret algorithms. This stands is sharp contrast to the standard (state-less) problem of prediction with expert advice, where no-regret is achievable even against an adaptive opponent. We then propose for this model a new no-regret algorithm in the style of [64], which we call the Lazy follow-the-perturbed-leader (FPL) algorithm. This algorithm periodically computes a single stationary policy, as the optimal policy against a properly perturbed version of the empirically observed reward functions, and applies the computed policy over a long enough time interval. We provide a modification to this algorithm (the Q-FPL algorithm) that avoids the exact computation of optimal policies by incorporating incremental improvement steps in the style of Q-learning [20]. Next, we extend our results to the model where only on-trajectory rewards are observed; namely, only the rewards along the actually traversed state-action pairs. Clearly, this is a more natural assumption in many cases, and may be viewed as a generalization of the bandits problem to the dynamic setting. Finally, we introduce a variant of our basic algorithm that minimizes regret with respect to non-stationary policies with infrequent changes, in the spirit of [70].

Our emphasis in this chapter is on asymptotic analysis and almost-sure convergence; namely, we show that the long-term average regret vanishes with probability one. Explicit finite-time bounds on the expected regret are provided as intermediate results or as part of the proofs. To summarize, the main contributions of this chapter are the following:

- Establishing the necessity of the oblivious opponent assumption in this model.
- A novel no-regret algorithm for MDPs with arbitrarily varying rewards. In contrast to [44], this algorithm is Hannan consistent and has diminishing average computational effort over time.
- The first reported no-regret algorithm for the MDP model when only on-trajectory rewards are observed.
- The incorporation of Q-learning style incremental updates that alleviate the computational load and spread out the load over time. Moreover, the Q-learning style updates eliminate the requirement of knowing the state transition probabilities.

The rest of the chapter is organized as follows. We describe the model in Section 2.2, and motivate our obliviousness and ergodicity assumptions in Section 2.3. Section 2.4 describes and analyzes our main algorithm. The Q-FPL variant and related approximation results are described in Section 2.5. The extension to the case of on-trajectory reward observations is described in Section 2.6. In Section 2.7, we consider regret minimization with respect to a subset of non-stationary policies: the policies with a limited number of changes from one step to another.

2.2 Problem Definition

We consider an agent facing a dynamic environment that evolves as a controlled Markov process with an arbitrarily varying reward process. The

reward process can be thought of as driven by an abstract *opponent*, which may stand for the collective effect of other agents, or the moves of Nature. The controlled state component is a standard *Markov decision process* (MDP) that is defined by a triple (S, A, P), where S is the finite set of states, A is the finite set of actions available to the agent, and P is the transition probability—that is, $P(s' \mid s, a)$ is the probability that the next state is s' if the current state is s and the action a is taken.

The discrete steps are indexed by $t = 0, 1, \ldots$. We assume throughout the chapter that the initial state at step 0 is fixed and denoted s_0. At the t-th step, the following happen:

1. The opponent chooses a reward function $r_t : S \times A \to [0, 1]$;
2. The state s_t is revealed;
3. The agent chooses an action a_t;
4. The entire reward function $r_t = \{r_t(s, a)\}_{(s,a)\in S\times A}$ is revealed; the agent receives reward $r_t(s_t, a_t)$;
5. The next state $\mathbf{s}_{t+1}$ is determined stochastically according to the transition function P.

Remark 1 (Notation). When confusion is possible, we will write the random variables with a bold typeface (*e.g.*, $\mathbf{s}_t$) to distinguish them from their realizations written with a normal typeface (*e.g.*, s_t).

In general, the opponent determines a sequence of reward functions $r_0, r_1, \ldots$, where r_t may be picked on the basis of the past state-action history

$$(s_0, a_0, \ldots, s_{t-1}, a_{t-1}).$$

In most of the following development, we consider oblivious opponents that pick the reward functions $r_1, r_2, \ldots$ independently of the past state-action history. This assumption is made exact in the following section.

We are interested in policies that respond to the observed sequence of rewards. When choosing action a_t at step t, we assume that the agent knows the current state s_t, as well as the past state-action history and the past reward functions. Hence, we define a *policy* as a mapping from the reward history $(r_0, \ldots, r_{t-1})$ and state-action history $(s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t)$ to an action in the simplex $\Delta(A)$[1] . A *stationary* policy is a function $\mu : S \to \Delta(A)$ that depends solely on the current state s_t—and not on the history of the rewards or states. We denote by Σ the set of stationary policies. A *deterministic stationary* policy is a mapping $\mu : S \to A$ from the current state to an action. We first present in Section 2.4 a policy for the agent that assumes that the transition probability function P is known. However, this requirement is not crucial and we shall dispense with it via simulation-based methods in Section 2.5.

Let us consider a sequence of state-action pairs $(\mathbf{s}_t, \mathbf{a}_t)_{t=0,1,\ldots}$ induced by following a stationary policy μ and starting from the initial state s_0. Let $d_t(\mu; s_0)$ denote the probability distribution of $(\mathbf{s}_t, \mathbf{a}_t)$. With respect to the stationary policy μ, if it admits a unique stationary state-action distribution, we denote the latter by $\pi(\mu)$. Given an arbitrary reward function $r : S \times A \to [0, 1]$, we introduce the following inner product notations to denote the expected reward at time step t starting from state s_0 and following policy μ, and the expected reward according to the stationary distribution associated

[1] $\Delta(A)$ denotes the set of all probability vectors over A.

with policy μ:

$$\langle r, d_t(\mu; s_0)\rangle \triangleq \sum_{(s,a)\in S\times A} r(s,a) P_\mu\big((\mathbf{s}_t, \mathbf{a}_t) = (s,a) \mid s_0\big), \tag{2.2.1}$$

$$\langle r, \pi(\mu)\rangle \triangleq \sum_{(s,a)\in S\times A} r(s,a)\pi(\mu)(s,a).$$

2.2.1 Assumptions

Our main results require the following assumptions. Their necessity will become clear from the counterexamples of Section 2.3. We begin with the following ergodicity assumption.

Assumption 2.1 (Uniform Ergodicity)**.** The induced Markov chain is uniformly ergodic over the set of stationary policies. This guarantees that there exists a unique stationary distribution $\pi(\mu)$ for each policy μ. Moreover, there exists (cf. [28]) a uniform mixing time $\gamma \geq 0$; *i.e.*, there exists a finite $\gamma \geq 0$ such that for every stationary policy $\mu \in \Sigma$, every initial state s_0, and $t \geq 0$, we have

$$\|d_t(\mu; s_0) - \pi(\mu)\|_1 \leq 2e^{1-t/\gamma}.$$

Remark 2. The ergodic assumption is quite weak as it only requires that all recurrent states in the Markov chain communicate and that the chain is aperiodic. However, there may exist transient states—which may depend on the stationary policy employed.

The main results of this chapter hold when the opponent is oblivious; in other words, the sequence of reward functions does not depend on the state-action history. There are two justifications for this approach. First, from a modelling perspective, the agent may interact with other agents that are truly oblivious, irrational, or have an unspecified or varying objective. This renders their behaviour "unpredictable" and seemingly arbitrary. Second, in

the presence of many agents, a single agent has little effect on the overall outcome (*e.g.*, price of commodities, traffic in networks) due to the effect of large numbers [12]. Moreover, as Example 2.1 shows, the regret can not be made asymptotically small when the opponent is not oblivious. Formally, we state the obliviousness assumption as follows.

Assumption 2.2 (Oblivious Opponent)**.** The reward functions $r_0, r_1, \ldots$ are deterministic and fixed in advance.

Remark 3. Alternatively, we may assume that the reward functions $r_0, r_1, \ldots$ are random variables on the null σ-algebra. Hence, for every random variable $\mathbf{X}_t$ measurable by the σ-algebra generated by $(\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_t, \mathbf{a}_t)$ satisfies the following:

$$\mathbb{E}\big[r_t(s,a)\mathbf{X}_t\big] = r_t(s,a)\mathbb{E}\big[\mathbf{X}_t\big], \quad \text{for all } (s,a) \in S \times A. \tag{2.2.2}$$

The following results can be shown to apply even when the reward functions are randomly chosen at each step, independently of the state-action history, so that (2.2.2) holds. This case can be handled similarly to the deterministic one, at the expense of somewhat more cumbersome notation that we avoid here.

2.2.2 Regret

In general, the goal of the agent is to maximize its cumulative reward $\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t)$ over a long time horizon of T steps, where T need not be specified a priori. We shall focus on policies that minimize the *regret*, which measures how worse off the agent is compared to the best stationary policy in retrospect. This regret arises from the lack of prior knowledge on the sequence of reward functions picked by the opponent. We present three related notions of regret that differ in how the sequence of reward functions is retained, and in the choice of initial state. All three definitions of regret for our model

collapse to the classical notion of regret for repeated games (cf. [34]). Our basic definition for regret is the following.

Definition 2.1 (Worst-case Regret)**.** The *worst-case* average regret, with respect to the realization $r_0, \ldots, r_{T-1}$ of the reward process, is

$$L_T^W \triangleq \sup_{\mu \in \Sigma} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_t(\tilde{\mathbf{s}}_t, \tilde{\mathbf{a}}_t)\right] - \frac{1}{T}\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t), \tag{2.2.3}$$

where $\mathbb{E}$ denotes expectation over the sequence $(\tilde{\mathbf{s}}_t, \tilde{\mathbf{a}}_t)$ induced by the stationary policy μ. It is implicitly understood that both sequences $\tilde{\mathbf{s}}_t$ and $\mathbf{s}_t$ start at the initial state s_0 and follow the transition kernel P. This regret is a *random* quantity since the trajectory $(\mathbf{s}_t, \mathbf{a}_t)$ is random.

The above definition of regret is one possible extension of the concept of regret introduced in [64]. However, it is not the only natural definition of regret and we shall provide two additional notions of regret. An alternative to defining the regret with respect to stationary policies is to take as basis for comparison an agent that possesses only prior knowledge of the *empirical frequency* of reward functions. In this case, it is natural to consider the MDP where the states, actions, and transition probabilities are as before, but where the reward function at every step t is

$$\widehat{r}_T(s,a) \triangleq \frac{1}{T}\sum_{j=0}^{T-1} r_j(s,a), \quad \text{for all } (s,a) \in S \times A.$$

With this concept, we present the following definitions.

Definition 2.2 (Steady-state and Empirical-frequency Regret)**.** The *steady-state* average regret is

$$L_T^S \triangleq \sup_{\mu \in \Sigma} \langle \widehat{r}_T, \pi(\mu) \rangle - \frac{1}{T}\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t). \tag{2.2.4}$$

The *empirical-frequency* average regret is

$$L_T^E \triangleq \sup_{\mu\in\Sigma} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} \widehat{r}_T(\tilde{\mathbf{s}}_t, \tilde{\mathbf{a}}_t)\right] - \frac{1}{T}\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t). \tag{2.2.5}$$

Under Assumptions 2.1 and 2.2, these three definitions are asymptotically equivalent, as established in the following lemma. This result is independent of the agent's learning algorithm. The proofs of this and other lemmata are provided in Section 2.8.

Lemma 2.1 (Asymptotic Equivalence)**.** *If Assumptions 2.1 and 2.2 hold, then*

$$\left|L_T^E - L_T^S\right| \leq 2e\gamma/T,$$

and

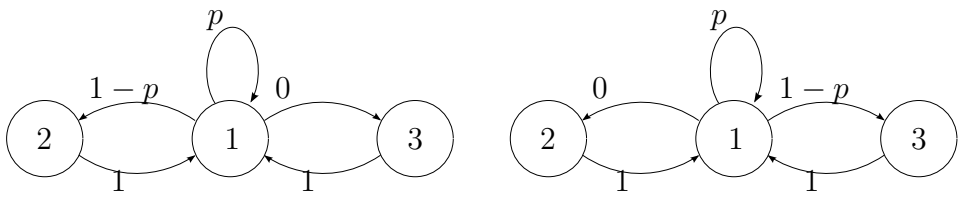
$$\left|L_T^W - L_T^S\right| \leq 2e\gamma/T.$$

This equivalence allows us to employ throughout our analysis the simpler notion of steady-state regret ((2.2.4)). We say that an agent's policy is a *no-regret* policy, with respect to one of the three definitions of regret, if the corresponding average regret tends to 0 with probability 1 as $T \to \infty$.

2.3 Counterexamples

In this section, we present examples where vanishing average regret can not be guaranteed. The first example considers a non-oblivious opponent that modifies the reward function according to the agent's action history. The second example displays a periodic state trajectory.

Example 2.1 (Non-oblivious Opponent)**.** Let the states $S = \{1, 2, 3\}$ be as in Figure 2–1. The agent has two actions to choose from: whether to go left or right. The corresponding transition probabilities are shown in Figure 2–1. The non-oblivious opponent assigns a reward of 0 to state 1



(a) Transition model if the agent chooses to go left.

(b) Transition model if the agent chooses to go right.

Figure 2–1: State transitions for Example 2.1. Taking the left action in state 1 leads to state 2 with probability $1 - p$. There is a small probability p of staying in state 1, regardless of the action taken; thus making the MDP aperiodic. From state 2 or 3, the agent moves to state 1 deterministically.

at all steps. It gives a reward of 1 to state 2 if the agent took the action leading to state 3 at the previous time step; otherwise, it gives zero reward to state 2. Similarly, the opponent gives a reward of 1 to state 3 if the agent took the action leading to state 2, and a zero reward otherwise. Consequently, for every policy, the reward attained by the agent is $\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) = 0$. In contrast, by computing the stationary distribution for each fixed action, we have either $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[r_t(\mathbf{s}_t, \text{left})\big] \geq (1-p)/(2-p)$ or $\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[r_t(\mathbf{s}_t, \text{right})\big] \geq (1-p)/(2-p)$. As a result, the average worst-case regret is always positive and bounded away from 0. Since the MDP is ergodic, a similar argument shows that the same holds true for the two other definitions of regret.

We note that this example is stronger than the counterexample presented in [91], where the non-vanishing regret is attributed to lack of observation of the reward.

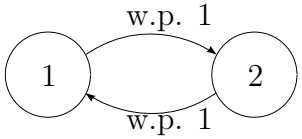


Figure 2–2: State transitions for Example 2.2.

Example 2.2 (Periodic MDP)**.** Consider an MDP with two states $S = \{1, 2\}$ as in Figure 2–2. The transition from state 1 to 2, and vice-versa, occurs with probability 1. The agent has a number of identical actions (same transitions and rewards). An oblivious opponent chooses the following rewards:

$$r_t(1) = 1, \quad r_t(2) = 0, \quad \text{if } t \text{ is even},$$
$$r_t(1) = 0, \quad r_t(2) = 1, \quad \text{if } t \text{ is odd}.$$

It follows that $\widehat{r}_T(1) \to 1/2$ as $T \to \infty$, and similarly for $\widehat{r}_T(2)$. If the initial state s_0 is 1, then the agent's cumulative reward is T; otherwise, if s_0 is 2, the cumulative reward is 0. This implies that the regret is either negative if $s_0 = 1$, or positive (and bounded away from zero) if $s_0 = 2$. Therefore, using the empirical-frequency or steady-state notion of regret, zero regret can not be achieved for periodic MDPs, even if the opponent is oblivious. Nonetheless, in this example, the regret is zero if we adopt the notion of worst-case regret (2.2.3). In this example, the value of the accumulated reward depends *solely* on the initial state s_0. Since we are interested in characterizing regret with respect to policies, such pathological cases shall be excluded.

In light of these counterexamples, we preclude via Assumptions 2.1 and 2.2 periodic MDPs and non-oblivious opponents.

2.4 Follow the Perturbed Leader

In this section, we present the basic algorithm of this chapter and show that it minimizes the regret under full observation of the reward functions.

2.4.1 Algorithm Description

The proposed algorithm is based on the concept, due to [64], of following the best action so far subject to random perturbations that vanish with time. The algorithm works in phases. We partition the time steps $0, 1, \ldots$ into phases

(*i.e.*, intervals of consecutive steps[2]), denoted by $\tau_0, \tau_1, \ldots$. We denote by M the number of phases up to step T. The phases are constructed long enough so that the state-action distribution approaches stationarity. As a result, the number of phases M also becomes sub-linear in T. The phases are nonetheless short enough so that the agent adapts fast enough to changes in the reward functions. This will be made precise in the results below. As a convention, we let the index t denote a step, whereas m denotes the index of phase τ_m. Moreover, we write $\tau_{0:m}$ to denote the union of phases $\tau_0 \cup \ldots \cup \tau_m$, and $|\tau_{0:m}|$ to denote its length. For ease of notation, we write the cumulative and average reward over one or more phases as

$$
\begin{aligned}
R_{\tau_m}(s,a) &\triangleq \sum_{t \in \tau_m} r_t(s,a), \\
\widehat{r}_{\tau_m}(s,a) &\triangleq \frac{1}{|\tau_m|} R_{\tau_m}(s,a), \\
\widehat{r}_{\tau_{0:m}}(s,a) &\triangleq \frac{1}{|\tau_{0:m}|} \sum_{t \in \tau_{0:m}} r_t(s,a),
\end{aligned}
$$

for all $(s,a) \in S \times A$. The algorithm takes as input the step index $t \in \tau_m$, the current state s_t, and the average reward function $\widehat{r}_{\tau_{0:m-1}}$. It outputs a random action $\mathbf{a}_t$. For the purpose of randomization, the algorithm samples a sequence $\mathbf{n}_1, \mathbf{n}_2, \ldots$ of independent random variables in $\mathbb{R}^{|A|}$. The distribution of these random variables will be specified later.

Remark 4 (Computational cost). The Lazy FPL algorithm, as well as the other algorithms presented in this chapter, has clear computational costs. In particular, the main step of the Lazy FPL algorithm solves a standard MDP

[2] The partition is constructed such that the order between steps within each phase is preserved.

1. (*Initialize.*) For $t \in \tau_0$, choose the action $\mathbf{a}_t$ according to an arbitrary stationary policy.
2. (*Update.*) At the start of phase τ_m, $m = 1, 2, \ldots$, solve the following linear program for (λ_m, h_m):

$$\begin{aligned} \min_{\lambda \in \mathbb{R}, h \in \mathbb{R}^{|S|}} \quad & \lambda \qquad (2.4.6) \\ \text{subject to:} \quad & \lambda + h(s) \geq \widehat{r}_{\tau_{0:m-1}}(s,a) + \sum_{s' \in S} P(s' \mid s,a) h(s'), \\ & \text{for every } (s,a) \in S \times A, \\ & h(s^+) = 0, \quad \text{for some fixed } s^+ \in S. \end{aligned}$$

3. (*Follow the perturbed leader.*) For $t \in \tau_m$, $m = 1, 2, \ldots$, choose the action

$$\mathbf{a}_t = \arg\max_{a \in A} \left\{ \widehat{r}_{\tau_{0:m-1}}(s_t, a) + \mathbf{n}_t(a) + \sum_{s' \in S} P(s' \mid s_t, a) h_m(s') \right\}, \qquad (2.4.7)$$

where the element of A with the lowest index is taken if the max is not unique.

Algorithm 2–1: Lazy FPL

with the average-reward objective (2.4.6). This step can further be approximated by more efficient iterative methods such as Q-learning. Moreover, we can use the methods of dynamic programming to deal with large state spaces [20].

Observe that the linear program (2.4.6) is a standard optimization problem for obtaining the optimal value function (and hence an optimal policy) in an average-reward MDP [19]. The Lazy FPL algorithm perturbs the average reward function $\widehat{r}_{\tau_{0:m-1}}$ with the random variable $\mathbf{n}_t$. Since the perturbing random variables $\mathbf{n}_t$ are identically distributed for all $t \in \tau_m$, while the other terms on the right-hand side of (2.4.7) are fixed, it follows that the actions $\mathbf{a}_t$ follow the same mixed stationary policy over the phase τ_m. We denote this policy by σ_m. The lazy aspect of this algorithm comes from the fact that it

updates its policy only once each phase, similar to other lazy learning schemes (*e.g.*, [97]).

There are two ingredients to this solution: randomization and phases. Randomization through small perturbations introduces continuity between policies from one phase to the next; thereby averting the precedent of Buridan's ass. This approach is reminiscent of the regret minimization technique in [64, 72] and smooth fictitious play in repeated games [55]. The motivation of increasing phase lengths is twofold. Firstly, using a fixed policy over long phases is computationally efficient. Secondly, in addition to vanishing *expected* regret, we show that the regret vanishes *almost surely*, provided that the agent does not change its policy too often. One intuition is that, on the one hand, our bases for comparison are the steady-state rewards of stationary policies; on the other hand, taking long phases ensures that the agent's accumulated reward in each phase approaches the steady-state reward of the corresponding policy. At the same time, we ensure that these phases are also short enough so that we may adapt to changes in the reward functions.

It is important to observe that prior knowledge of the time horizon T is not necessary to run the Lazy FPL algorithm. The only prerequisite is a pre-established scheme to partition every time interval into phases.

2.4.2 Results

In this section, we show that the Lazy FPL algorithm has the no-regret property. Our main result shows that increasing phase lengths in the Lazy FPL algorithm yields not only an efficient implementation, but also allows us to establish almost-sure convergence for the average regret. The proof relies on a probabilistic bound on the regret, which is derived using a modified version of Azuma's Inequality. The proof of this theorem will come after a number of intermediate results.

Theorem 2.2 (No-regret Property of Lazy FPL)**.** *Suppose that Assumptions 2.1 and 2.2 hold. Let the time horizon* $0, 1, \ldots$ *be partitioned into phases* $\tau_0, \tau_1, \ldots$ *such that* $|\tau_m| = \lceil m^{1/3} \rceil$*, for* $m = 0, 1, \ldots$*. Further, suppose that the random variables* $\mathbf{n}_t(a)$*, for* $t = 1, 2, \ldots$ *and* $a \in A$*, are independent and uniformly distributed*[3] *over the support* $[-1/\zeta_m, 1/\zeta_m]$*, where* $\zeta_m \triangleq \sqrt{|\tau_{0:m}|}$ *and* $t \in \tau_m$*. Then, the average regret of the Lazy FPL algorithm vanishes almost surely,* i.e.*,*

$$\limsup_{T \to \infty} L_T^W \leq 0, \quad \textit{w.p. } 1.$$

Remark 5. Theorem 2.2 makes no assumption about the sequence of reward functions $r_0, r_1, \ldots$ except for boundedness and obliviousness.

Remark 6. Observe that the partition of Theorem 2.2 can be constructed incrementally over time, without prior knowledge of the time horizon T. Moreover, having a slowly increasing phase length suffices for obtaining convergence.

Theorem 2.2 builds upon the following proposition that establishes the rate of convergence of the expected average regret under the Lazy FPL algorithm.

Proposition 2.3 (Expected Regret Bound)**.** *Suppose that the assumptions of Theorem 2.2 hold. In particular, suppose that* $|\tau_m| = \lceil m^{1/3} \rceil$*, for* $m = 0, 1, \ldots$*. Then, the expected average regret of the Lazy FPL algorithm is bounded as*

[3] The random variable $\mathbf{n}_t(a)$ has probability density function

$$f_{\mathbf{n}_t(a)}(z) = \begin{cases} \zeta_m/2, & \text{if } z \in [-1/\zeta_m, 1/\zeta_m], \\ 0, & \text{otherwise.} \end{cases}$$

follows:

$$\mathbb{E}\big[L_T^W\big] \leq \frac{4}{3}\big(2e\gamma + 2\,|A| + 4e + 1 + 2(|S|+3)\,|A|^2\,\gamma\log(T)\big)\;T^{-1/4}. \quad (2.4.8)$$

Remark 7. The bound of (2.4.8) is weaker than the $O(T^{-1/2})$ bound that was obtained for the algorithm of [44]. This can be attributed to the fact that the Lazy FPL algorithm computes a single policy each phase and follows it throughout increasingly long phases. It is a common feature of lazy learning schemes (cf. *e.g.*, [97]).

The proof of Proposition 2.3 relies on the following lemmata. The proofs of the lemmata are postponed to Section 2.8. The first lemma gives a convenient expression for expected regret.

Lemma 2.4. *Let s_0 be an arbitrary state and μ be an arbitrary stationary policy. Let $(\mathbf{s}_t, \mathbf{a}_t)$ be the state-action pair at step t following policy μ and starting at initial state s_0. If the opponent is oblivious (Assumption 2.2), then for every $j = 0, \ldots, T-1$, we have*

$$\mathbb{E}\big[r_j(\mathbf{s}_t, \mathbf{a}_t)\big] = \langle r_j, d_t(\mu; s_0)\rangle, \quad (2.4.9)$$

where the expectation is taken over both the MDP transitions and the randomization of policy μ.

Let $t \in \tau_m$. We define the following unperturbed counterpart to the action $\mathbf{a}_t$ of (2.4.7):

$$\mathbf{a}_t^+ = \arg\max_{a\in A}\Big\{\widehat{r}_{\tau_{0:m-1}}(s_t, a) + \sum_{s'\in S} P(s' \mid s_t, a) h_m(s')\Big\},$$

where h_m is part of the solution to the linear program (2.4.6). Note that $\mathbf{a}_t$ is a random variable whereas $\mathbf{a}_t^+$ is deterministic given the reward sequence. We

also define the following stationary policies, for all $(s, a) \in S \times A$,

$$\sigma_m(a; s) = \Pr(\mathbf{a}_t = a \mid \mathbf{s}_t = s),$$
$$\sigma_m^+(a; s) = \Pr(\mathbf{a}_t^+ = a \mid \mathbf{s}_t = s).$$

Note that σ_m is a mixed policy, whereas σ_m^+ is a deterministic one. Both are determined by the sequence of reward functions, and hence, independent of the state-trajectory. The following lemma—a consequence of [19, Section 4.3.3]—asserts the optimality of σ_m^+.

Lemma 2.5 (Optimality)**.** *Suppose that Assumption 2.1 holds. In phase τ_m, the policy σ_m^+ is optimal against the reward function $\widehat{r}_{\tau_{0:m-1}}$ in the sense that*

$$\langle \widehat{r}_{\tau_{0:m-1}}, \pi(\sigma_m^+) \rangle \geq \sup_{\mu \in \Sigma} \langle \widehat{r}_{\tau_{0:m-1}}, \pi(\mu) \rangle ,$$

where $\pi(\sigma_m^+)$ is the stationary state-action distribution corresponding to policy σ_m^+.

Next, we bound the rate of change of the empirical average reward function.

Lemma 2.6 (Difference in Partial Averages)**.** *Let n and ℓ be non-negative integers such that $n \geq \ell$, then*

$$\left\| \frac{1}{n} \sum_{j=0}^{n-1} r_j - \frac{1}{\ell} \sum_{j=0}^{\ell-1} r_j \right\|_\infty \leq 2 \, \frac{n - \ell}{n}.$$

The following lemma quantifies the change in policy of the Lazy FPL algorithm from phase to phase.

Lemma 2.7 (Policy Continuity)**.** *Suppose that the assumptions of Theorem 2.2 hold. Then, for $m = 0, 1, \ldots$, every $s \in S$, and for every positive integer g,*

$$\|\sigma_{m+1}(\cdot; s) - \sigma_m(\cdot; s)\|_1 \leq (|S| + 3) \, |A|^2 \left(\zeta_{m+1} \frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1} - \zeta_m}{\zeta_{m+1}} \right),$$

and

$$\|\pi(\sigma_{m+1}) - \pi(\sigma_m)\|_1 \leq (|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) g + 4e^{1-g/\gamma}.$$

The following lemma characterizes the effect of randomization in the Lazy FPL algorithm on the expected cumulative reward.

Lemma 2.8 (Effect of Randomization)**.** *Suppose that the assumptions of Theorem 2.2 hold. For phases indexed $m = 1, 2, \ldots$, we have*

$$\langle R_{\tau_{0:m-1}}, \pi(\sigma_m)\rangle \geq \langle R_{\tau_{0:m-1}}, \pi(\sigma_m^+)\rangle - 2\,|A|\,\frac{|\tau_{0:m-1}|}{\zeta_m^2}.$$

We now prove Proposition 2.3 and Theorem 2.2.

Proof. Proposition 2.3 The proof proceeds along the following lines. The oblivious opponent assumption makes stationary policies as good as any other within long phases. The ergodicity assumption allows us to concentrate on the stationary distributions of the baseline policies, as well as the policies of the sequence of phases. The perturbation noise enforces a certain continuity between policies of consecutive phases, yet it vanishes quickly enough as not to severely affect the optimality of the stationary policy computed at each phase. Letting M denote the number of phases up to time step T, we divide the proof into the following sequence of bounds.

$$\begin{aligned}
&\sum_{t=0}^{T-1} \mathbb{E}\big[r_t(\mathbf{s}_t, \mathbf{a}_t)\big] \\
\text{(Step 0)}\quad &\geq \sum_{m=0}^{M-1} \Big(\langle R_{\tau_m}, \pi(\sigma_m)\rangle - 2e\gamma\Big) && (2.4.10)\\
\text{(Step 1)}\quad &\geq \sum_{m=0}^{M-2} \big(\langle R_{\tau_m}, \pi(\sigma_{m+1})\rangle - 2e\gamma - 4e - 2(|S|+3)\,|A|^2\,\gamma\log(T)\big) \\
\text{(Step 2)}\quad &\geq T\cdot\sup_{\mu\in\Sigma}\langle \widehat{r}_T, \pi(\mu)\rangle - (M-1)\left(2e\gamma + 4e + 2(|S|+3)\,|A|^2\,\gamma\log(T)\right) \\
&\quad - 2(M-1)\,|A| - M^{1/3}, && (2.4.11)
\end{aligned}$$

where the expectation $\mathbb{E}$ is over both the MDP transitions and the randomization through $\mathbf{n}_t$ in Algorithm 2–1. (2.4.8) now follows from (2.4.11) by Lemma 2.1 and the fact that since $|\tau_m| = \lceil m^{1/3} \rceil$ for $m = 0, \ldots, M-1$, we have $M \leq (4/3)T^{3/4}$.

Step 0. Let s^- denote the state at the beginning of phase τ_m. By Lemma 2.4 and Assumption 2.1, for every phase τ_m, we have

$$\begin{aligned}\sum_{t\in\tau_m} \mathbb{E}\big[r_t(\mathbf{s}_t, \mathbf{a}_t) \mid s^-\big] &= \sum_{t\in\tau_m} \langle r_t, d_t(\sigma_m; s^-)\rangle \\ &\geq \sum_{t\in\tau_m} \langle r_t, \pi(\sigma_m)\rangle - \sum_{t=0}^{|\tau_m|-1} 2e^{1-t/\gamma} \\ &\geq \langle R_{\tau_m}, \pi(\sigma_m)\rangle - 2e\gamma,\end{aligned}$$

as in (2.4.10).

Step 1. By Lemma 2.7 with $\zeta_m = \sqrt{|\tau_{0:m}|}$ for $m = 0, 1, \ldots$, and by picking $g = \gamma \log(|\tau_{0:m+1}|)$, we obtain

$$\begin{aligned}\|\pi(\sigma_m) - \pi(\sigma_{m+1})\|_1 &\leq g(|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) + 4e^{1-g/\gamma} \\ &\leq 2(|S|+3)\,|A|^2\,\gamma\frac{|\tau_{m+1}|\log(|\tau_{0:m+1}|)}{|\tau_{0:m+1}|^{1/2}} + \frac{4e}{|\tau_{0:m+1}|}.\end{aligned}$$

It follows that

$$\begin{aligned}&\sum_{m=0}^{M-1} \langle R_{\tau_m}, \pi(\sigma_m)\rangle \\ &\geq \sum_{m=0}^{M-2} |\tau_m|\,\langle \widehat{r}_{\tau_m}, \pi(\sigma_{m+1})\rangle \\ &\quad - |\tau_m| \left(2(|S|+3)\,|A|^2\,\gamma\frac{|\tau_{m+1}|\log(|\tau_{0:m+1}|)}{|\tau_{0:m+1}|^{1/2}} + \frac{4e}{|\tau_{0:m+1}|}\right) \\ &\geq \sum_{m=0}^{M-2} \big(\langle R_{\tau_m}, \pi(\sigma_{m+1})\rangle - 2(|S|+3)\,|A|^2\,\gamma\log(T) - 4e\big)\,,\end{aligned}$$

where the second inequality follows from the construction of the partition. Indeed, choosing $|\tau_m| = \lceil m^{1/3} \rceil$ for $m = 0, \ldots, M-1$ implies that

$$|\tau_m| \, |\tau_{m+1}| \log(|\tau_{0:m+1}|) \leq \log(T) \, |\tau_{0:m+1}|^{1/2} .$$

Step 2. In this step, we show that by taking into account rewards for phases $\tau_{m+1}, \ldots, \tau_{M-1}$, we cannot improve the expected reward for phases $\tau_1, \ldots, \tau_{m-1}$. To this end, we show by induction on $J = 0, \ldots, M-2$ that

$$\sum_{m=0}^{M-2} \langle R_{\tau_m}, \pi(\sigma_{m+1}) \rangle \geq \sum_{m=0}^{M-2} \langle R_{\tau_m}, \pi(\sigma_{M-1}) \rangle - 2(M-2) \, |A| \tag{2.4.12}$$

For the base case of $J = 0$, we clearly have

$$\langle R_{\tau_0}, \pi(\sigma_1) \rangle \geq \langle R_{\tau_0}, \pi(\sigma_1) \rangle.$$

Assume that for some J, we have

$$\sum_{m=0}^{J} \langle R_{\tau_m}, \pi(\sigma_{m+1}) \rangle \geq \sum_{m=0}^{J} \langle R_{\tau_m}, \pi(\sigma_{J+1}) \rangle - 2J \, |A| .$$

Then

$$\begin{aligned}
\sum_{m=0}^{J} \langle R_{\tau_m}, \pi(\sigma_{m+1}) \rangle &\geq \langle R_{\tau_{0:J}}, \pi(\sigma_{J+1}) \rangle - 2J \, |A| \\
&\geq \langle R_{\tau_{0:J}}, \pi(\sigma^+_{J+1}) \rangle - 2 \, |A| \, \frac{|\tau_{0:J}|}{\zeta^2_{J+1}} - 2J \, |A| \\
&\geq \langle R_{\tau_{0:J}}, \pi(\sigma_{J+2}) \rangle - 2(J+1) \, |A| ,
\end{aligned}$$

where the first inequality follows by definition, the second inequality follows from Lemma 2.8, and the third inequality use the assumption that $\zeta_m = \sqrt{|\tau_{0:m}|}$ and the optimality of the policy σ^+_{J+1}. Finally, adding $\langle R_{\tau_{J+1}}, \pi(\sigma_{J+2}) \rangle$ to both sides of the above inequalities, we complete the induction step:

$$\sum_{m=0}^{J+1} \langle R_{\tau_m}, \pi(\sigma_{m+1}) \rangle \geq \sum_{m=0}^{J+1} \langle R_{\tau_m}, \pi(\sigma_{J+2}) \rangle - 2(J+1) \, |A| ,$$

and (2.4.12) follows.

Finally, observe that

$$\begin{aligned}
&\sum_{m=0}^{M-2} \langle R_{\tau_m}, \pi(\sigma_M) \rangle - 2(M-2)\,|A| \\
&\geq \sum_{m=0}^{M-1} \langle R_{\tau_m}, \pi(\sigma_M^+) \rangle - 2(M-2)\,|A| - 2\,|A| - |\tau_{M-1}| \qquad (2.4.13)
\end{aligned}$$

by Lemma 2.8 and the fact that σ_M^+ is an optimal policy in an MDP with reward function $\widehat{r}_T \triangleq \widehat{r}_{\tau_{0:M-1}}$. (2.4.13) uses the fact that the reward attained in phase τ_{M-1} is bounded by $|\tau_{M-1}| \leq M^{1/3}$. The required result of (2.4.11) follows by observing that

$$\sum_{m=0}^{M-1} \langle R_{\tau_m}, \pi(\sigma_M^+) \rangle = T \langle \widehat{r}_T, \pi(\sigma_M^+) \rangle = T \cdot \sup_{\mu \in \Sigma} \langle \widehat{r}_T, \pi(\mu) \rangle,$$

where the first equality is due to the linearity of the inner product and the definition of $\widehat{r}_T$, and the second equality is due to the optimality of σ_M^+ against $\widehat{r}_T$. ■

Proof. Theorem 2.2 The proof relies on a modified version of Azuma's Inequality [34, Appendix A.6]. We first define:

$$\begin{aligned}
\mathbf{V}_m &= \sum_{t \in \tau_m} \mathbb{E}\big[r_t(\mathbf{s}_t, \mathbf{a}_t)\big] - r_t(\mathbf{s}_t, \mathbf{a}_t), \\
\mathbf{W}_{M-1} &= \sum_{m=0}^{M-1} \mathbf{V}_m.
\end{aligned}$$

By Assumption 2.1, for all m, we have (with probability 1)

$$\mathbb{E}\big[\mathbf{V}_m \mid \mathbf{s}_t, \mathbf{a}_t, \text{ for } t \in \tau_{0:m-1}\big] = 0.$$

Next, observe that for every real-valued x,

$$\begin{aligned}\mathbb{E}\left[e^{x\mathbf{W}_{M-1}}\right] &= \mathbb{E}\left[e^{x\mathbf{W}_{M-2}}\mathbb{E}[e^{x\mathbf{V}_{M-1}} \mid \mathbf{s}_t, \mathbf{a}_t, \text{ for } t \in \tau_{0:M-2}]\right] \\ &\leq \mathbb{E}\left[e^{x\mathbf{W}_{M-2}}\right] \exp\left(\frac{x^2}{8} 4 \left|\tau_{M-1}\right|^2\right),\end{aligned}$$

where the inequality follows from [34, Lemma A.1]. By recursion on M, we obtain

$$\mathbb{E}\left[e^{x\mathbf{W}_{M-1}}\right] \leq \exp\left(\frac{x^2}{2} \sum_{m=0}^{M-1} \left|\tau_m\right|^2\right).$$

By Chebychev's Inequality, for every real x, we obtain

$$\begin{aligned}\Pr\left(\frac{1}{T}\mathbf{W}_{M-1} > \delta\right) &\leq \frac{\mathbb{E}\left[e^{x\mathbf{W}_{M-1}}\right]}{e^{x\delta T}} \\ &\leq \exp\left(-\frac{\left(\delta T\right)^2}{2\sum_{m=0}^{M-1}\left|\tau_m\right|^2}\right), \qquad (2.4.14)\end{aligned}$$

where the second inequality is obtained by choosing x to minimize the exponent. Next, observe that the phase partition $|\tau_m| = \lceil m^{1/3} \rceil$ defined in Proposition 2.3 implies that $M \leq (4/3)T^{3/4}$. Hence, we have $\sum_{m=0}^{M-1}\left|\tau_m\right|^2 \leq (3/5)M^{5/3} \leq (4/5)T^{5/4}$. Following substitutions, we obtain

$$\begin{aligned}\Pr\left(\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[r_t(\mathbf{s}_t, \mathbf{a}_t)\right] - \frac{1}{T}\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) > \delta\right) &\leq \exp\left(-\frac{\delta^2\, T^2}{(8/5)T^{5/4}}\right) \\ &= \exp\left(-(5/8)\ \delta^2\ T^{3/4}\right).\end{aligned}$$

Therefore, the right-hand side of (2.4.14) is summable over non-negative integers T for every $\delta > 0$. An application of Proposition 2.3 and the Borel-Cantelli Lemma completes the proof. ∎

2.5 Approximate Algorithms

In many cases of interest, computing the exact policy σ_m at each phase τ_m of the Lazy FPL algorithm might be intractable due to the size of the state space. One solution is to compute an approximation ρ_m to σ_m. The policy

ρ_m is still computed once every phase, but using a computationally efficient method. We consider the approach of approximating the optimal state-action value function or *Q-function*. Recall that in average-reward MDPs, the Q-function $Q : S \times A \to \mathbb{R}$ represents the relative utility of choosing a particular action at a particular state. Let (λ_m, h_m) denote the optimal solution to the linear program (2.4.6) at the start of phase τ_m. The corresponding optimal Q-function is therefore defined as

$$Q_m^*(s,a) = \widehat{r}_{\tau_{0:m-1}}(s,a) + \sum_{s' \in S} P(s' \mid s,a) h_m(s').$$

Definition 2.3. Let ϵ and δ be non-negative constants. Consider an algorithm that computes an approximate Q-function Q_m for each phase τ_m, and chooses an action

$$\mathbf{a}_t = \arg\max_{a \in A} \Big\{ Q_m(s_t, a) + \mathbf{n}_t(a) \Big\}$$

at every step t in phase τ_m, with the random variable $\mathbf{n}_t$ distributed as in Theorem 2.2. Such an algorithm is an (ϵ, δ)-*approximation* algorithm if there exists an integer N such that, for $m \geq N$,

$$\Pr\Big(\|Q_m(s,\cdot) - Q_m^*(s,\cdot)\|_1 \leq \epsilon, \quad \text{for every } s \in S \Big) \geq 1 - \delta, \tag{2.5.15}$$

where Q_m^* is the optimal Q-function.

The following corollary (proved in Section 2.8) relaxes the need for an exact optimization procedure.

Corollary 2.9. *Let $\mathbf{P}_\sigma$ denote the matrix whose (s', s)-element is $P(s' \mid s, \sigma(s))$,* i.e.*, the transition matrix induced by the stationary policy $\sigma : S \to A$. Let $\mathbf{Z}_\sigma$ denote the fundamental matrix (cf. [115]) associated with the same*

transition kernel $P(s' \mid s, \sigma(s))$. *In other words,*

$$\mathbf{Z}_\sigma \triangleq [\mathbf{I} - \mathbf{P}_\sigma + \mathbf{P}_\sigma^\infty]^{-1}, \quad \textit{where} \quad \mathbf{P}_\sigma^\infty \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \mathbf{P}_\sigma^k.$$

Further, let the norm $\|\mathbf{M}\|_\infty$ *of a matrix* $\mathbf{M}$ *denote its maximum absolute row-sum. Suppose that the assumptions of Theorem 2.2 hold. The average regret of an* (ϵ, δ)*-approximation algorithm is bounded as follows:*

$$\limsup_{T\to\infty} L_T^W \leq \sup_{\sigma\in\Sigma} \|\mathbf{Z}_\sigma\|_\infty (\epsilon + \delta), \quad w.p.\ 1.$$

Remark 8. If an algorithm is an (ϵ, δ)-approximation for every pair of positive numbers ϵ and δ, then the average regret tends to zero almost surely. It is also possible to obtain almost-sure convergence of the average regret to zero if the Q-functions Q_m computed by an approximation algorithm improve in accuracy from phase to phase, such that (2.5.15) holds for sequences ϵ_m and δ_m that decrease quickly enough to zero.

In the following algorithm, we use Q-learning [20, Chapter 7] to compute an approximation ρ_m to the policy σ_m of the Lazy FPL algorithm. In essence, Q-learning is employed as a method of solving the linear program of the Lazy FPL algorithm. It is well known that Q-learning is an iterative simulation-based method that does not need to keep track of the transition probabilities. Let Q_t denote the sequence of Q-functions, and $Q_{\tau_{0:m-1}}$ denote the Q-function obtained at the last step of phase τ_{m-1}. During every step t of phase τ_m, we choose our action to maximize the Q-function $Q_{\tau_{0:m-1}}$ obtained over the previous phases, perturbed by a random term $\mathbf{n}_t$; simultaneously, we update the sequence of Q-functions Q_t at every step.

Remark 9. As for the Lazy FPL algorithm, the reward function $\widehat{r}_{\tau_{0:m-1}}$ is fixed throughout phase τ_m.

1. (*Initialize.*) For $t \in \tau_0$, set $Q_t = 0$ and choose action $\mathbf{a}_t$ according to an arbitrary deterministic policy $\mu : S \to A$.
2. (*Update.*) At every step $t \in \tau_m$, for $m = 1, 2, \ldots$, set $\kappa_m = 1/\sqrt{m}$ and update Q_t iteratively as follows:

$$Q_t(s_{t-1}, a_{t-1}) = (1 - \kappa_m) Q_{t-1}(s_{t-1}, a_{t-1}) + \kappa_m \Big(\widehat{r}_{\tau_{0:m-1}}(s_{t-1}, a_{t-1}) + \max_{a \in A} Q_{t-1}(s_t, a) - Q_{t-1}(s', a') \Big), \tag{2.5.16}$$

   where s' and a' are fixed, and the term $Q_{t-1}(s', a')$ serves the purpose of normalization.
3. (*Perturb.*) At every step $t \in \tau_m$, for $m = 1, 2, \ldots$, choose action

$$\mathbf{a}_t = \arg\max_{a \in A} \left\{ Q_{\tau_{0:m-1}}(s_t, a) + \mathbf{n}_t(a) \right\},$$

   where the random variables $\mathbf{n}_t$ are distributed as in Theorem 2.2.

Algorithm 2–2: Q-FPL

The sequence κ_m is selected such that it satisfies the conditions for stochastic approximation (cf. Section 4.3 of [29]). Let $Q^*_{\tau_{0:m-1}}$ denote the optimal Q-function against the fixed reward function $\widehat{r}_{\tau_{0:m-1}}$. By [29, Theorem 2.4], within each phase where the reward function is fixed and the length is long enough, for every $\beta > 0$ and $\gamma > 0$, we have

$$\Pr\left(\left\| Q_{\tau_{0:m-1}} - Q^*_{\tau_{0:m-1}} \right\|_1 > \beta \right) < \gamma, \tag{2.5.17}$$

so that (2.5.15) holds[4] . We observe that the Q-FPL Algorithm is in fact an (ϵ, δ)-approximation algorithm for every positive ϵ and δ, which leads to the following corollary by an argument similar to Theorem 2.2.

[4] To be accurate, for the off-policy Q-function evaluation in Step 2 of the Q-FPL algorithm to converge at the end of each phase, we must ensure that the policy induced by Step 3 performs sufficient exploration. Hence, we sample an independent perturbation $\mathbf{n}_t$ at every time step.

Corollary 2.10. *Suppose that the assumptions of Theorem 2.2 hold. Then, the average regret of the Q-FPL algorithm tends to zero almost surely.*

Other algorithms, especially some actor-critic algorithms that are equivalent to Q-learning [39], may be used as well, so long as they are (ϵ, δ)-approximations for every pair of positive ϵ and δ.

Remark 10 (Computational Load). The Q-FPL algorithm has a fixed computational load per step. This complexity is less demanding than that of [44], although the latter is also fixed per step. In comparison, the Lazy FPL algorithm requires solving an MDP at the beginning of every phase, but it has the advantage of diminishing the per-step complexity.

2.6 Observing Rewards Only on Trajectory

In this section, we present a modification of the Lazy FPL algorithm in the spirit of [10] to deal with instances where the reward functions are partially observed. More precisely, we consider the case where the agent only observes the value of the reward function sequence on the traversed state-action trajectory. Consequently, we restrict the space of the agent's policies to those that map the observed reward-history $r_0(s_0, a_0), \ldots, r_{t-1}(s_{t-1}, a_{t-1})$ and the current state s_t to a mixed action.

Our approach is to construct an unbiased estimate of $\widehat{r}_{\tau_{0:m-1}}$ at each phase τ_m. Following an initialization phase τ_0, we construct a *random* reward function at every step t. The length of the phase τ_0 and the policy adopted therein are such that, for $t \geq |\tau_0|$, $\Pr\left((\mathbf{s}_t, \mathbf{a}_t) = (s, a) \mid s_0\right) > 0$ for all $(s, a) \in S \times A$. For all $t \geq |\tau_0|$ and $(s, a) \in S \times A$, we let

$$\mathbf{z}_t(s, a) = \begin{cases} \frac{r_t(s,a)}{\Pr\left((\mathbf{s}_t, \mathbf{a}_t) = (s,a) \mid s_0\right)}, & \text{if } (\mathbf{s}_t, \mathbf{a}_t) = (s, a), \\ 0, & \text{otherwise.} \end{cases}$$

Observe that only the value of r_t at the traversed state-action pair $(\mathbf{s}_t, \mathbf{a}_t)$ is required to evaluate $\mathbf{z}_t$. The probability $\Pr\big((\mathbf{s}_t, \mathbf{a}_t) = (s, a) \mid s_0\big)$ is readily computed recursively using the transition probabilities associated with the policy followed at step $t-1$. From the sequence $\mathbf{z}_j$, we construct $\widehat{\mathbf{z}}_t \triangleq \frac{1}{t}\sum_{j=0}^{t-1} \mathbf{z}_j$ as an estimate of $\widehat{r}_t = \frac{1}{t}\sum_{j=0}^{t-1} r_j$. In conformance with our notation, $\widehat{\mathbf{z}}_{\tau_{0:m-1}}$ denotes $\widehat{\mathbf{z}}_t$, where t is the first step of phase τ_m.

1. (*Initialize.*) Let the length of phase τ_0 be long enough that $\Pr\big((\mathbf{s}_t, \mathbf{a}_t) = (s, a) \mid s_0\big) > 0$ for $t \geq |\tau_0|$ and $(s, a) \in S \times A$. For $t \in \tau_0$, choose action $\mathbf{a}_t$ uniformly at random over A.
2. (*Estimate.*) At every step $t = 1, 2, \ldots$, compute the estimate $\widehat{\mathbf{z}}_t$ recursively.
3. (*Sample.*) At the start of phase τ_m, for $m = 1, 2, \ldots$, sample an independent Bernoulli random variable $\mathbf{x}_m$ that takes value 1 with probability ϕ_m.
4. (*Lazy FPL.*) If $\mathbf{x}_m = 0$, by substituting $\widehat{\mathbf{z}}_{\tau_{0:m-1}}$ for $\widehat{r}_{\tau_{0:m-1}}$, solve the linear program (2.4.6) and follow the policy of (2.4.7) throughout phase τ_m.
5. (*Explore.*) If $\mathbf{x}_m = 1$, for $t \in \tau_m$ and $m = 1, 2, \ldots$, choose action $\mathbf{a}_t$ uniformly at random over A.

Algorithm 2–3: Exploratory FPL

The following corollary (see Section 2.8 for a proof outline) asserts a result analogous to Theorem 2.2 for the Exploratory FPL algorithm (Algorithm 2–3).

Corollary 2.11 (No-regret Property of Exploratory FPL)**.** *Suppose that the assumptions of Theorem 2.2 hold. Let M denote the number of phases up to time step T. Suppose that the agent follows the Exploratory FPL algorithm with a sequence $\phi_m > 0$, for $m = 0, \ldots, M-1$, ensuring infinitely many exploration phases, and such that*

$$\sum_{m=0}^{M-1} |\tau_m|\, \phi_m = O(M). \tag{2.6.18}$$

Then, the average regret of the Exploratory FPL algorithm vanishes almost surely.

Remark 11. If ϕ_m is set to a positive constant, then the Exploratory FPL algorithm reduces to an approximation algorithm governed by Corollary 2.9.

Remark 12. Corollary 2.11 guarantees that the Exploratory FPL algorithm minimizes regret in generalized multi-arm bandit problems with a state variable.

2.7 Regret with Respect to Dynamic Policies

In this section, we consider a more general notion of regret that encompasses some dynamic policies. Consider a sequence of policies $\vec{\mu} = (\mu_0, \ldots, \mu_{T-1})$, where every element μ_j of the sequence is a deterministic policy $\mu_j : S \to A$. Let the number of switches in this sequence of policies be

$$K(\vec{\mu}) = \sum_{j=1}^{T-1} 1_{[\mu_{j-1} \neq \mu_j]}.$$

Let K_0 be a fixed integer. A more challenging baseline of comparison for the cumulative reward is

$$\mathcal{B}_T(K_0) \triangleq \sup_{\substack{(\mu_0,\ldots,\mu_{T-1}): \\ K(\vec{\mu}) \leq K_0}} \mathbb{E}\left[\sum_{t=0}^{T-1} r_t(\tilde{\mathbf{s}}_t, \mu_t(\tilde{\mathbf{s}}_t))\right], \tag{2.7.19}$$

where $(\tilde{\mathbf{s}}_0, \mu_0(\tilde{\mathbf{s}}_0)), \ldots, (\tilde{\mathbf{s}}_{T-1}, \mu_{T-1}(\tilde{\mathbf{s}}_{T-1}))$ denote state-action pairs induced by the sequence of policies $\mu_0, \ldots, \mu_{T-1}$, and the maximum is taken over all possible sequences of policies with at most K_0 switches. If $K_0 = 0$, then (2.7.19) reduces to the baseline considered so far (cf. (2.2.3)). We present an algorithm that guarantees a reward consistent with the above baseline. This algorithm adapts the Fixed-share algorithm of [70] to the MDP framework.

Remark 13. Observe that, as before, the algorithm elects a single policy in each phase and follows it throughout. The fixed-share scheme occurs *once in each phase*—at the outset. Observe also that the uniformly random policy $\mathbf{y}_m$ can be constructed efficiently. As in the Fixed-share algorithm of [70], the action

1. (*Initialize.*) Fix $\alpha \in [0,1]$. For $t \in \tau_0$, choose action $\mathbf{a}_t$ according to an arbitrary deterministic policy $\mu : S \to A$.
2. (*Sample.*) At the outset of phase τ_m, for $m = 1, 2, \ldots$, sample a Bernoulli random variable $\mathbf{x}_m$ with $\Pr(\mathbf{x}_m = 1) = \alpha$.
3. (*Fixed-share.*) If $\mathbf{x}_m = 0$, sample a policy $\mathbf{y}_m$ uniformly at random from the set of deterministic policies $\{\mu : S \to A\}$, then follow the policy $\mathbf{y}_m$ throughout phase τ_m.
4. (*Lazy FPL.*) If $\mathbf{x}_m = 1$, solve the linear program (2.4.6) and follow the policy of (2.4.7) throughout phase τ_m.

Algorithm 2–4: Tracking FPL

at each step is equal to the previous action with probability $1 - \alpha + \alpha/|A|$, and equal to each different action with probability $\alpha/|A|$.

Remark 14. In the MDP setting, the most obvious extension of the Fixed-share algorithm of [70] is to associate an expert to every deterministic policy $\mu : S \to A$. This creates an exponential number of such experts, which our approach avoids.

The following analog of Theorem 2.2 guarantees that the regret with respect to the reward achieved by the best sequence of policies with a finite number of switches vanishes asymptotically if the agent employs the Tracking FPL algorithm.

Theorem 2.12 (No-regret Property of Tracking FPL)**.** *Suppose that the assumptions of Theorem 2.2 hold. Let K_0 be a positive integer. Suppose further that the agent follows the Tracking FPL algorithm with the parameter $\alpha = K_0/(\lceil T/\lceil T^{1/3}\rceil\rceil - 1)$. Then, the average regret with respect to the baseline of* (2.7.19) *vanishes almost surely,* i.e.*,*

$$\limsup_{T\to\infty} \left\{ \frac{1}{T}\mathcal{B}_T(K_0) - \frac{1}{T}\sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) \right\} \leq 0, \quad \textit{w.p. } 1.$$

Remark 15. Although we only consider the case of a fixed number of switches K_0 and a fixed parameter α, it can be shown, by using doubling trick of [34,

Section 3.2], that the result of Theorem 2.12 holds as long as the number of switches K_0 increases slowly enough in T.

The proof of this theorem hinges on a bound on the rate of convergence of the expected regret similar to Proposition 2.3. To derive this bound, we first prove a bound for a different hypothetical—and less practical—algorithm. Consider Algorithm 2–5: a modified version of the exponentially weighted average forecaster [34], which also resembles the algorithm of [44]. To every deterministic policy $\mu : S \to A$, we associate a weight $w_m(\mu)$ that is updated at every phase τ_m, for $m = 0, 1, \ldots$. Once at the start of every phase, the algorithm picks a deterministic policy with probability proportional to its weight, and follows this policy throughout the phase. The weights are adjusted in the spirit of the Fixed-share algorithm [70] to track infrequent changes in optimal policy.

Remark 16. The main problem with the Lazy Tracking Forecaster algorithm is that the number of weight variables $|A|^{|S|}$ is exponential in the size of the state space.

Remark 17. The term $\langle R_{\tau_{m-1}}, \pi(\mu) \rangle$ in (2.7.20) approximates the expected reward accumulated by following policy μ over the course of phase τ_{m-1}. The weights are updated recursively according to each policy's reward over the previous phase. The probability measure defined in (2.7.21) tracks the optimal policy in the fashion of the Fixed-share algorithm [70].

Remark 18. In contrast to the algorithms presented in the previous sections, the length of every phase is kept the same. By using the doubling trick of [34, Section 3.2], we can adapt the Lazy Tracking Forecaster algorithm to problems where the time horizon T is unknown. This technique partitions the time horizon into periods of exponentially increasing length and runs the Lazy Tracking Forecaster algorithm on each period independently.

1. (*Initialize.*) Fix $\alpha \in [0,1]$ and $\eta \in (0,\infty)$. For every deterministic policy $\mu : S \to A$, set

$$w_0(\mu) = \frac{1}{|A|^{|S|}}.$$

2. (*Update weights & choose policy.*) At the start of every phase τ_m, for $m = 1, 2, \ldots$, evaluate

$$w_m(\mu) = w_{m-1}(\mu)\ \exp\left(\eta\ \langle R_{\tau_{m-1}}, \pi(\mu)\rangle\right), \qquad \text{for every } \mu : S \to A. \tag{2.7.20}$$

Sample a random variable $\mathbf{q}_m$ over the set of deterministic policies $\{\mu : S \to A\}$ and with the following probability measure[a] :

$$\Pr(\mathbf{q}_m = \mu) = (1-\alpha)\frac{w_m(\mu)}{\sum_{\mu' : S\to A} w_m(\mu')} + \alpha\frac{1}{|A|^{|S|}}, \qquad \text{for all } \mu : S \to A. \tag{2.7.21}$$

3. (*Follow chosen policy.*) For $t \in \tau_m$, and $m = 1, 2, \ldots$, choose the action $\mathbf{a}_t = \mathbf{q}_m(s_t)$.

[a] As in the Fixed-share algorithm of [70], the action at each step is equal to the previous action with probability $1 - \alpha + \alpha/|A|$, and equal to each different action with probability $\alpha/|A|$.

Algorithm 2–5: Lazy Tracking Forecaster

As asserted in the following proposition, the Lazy Tracking Forecaster (Algorithm 2–5) minimizes the regret with respect to the new baseline of (2.7.19). The proof (in Section 2.8) derives from existing results on the Fixed-share algorithm of [70].

Proposition 2.13 (Expected Regret of Lazy Tracking Forecaster)**.** *Let the length of all phases be* $|\tau| = \lceil T^{1/3} \rceil$. *Suppose that Assumptions 2.1 and 2.2 hold. If the agent follows the Lazy Tracking Forecaster algorithm with parameters* $\eta = T^{-2/3}$ *and* $\alpha = K_0/(\lceil T/\lceil T^{1/3}\rceil\rceil - 1)$, *then the following cumulative*

regret bound holds for large enough T:

$$\mathcal{B}_T(K_0) - \sum_{t=0}^{T-1} \mathbb{E}\big[r_t(\mathbf{s}_t, \mathbf{a}_t)\big] \leq |S| \log(|A|)(K_0 + 1)T^{2/3}$$
$$+ 2K_0 \log\left(T^{2/3}/K_0\right) T^{2/3} + \frac{1}{2}T^{2/3} + (2e\gamma)T^{2/3}.$$

Remark 19. Observe that this bound is tighter than the bound of Proposition 2.3.

We now prove Theorem 2.12.

Proof. Theorem 2.12 Consider the Tracking FPL algorithm (Algorithm 2–4) and the Lazy Tracking Forecaster (Algorithm 2–5) with their parameters α set equal. Let all phases for the Lazy Tracking Forecaster algorithm have fixed length τ. Let M denote the number of phases for the Tracking FPL algorithm. By their definition, at every given step t and with probability α, the two algorithms follow a policy μ chosen uniformly at random. Hence, the difference in their expected cumulative reward is $1 - \alpha$ times the same difference when the parameters α are set to 0. We will proceed to bound this latter quantity.

Observe that the Tracking FPL algorithm with $\alpha = 0$ is simply the Lazy FPL algorithm. The Lazy Tracking Forecaster with $\alpha = 0$ is just an exponentially weighted average forecaster [34] with one phase as the fundamental time step. Let $\mathbf{a}_t$ and $\mathbf{b}_t$ denote the actions generated by the Lazy Tracking Forecaster and the Tracking FPL algorithms, respectively. By setting the argument K_0 to the baseline $\mathcal{B}_T$ to 0, we shall derive the following bounds on

their respective cumulative regrets:

$$\Omega\left(\sqrt{T\log(|A|)}\right) \leq \mathcal{B}_T(0) - \sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{a}_t)\big] \leq \frac{|S|\log(|A|)}{\eta} + \frac{\eta\,\lceil T/|\tau|\rceil\;|\tau|^2}{2}, \tag{2.7.22}$$

$$\Omega\left(\sqrt{T\log(|A|)}\right) \leq \mathcal{B}_T(0) - \sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{b}_t)\big] \leq \frac{4}{3}\big(2e\gamma + 2\,|A| + 4e + 1 + 2(|S|+3)\,|A|^2\,\gamma\log(T)\big)\;T^{3/4+\epsilon}. \tag{2.7.23}$$

The upper bound of (2.7.22) follows from an argument similar to [34, Theorem 2.1]; that of (2.7.23) follows from Proposition 2.3. Both lower bounds are due to instances where the regret is no less than of the order of $\Omega(T^{1/2})$ [34, Theorem 3.7]. The above bounds combine to give

$$\begin{aligned}&\left|\sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{a}_t)\big] - \sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{b}_t)\big]\right| \\ &\leq \frac{|S|\log(|A|)}{\eta} + \frac{\eta\,\lceil T/|\tau|\rceil\;|\tau|^2}{2} \\ &\quad + \frac{4}{3}\big(2e\gamma + 2\,|A| + 4e + 1 + 2(|S|+3)\,|A|^2\,\gamma\log(T)\big)\;T^{3/4+\epsilon},\end{aligned} \tag{2.7.24}$$

since the lower bounds are superseded by the upper bounds for all phase-partitions consistent with the assumptions of Proposition 2.3. By substituting the values $|\tau| = \lceil T^{1/3}\rceil$ and $\eta = T^{-2/3}$ and compounding the bound of (2.7.24) to that of Proposition 2.13, we obtain the following bound:

$$\begin{aligned}&\mathcal{B}_T(K_0) - \sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{a}_t)\big] \leq |S|\log(|A|)(K_0+2)T^{2/3} + 2K_0\log\left(T^{2/3}/K_0\right)T^{2/3} \\ &+ T^{2/3} + (2e\gamma)T^{2/3} + \frac{4}{3}\big(2e\gamma + 2\,|A| + 4e + 1 + 2(|S|+3)\,|A|^2\,\gamma\log(T)\big)\;T^{3/4+\epsilon}.\end{aligned} \tag{2.7.25}$$

At last, the claimed result follows by an argument similar to the proof of Theorem 2.2. ■

Remark 20. The bound on expected cumulative regret of the Tracking FPL algorithm (cf. (2.7.25)) is of the same order as that afforded by the Lazy FPL algorithm (cf. Proposition 2.3). This indicates that the critical factor in the convergence of the algorithm is its "laziness."

2.8 Proofs

Proof. Lemma 2.4 By introducing indicator functions, we obtain

$$\mathbb{E}\big[r_j(\mathbf{s}_t, \mathbf{a}_t)\big] = \mathbb{E}\left\{ \sum_{(s,a)\in S\times A} r_j(s,a)\mathbf{1}_{[(\mathbf{s}_t,\mathbf{a}_t)=(s,a)]} \right\} \tag{2.8.26}$$

$$= \sum_{(s,a)\in S\times A} r_j(s,a)\ \mathbb{E}\ \mathbf{1}_{[(\mathbf{s}_t,\mathbf{a}_t)=(s,a)]} \tag{2.8.27}$$

$$= \sum_{(s,a)\in S\times A} r_j(s,a)\Pr\big((\mathbf{s}_t,\mathbf{a}_t)=(s,a)\big), \tag{2.8.28}$$

where (2.8.26) follows by definition and the use of indicator functions, (2.8.27) is justified by Assumption 2.2, and (2.8.28) follows again by definition. ∎

Proof. Lemma 2.1 By Lemma 2.4, for a stationary policy $\mu \in \Sigma$, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{a}_t)\big] = \frac{1}{T}\sum_{t=0}^{T-1}\langle r_t, d_t(\mu; s_0)\rangle.$$

By Assumption 2.1, and the summability of the sequence $e^{1-t/\gamma}$, we have

$$\left|\frac{1}{T}\sum_{t=0}^{T-1}\langle r_t, d_t(\mu;s_0)\rangle - \frac{1}{T}\sum_{t=0}^{T-1}\langle r_t, \pi(\mu)\rangle\right| \le \frac{1}{T}\sum_{t=0}^{T-1} 2e^{1-t/\gamma} = 2e\gamma/T.$$

By definition, we have

$$\frac{1}{T}\sum_{t=0}^{T-1}\langle r_t, \pi(\mu)\rangle = \langle \widehat{r}_T, \pi(\mu)\rangle.$$

Putting these pieces together, we obtain

$$\left|\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[r_t(\mathbf{s}_t,\mathbf{a}_t)\big] - \langle \widehat{r}_T, \pi(\mu)\rangle\right| \le 2e\gamma/T.$$

By a similar argument, we have

$$\left| \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\widehat{r}_T(\mathbf{s}_t, \mathbf{a}_t)\big] - \langle \widehat{r}_T, \pi(\mu) \rangle \right| \leq 2e\gamma/T.$$

The two claims follow from taking the supremum over the set of stationary policies. ■

Proof. Lemma 2.6 For non-negative integers n and m such that $n \geq m$, algebraic manipulation yields

$$\begin{aligned}
\left\| \frac{1}{n} \sum_{j=0}^{n-1} r_j - \frac{1}{\ell} \sum_{j=0}^{\ell-1} r_j \right\|_\infty &= \left\| \frac{1}{n} \sum_{j=0}^{\ell-1} r_j + \frac{1}{n} \sum_{j=\ell}^{n-1} r_j - \frac{1}{\ell} \sum_{j=0}^{\ell-1} r_j \right\|_\infty \\
&\leq \frac{1}{n} \left\| \sum_{j=\ell}^{n-1} r_j \right\|_\infty + \left| \frac{n-\ell}{n} \right| \left\| \frac{1}{\ell} \sum_{j=0}^{\ell-1} r_j \right\|_\infty \\
&\leq 2 \, \frac{n-\ell}{n}
\end{aligned}$$

where the last inequality follows from the fact that $r_0, r_1, \ldots$ are bounded by 1. ■

Proof. Lemma 2.7 Let $t' \in \tau_{m+1}$ and $t \in \tau_m$. By the assumption of Theorem 2.2, the cumulative distribution functions of $\mathbf{n}_{t'}(a)$ and $\mathbf{n}_t(a)$ satisfy the following bounds for all $z, z' \in \mathbb{R}$:

$$\begin{aligned}
\left| F_{\mathbf{n}_{t'}(a)}(z) - F_{\mathbf{n}_t(a)}(z) \right| &\leq \frac{\zeta_{m+1} - \zeta_m}{2\zeta_{m+1}}, \\
\left| F_{\mathbf{n}_{t'}(a)}(z) - F_{\mathbf{n}_{t'}(a)}(z') \right| &\leq \frac{\zeta_{m+1}}{2} |z - z'|.
\end{aligned}$$

Likewise, for $a, a' \in A$, we have:

$$\left| F_{\mathbf{n}_{t'}(a) - \mathbf{n}_{t'}(a')}(z) - F_{\mathbf{n}_t(a) - \mathbf{n}_t(a')}(z) \right| \leq \frac{\zeta_{m+1} - \zeta_m}{2\zeta_{m+1}}, \tag{2.8.29}$$

$$\left| F_{\mathbf{n}_{t'}(a) - \mathbf{n}_{t'}(a')}(z) - F_{\mathbf{n}_{t'}(a) - \mathbf{n}_{t'}(a')}(z') \right| \leq \frac{\zeta_{m+1}}{2} |z - z'|. \tag{2.8.30}$$

By Lemma 2.6, we have

$$\left\|\widehat{r}_{\tau_{0:m+1}} - \widehat{r}_{\tau_{0:m}}\right\|_\infty \leq 2\left|\tau_{m+1}\right| / \left|\tau_{0:m+1}\right|. \tag{2.8.31}$$

Observe that the linear programs (cf. (2.4.6)) at the m-th and $m+1$-th phases differ only in their right-hand constraint vectors, whose difference is bounded by (2.8.31). It follows by [110, Theorem 1.1] that the optimal values λ_m and λ_{m+1} satisfy

$$\left|\lambda_{m+1} - \lambda_m\right| \leq \left\|\widehat{r}_{\tau_{0:m+1}} - \widehat{r}_{\tau_{0:m}}\right\|_\infty.$$

Likewise, by [113, Corollary 3.1], the solutions h_{m+1} and h_m differ as follows:

$$\begin{aligned}\left\|h_{m+1} - h_m\right\|_\infty &\leq (|S|+1)\left\|\widehat{r}_{\tau_{0:m+1}} - \widehat{r}_{\tau_{0:m}}\right\|_\infty &(2.8.32)\\ &\leq 2(|S|+1)\left|\tau_{m+1}\right| / \left|\tau_{0:m+1}\right|. &(2.8.33)\end{aligned}$$

Starting from the definition of Algorithm 2–1, for every $s \in S$, $a \in A$ and $m = 0, 1, \ldots,$

$$\begin{aligned}&\sigma_{m+1}(a; s)\\ &\triangleq \Pr(\mathbf{a}_{t'} = a \mid \mathbf{s}_{t'} = s)\\ &= \Pr\Big(\widehat{r}_{\tau_{0:m+1}}(s,a) + \sum_{s'\in S} P(s' \mid s_t, a) h_m(s') + \mathbf{n}_{t'}(a) > &(2.8.34)\\ &\widehat{r}_{\tau_{0:m+1}}(s,a') + \sum_{s'\in S} P(s' \mid s_t, a') h_m(s') \big[h_{m+1}(\mathbf{s}_{t+1})\big] + \mathbf{n}_{t'}(a'), \quad \text{for all } a' \neq a\Big)\\ &= \prod_{a'\neq a} \Pr\Big(\mathbf{n}_{t'}(a) - \mathbf{n}_{t'}(a') > \widehat{r}_{\tau_{0:m+1}}(s,a') - \widehat{r}_{\tau_{0:m+1}}(s,a) &(2.8.35)\\ &+ \sum_{s'\in S} P(s' \mid s_t, a') h_m(s') - \sum_{s'\in S} P(s' \mid s_t, a) h_m(s')\Big),\end{aligned}$$

where the probability measure is over the randomization $\mathbf{n}_{t'}$, whereas the expectation is over the transition probabilities of the MDP. (2.8.34) is due to the definition of Algorithm 2–1 ((2.4.7)). (2.8.35) is obtained by independence of

the random variables $\mathbf{n}_{t'}(a)$, for $a \in A$. By comparing Equations (2.8.35) applied to σ_{m+1} and σ_m, and using Equations (2.8.31), (2.8.33), (2.8.29) and (2.8.30), we obtain

$$\begin{aligned}&\|\sigma_{m+1}(\cdot;s) - \sigma_m(\cdot;s)\|_\infty \\ &\leq (|A|-1)\left(\frac{\zeta_{m+1}}{2}(4+2(|S|+1))\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{2\zeta_{m+1}}\right),\end{aligned}$$

for all $s \in S$. For the 1-norm, we have

$$\|\sigma_{m+1}(\cdot;s) - \sigma_m(\cdot;s)\|_1 \leq (|S|+3)\,|A|\,(|A|-1)\left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) \tag{2.8.36}$$

for all $s \in S$.

For the second part of the lemma, let $\mathbf{P}_\mu$ be the transition matrix associated with a stationary policy $\mu : S \to A$. The element of $\mathbf{P}_\mu$ in row (s', a') and column (s, a) is the probability that the next state-action pair is (s', a') if the current one is (s, a) and policy μ is followed. Let $d \in \Delta(S \times A)$ be a probability vector specifying the initial state-action pair $(s_0, \mu(s_0))$. We first show by induction that

$$\left\|\mathbf{P}^j_{\sigma_{m+1}} d - \mathbf{P}^j_{\sigma_m} d\right\|_1 \leq j(|S|+3)\,|A|^2\left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) \tag{2.8.37}$$

for $j = 1, 2, \ldots$. Let $e_1, \ldots, e_{|S\times A|}$ denote the elementary vectors in $\mathbb{R}^{|S\times A|}$. For the base case $j = 1$, we have

$$
\begin{aligned}
&\left\|\mathbf{P}_{\sigma_{m+1}} d - \mathbf{P}_{\sigma_m} d\right\|_1 \\
&\leq \max_{n=1,\ldots,|S\times A|} \left\|\mathbf{P}_{\sigma_{m+1}} e_n - \mathbf{P}_{\sigma_m} e_n\right\|_1 \\
&= \max_{(s,a)\in S\times A} \Big| \sum_{(s',a')\in S\times A} P(s' \mid s,a)\sigma_{m+1}(a';s') - P(s' \mid s,a)\sigma_m(a';s') \Big| \\
&= \max_{(s,a)\in S\times A} \Big| \sum_{s'\in S} P(s' \mid s,a) \sum_{a'} \sigma_{m+1}(a';s') - \sigma_m(a';s') \Big| \\
&\leq \max_{s'\in S} \Big| \sum_{a'\in A} \sigma_{m+1}(a';s') - \sigma_m(a';s') \Big| \\
&= \max_{s'\in S} \left\|\sigma_{m+1}(\cdot;s') - \sigma_m(\cdot;s')\right\|_1 \\
&\leq (|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right),
\end{aligned}
$$

where the last inequality follows from (2.8.36). Next, suppose that for some j, we have

$$
\left\|\mathbf{P}^j_{\sigma_{m+1}} d - \mathbf{P}^j_{\sigma_m} d\right\|_1 = j(|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right).
$$

By the triangle inequality and the same argument as the base case, we obtain

$$
\begin{aligned}
&\left\|\mathbf{P}^{j+1}_{\sigma_{m+1}} d - \mathbf{P}^{j+1}_{\sigma_m} d\right\|_1 \\
&\leq \left\|\mathbf{P}_{\sigma_{m+1}}\mathbf{P}^j_{\sigma_{m+1}} d - \mathbf{P}_{\sigma_m}\mathbf{P}^j_{\sigma_{m+1}} d\right\|_1 + \left\|\mathbf{P}_{\sigma_m}\mathbf{P}^j_{\sigma_{m+1}} d - \mathbf{P}_{\sigma_m}\mathbf{P}^j_{\sigma_m} d\right\|_1 \\
&= (|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) \\
&+ j(|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right),
\end{aligned}
$$

which establishes (2.8.37). At last, by the triangle inequality, (2.8.37) and Assumption 2.1, it follows that for an every positive integer g, and every

initial state s_0 and corresponding distribution d,

$$\begin{aligned}\left\|\pi(\sigma_{m+1}) - \pi(\sigma_m)\right\|_1 &= \left\|\mathbf{P}^g_{\sigma_{m+1}} d - \mathbf{P}^g_{\sigma_m} d\right\|_1 \\ &\quad + \left\|\pi(\sigma_{m+1}) - \mathbf{P}^g_{\sigma_{m+1}} d\right\|_1 + \left\|\pi(\sigma_m) - \mathbf{P}^g_{\sigma_m} d\right\|_1 \\ &\leq g(|S|+3)\,|A|^2 \left(\zeta_{m+1}\frac{|\tau_{m+1}|}{|\tau_{0:m+1}|} + \frac{\zeta_{m+1}-\zeta_m}{\zeta_{m+1}}\right) + 4e^{1-g/\gamma}.\end{aligned}$$

■

Proof. Lemma 2.8 Let $t \in \tau_m$; let action $\mathbf{a}_t^+$ follow policy σ_m^+, and action $\mathbf{a}_t$ follow σ_m. Recall that the action $\mathbf{a}_t^+$ is an optimal action against an MDP with fixed reward function $\widehat{r}_{\tau_{0:m-1}}$. Let us consider the following random variables, for $(s,a) \in S \times A$,

$$\widehat{r}_{\tau_{0:m-1}}(s,a) + \sum_{s' \in S} P(s' \mid s_t, a) h_m(s') + \mathbf{n}_t(a). \tag{2.8.38}$$

For ease of notation, we define, for $(s,a) \in S \times A$,

$$\xi_m(s,a) = \widehat{r}_{\tau_{0:m-1}}(s,a) + \sum_{s' \in S} P(s' \mid s_t, a) h_m(s').$$

Observe that $\xi_m(s, \mathbf{a}_t^+) \geq \xi_m(s,a)$ for every $a \neq \mathbf{a}_t^+$ by definition. Let Ψ denote the interval over which the supports of the random variables $\mathbf{n}_t(\mathbf{a}_t^+) + \xi_m(s, \mathbf{a}_t^+)$ and $\mathbf{n}_t(a) + \xi_m(s,a)$ overlap. This interval Ψ has length $2/\zeta_m - \big(\xi_m(s, \mathbf{a}_t^+) - \xi_m(s,a)\big)$. Combining this fact with the fact that $\mathbf{n}_t(\mathbf{a}_t^+)$ and $\mathbf{n}_t(a)$ are independent and have uniform distributions specified by the assumption

of Theorem 2.2, we have, for every $s \in S$,

$$\begin{aligned}
\Pr(\mathbf{a}_t = a \mid \mathbf{s}_t = s) &= \Pr\Big(\mathbf{n}_t(a) + \xi_m(s,a) > \mathbf{n}_t(\mathbf{a}_t^+) + \xi_m(s,\mathbf{a}_t^+)\Big) \\
&= \frac{1}{2}\Pr\big(\mathbf{n}_t(\mathbf{a}_t^+) + \xi_m(s,\mathbf{a}_t^+) \in \Psi, \\
&\qquad\qquad \mathbf{n}_t(a) + \xi_m(s,a) \in \Psi\big) \\
&\leq \begin{cases} \frac{\zeta_m}{4}\Big(2/\zeta_m - \big(\xi_m(s,\mathbf{a}_t^+) - \xi_m(s,a)\big)\Big)^2, \\ \qquad \text{if } \xi_m(s,\mathbf{a}_t^+) - \xi_m(s,a) \leq 2/\zeta_m, \\ 0, \quad \text{otherwise.} \end{cases} \qquad (2.8.39)
\end{aligned}$$

Observe next that

$$\begin{aligned}
&\big|\langle R_{\tau_{0:m-1}}, \pi(\sigma_m) - \pi(\sigma_m^+)\rangle\big| \\
&= |\tau_{0:m-1}|\,\big|\langle \widehat{r}_{\tau_{0:m-1}}, \pi(\sigma_m)\rangle - \langle \widehat{r}_{\tau_{0:m-1}}, \pi(\sigma_m^+)\rangle\big| \\
&\leq |\tau_{0:m-1}| \max_{s\in S} \sum_{a \neq \mathbf{a}_t^+} \big(\xi_m(s,\mathbf{a}_t^+) - \xi_m(s,a)\big)\Pr(\mathbf{a}_t = a \mid \mathbf{s}_t = s) \\
&\leq |\tau_{0:m-1}|\,(|A|-1)(2/\zeta_m)\frac{\zeta_m}{4}(2/\zeta_m)^2 \\
&\leq 2\,|A|\,\frac{|\tau_{0:m-1}|}{\zeta_m^2},
\end{aligned}$$

where the second to last inequality follows by (2.8.39). ∎

Proof. Corollary 2.9 (Outline) The desired result follows an approach similar to Proposition 2.3 and Theorem 2.2. First, let ρ_m denote the policy induced by the (ϵ, δ)-approximation algorithm for the m-th phase. Let $\mathbf{P}_{\rho_m}$ and $\pi(\rho_m)$ denote the transition probability matrix and the stationary distribution associated with ρ_m; and likewise for σ_m. Observe that, by Definition 2.3,

$$\|\mathbf{P}_{\rho_m} - \mathbf{P}_{\sigma_m}\|_\infty \leq \max_{s\in S} \|\rho_m(\cdot; s) - \sigma_m(\cdot; s)\|_1 \leq \epsilon + \delta.$$

By [115, Section 6], the stationary distributions $\pi(\rho_m)$ and $\pi(\sigma_m)$ satisfy

$$\|\pi(\rho_m) - \pi(\sigma_m)\|_1 \leq \|\mathbf{Z}_{\sigma_m}\|_\infty \|\mathbf{P}_{\rho_m} - \mathbf{P}_{\sigma_m}\|_\infty \leq \sup_{\sigma\in\Sigma} \|\mathbf{Z}_\sigma\|_\infty (\epsilon + \delta).$$

Hence, we have

$$\begin{aligned}\sum_{t=0}^{T-1} \mathbb{E}\big[r_t(\mathbf{s}_t, \mathbf{a}_t)\big] &\geq \sum_{m=0}^{M-1} \big(\langle R_{\tau_m}, \pi(\rho_m)\rangle - 2e\gamma\big) \\ &\geq \sum_{m=0}^{M-1} \big(\langle R_{\tau_m}, \pi(\sigma_m)\rangle - 2e\gamma\big) - \sup_{\sigma\in\Sigma} \|\mathbf{Z}_\sigma\|_\infty (\epsilon + \delta)T,\end{aligned}$$

where the first inequality is justified by the same argument as Step 0 of the proof of Proposition 2.3. This bound is similar to (2.4.10) of the proof of Proposition 2.3 with one additional term. The claimed result follows by arguments similar to the proofs of Proposition 2.3 and Theorem 2.2. ■

Proof. Corollary 2.11 (Outline) By introducing exploration phases as described above, we ensure that $\mathbf{z}_t$ is an unbiased estimator for r_t. Indeed, observe that for every $(s, a) \in S \times A$ and t large enough,

$$\begin{aligned}\Pr\big((\mathbf{s}_t, \mathbf{a}_t) = (s, a) \mid s_0\big) &= \Pr(\mathbf{s}_t = s \mid s_0) \Pr(\mathbf{a}_t = a \mid \mathbf{s}_t = s) \\ &\geq \Pr(\mathbf{s}_t = s \mid s_0)\, \phi_m / |A|\,.\end{aligned}$$

Next, observe that the ergodicity assumption (Assumption 2.1) guarantees that there exists an $\epsilon > 0$ such that for every $s \in S$ and large enough t,

$$\Pr(\mathbf{s}_t = s \mid s_0) > \epsilon.$$

Moreover, we have $\phi_m > 0$ by assumption. Hence, if the opponent is oblivious and for large enough t, we obtain

$$\mathbb{E}\big[\mathbf{z}_t(s, a)\big] = r_t(s, a), \quad \text{for all } (s, a) \in S \times A,$$

and in turn,

$$\mathbb{E}\left[\frac{1}{t}\sum_{j=0}^{t-1}\mathbf{z}_j(s,a)\right] = \widehat{r}_t(s,a), \quad \text{for all } (s,a) \in S \times A.$$

Therefore, we conclude by Lemma 2.5 that the policy induced by the Exploratory FPL algorithm is still optimal against $\widehat{r}_{\tau_{0:m-1}} + \mathbf{n}_t$. All the remaining steps of the proof of Proposition 2.3 hold unchanged, if we exclude the exploration phases. Since these phases incur an overhead of the order of $O(M)$ by (2.6.18), we obtain a bound analogous to (2.4.8). Finally, the claim follows by the same argument as the proof of Theorem 2.2. ∎

Proof. Proposition 2.13 For ease of notation, we write $M = \lceil T/\left|\tau\right| \rceil$ to denote the number of phases of the Lazy Tracking Forecaster algorithm. Observe that Lazy Tracking Forecaster is the same as the tracking forecaster of [70], with the exception that the fundamental time step is an entire phase in our new setting. Our claim follows from [34, Theorem 5.2 and Corollary 5.1] by adjusting the time scale.

The crucial observation is that at Step 2 of Algorithm 2–5, the weights are not updated according to the aggregate reward obtained by following policy each μ over each phase τ_m, but according to the expected reward in the stationary state-action distribution of each policy μ in each phase τ_m. Consequently, [34, Theorem 5.2] gives the bound

$$\mathcal{B}_T(K_0) - \sum_{m=0}^{M-1}\langle R_{\tau_m}, \pi(\mathbf{q}_m)\rangle \leq \frac{|S|\log(|A|)}{\eta}(K_0+1) + \frac{1}{\eta}(M-1)H\left(\frac{K_0}{M-1}\right) + \frac{\eta\, M\ |\tau|^2}{2}.$$

The required result follows by observing that we can approximate

$$\sum_{j\in\tau_m}\mathbb{E}\big[r_j(\mathbf{s}_j,\mathbf{a}_j) \mid s^-\big],$$

where the actions $\mathbf{a}_j$ follow policy $\mathbf{q}_m$ and s^- is the state of the MDP at the beginning of phase τ_m, by

$$\langle R_{\tau_m}, \pi(\mathbf{q}_m)\rangle \triangleq \sum_{j\in\tau_m} \langle r_j, \pi(\mathbf{q}_m)\rangle.$$

As shown in Step 0 of the proof of Proposition 2.3, we have

$$\left|\sum_{j\in\tau_m} \mathbb{E}\big[r_j(\mathbf{s}_j, \mathbf{a}_j) \mid s^-\big] - \langle R_{\tau_m}, \pi(\mathbf{q}_m)\rangle\right| \leq 2e\gamma,$$

for $m = 0, \ldots, M-1$, which accounts for the term $2e\gamma M$. Finally, the claim follows by substituting $|\tau| = \lceil T^{1/3}\rceil$ and $\eta = T^{-2/3}$, and observing that for $0 \leq p < 1/2$, we have

$$H(p) < 2p\log(1/p),$$

so that, for large enough T,

$$H\left(\frac{K_0}{\lceil T/|\tau|\rceil - 1}\right) < 2\frac{K_0}{\lceil T/|\tau|\rceil - 1}\log\left(\frac{\lceil T/|\tau|\rceil - 1}{K_0}\right).$$

■

CHAPTER 3
Regret Minimization in Non-stationary Markov Decision Processes

3.1 Introduction

In this chapter, we consider decision-making problems in Markov decision processes where *both* the rewards and the transition probabilities vary in an arbitrary (*e.g.*, nonstationary) fashion. We propose online learning algorithms and provide guarantees on their performance evaluated in retrospect against alternative policies. Unlike previous works, the guarantees depend critically on the variability of the uncertainty in the transition probabilities, but hold regardless of arbitrary changes in rewards and transition probabilities over time. First, we use an approach based on robust dynamic programming and extend it to observation of the reward is limited to the actual state-action trajectory. Next, we present a computationally efficient simulation-based Q-learning style algorithm that requires neither prior knowledge nor estimation of the transition probabilities. We show both probabilistic performance guarantees and deterministic guarantees on the expected performance.

We consider a problem where an agent controls a system subject to uncertainty. The system under control is modelled as a Markov decision process (MDP). We consider a generalized version of the MDP model that captures an additional feature of real-life problems: the rewards and transition probabilities may change over time in an arbitrary and nonstationary manner (*e.g.*, under the influence of an adversary or Nature). This generalized version is reminiscent of stochastic games [116]. However, in stochastic games, one usually assumes that there is an adversary whose utility is well-defined. In our setup, as in [44], [91], and more generally in the online learning setting [34],

we do not assume that such an adversary exists, but rather that the reward and transition processes are modulated[1] by arbitrary individual sequences.

Uncertainty can be intrinsic and unavoidable to the system (probabilistic transitions, uncertainty principles), or may arise from measurements. In many decision-making problems, uncertainty exists in the rewards and transition probabilities (cf. [101] and references therein). When this uncertainty follows a stochastic model ([10, 30]), sampling can give estimates on the parameters of the transition model, but some residual uncertainty always remains as a result of limited samples. Under these circumstances, if it is imperative to take precautions against the worst possible occurrence, then a standard approach is to handle this uncertainty through *robust optimization*. This approach typically assumes that the uncertainty has a *fixed*—albeit unknown—realization and constructs a policy that performs well in the worst case. Since the uncertainty evolves in a nonstationary fashion in our setting, the standard robust approach does not offer a satisfying solution. In particular, it only guarantees optimal performance against the *worst* realization of the *environment*. It does not promise anything about the relative performance compared to the *best* alternative *policy* in *hindsight*. The goal of this chapter is to address this *relative* performance.

In this chapter, we distinguish between two notions of uncertainty: arbitrary variations in the *reward function*, and arbitrary, but constrained, variations in the *transition probabilities*. These notions have previously been studied individually. Online learning yields solutions that are robust against arbitrary

[1] This terminology refers to random processes whose parameters change according to another process; *e.g.*, a Markov modulated Bernoulli process is a Bernoulli process whose success probability changes according to a Markov chain [102].

variations in the reward functions when the transition probabilities are fixed ([44] and Chapter 2). Robust dynamic programming has been used to control MDPs where the transition probabilities may vary arbitrarily, but where the reward functions may not [101]. In this chapter, we address both uncertainties simultaneously.

It is important to note that, in contrast to most online learning settings, the average regret (or performance-loss) does *not* vanish with time when the transition probabilities can change arbitrarily over time. The regret actually grows linearly with time for some sequences of transition probabilities [44]. Therefore, the setting of interest is where the uncertainty in the transition probabilities is limited, whereas the uncertainty in the rewards is not. Our results will quantify the extent to which the regret may grow as the transition probabilities are given more leeway to change.

3.1.1 Related Works

Special cases of our model have been studied before. MDPs with fixed, but unknown, reward functions and transition probabilities have been solved using robust dynamic programming with finite- and infinite-horizon objectives ([101, 127]). In general, the models for nonstationary settings limit the nonstationarity to *either* the rewards *or* the transition probabilities. The problem of nonstationary MDPs where the reward and transition functions are known a priori has been studied in [63]

MDPs where the reward functions may change arbitrarily, but the transition function is fixed, have been considered using regret-minimizing approaches. In ([91, 44] and Chapter 2), various notions of regret are introduced, and solutions are proposed based on approachability theory, weighted-experts forecasters, and perturbed dynamic programming. However, as we will show

with a counterexample, some of the regret-minimizing strategies fail to obtain acceptable performance when the transitions probabilities are allowed to change arbitrarily. These works generalize the standard online learning problem, which does not include a notion of state. Such a model is suitable when the transitions are so arbitrary as to make the notion of states meaningless. This model has been extensively studied in the context of repeated games [64], prediction with expert advice [87], the adversarial multi-armed bandit [10], and the online shortest path problem [72].

In MDPs with arbitrarily varying transition functions, but a fixed reward function, the reference [101] proposes a solution using robust dynamic programming. However, this method of robustifying only against variations in the transition model does not guarantee good performance when reward may also change arbitrarily.

To the best of our knowledge, our work is the first to propose a solution that handles nonstationary uncertainty in both rewards and transitions in the online setting. In order to do this, we employ a combination of methods from robust dynamic programming and regret-minimization. Although our algorithms are similar to those of Chapter 2, the analysis differs greatly because our model assumes that the uncertainty extends to both rewards and transitions. For instance, it is not possible in our setting to achieve the same optimal values as in [101] and Chapter 2, even approximately. We show methods to achieve approximate optimality when the variability of transition uncertainty is restricted.

3.1.2 Stochastic Games

Nonstationary MDPs can also be presented from a game theoretic perspective, where the rewards and state-transitions are controled by the actions of the agent and an opponent. Consider a one-sided infinite-horizon stochastic

game [116] between two players. By "one-sided," we mean that the opponent's preference relation is left unspecified. This model comprises elements reminiscent of both MDPs and repeated games:

1. a finite set of states S, as in an MDP,
2. a finite set A of actions available to the agent,
3. a set B of actions at the disposal of the opponent,
4. a probability kernel P governing state transitions, *i.e.*, $P(s' \mid s, a, b)$ defines the probability that the next state is s' if the current one is s and the agent and opponent respectively play actions a and b.
5. a function $r : S \times A \times B \to [0, 1]$ that specifies the reward or payoff $r(s, a, b)$ to the agent, based on the current state s and the agent and opponent's current actions—a and b, respectively.

If the opponent follows a stationary policy, then stochastic game reduces to a standard MDP from the perspective of the agent. If there is only one state in S, then the stochastic game reduces to the repeated game setting of online learning. Under the standard game theoretic assumption of rationality, solution concepts exist for stochastic games. These settings have solutions that can be computed efficiently [20, 34, 50, respectively].

Mannor and Shimking [91] consider stochastic games where the opponent is possibly non-rational. They introduce a notion of regret based on the joint *frequency* of state traversed and opponent actions, and show the existence of a no-regret policy. However, computing this policy is a challenging task. For example, one possibility is to first identify a target set, and then approach this set by solving a standard MDP [118]. However, computing each value of the target set boils down to the difficult task of computing the value of a zero-sum stochastic game [109].

3.1.3 Hardness

Robustness against arbitrary changes in the transition function is fundamentally different from robustness against arbitrary rewards. The known regret-minimizing techniques do not apply directly because some sequences of transition functions may prevent ergodicity or create periodicity, which causes non-vanishing average regret Chapter 2. With the regret notion that we will adopt, arbitrary transition functions $P_1, P_2, \ldots$ present a significant challenge. Whereas it is possible to obtain asymptotically vanishing average regret when only the reward functions $r_1, r_2, \ldots$ are arbitrary, if both the transition probabilities and rewards are arbitrary, as in our model, it is NP-hard to compute a policy that comes close to the best stationary policy [44]. Hence, without an assumption that constrains the magnitude of changes in the sequence $\{P_t\}$, it is unknown whether it is possible to obtain a tractable solution with a non-trivial regret guarantee. Since vanishing regret can not be achieved, we characterize the dependence of the regret on the variability of transition uncertainty.

3.1.4 Contributions and Outline

In Section 3.2, we formulate a model for Markov decision processes with arbitrarily modulated reward and transition functions. This model generalizes existing models of time-varying MDPs; it also encompasses the general setting of stochastic games, where both the agent and opponent affect the transitions, but where the opponent may play an arbitrary sequence of actions. We show by an example in Section 3.3 that standard online learning algorithms are not robust against small but arbitrary changes in the transition function.

The following are our main contributions. In Section 3.4, we propose a solution that uses online learning to adapt to time-varying reward functions and

uses robust dynamic programming to deal with the range of possible transition models. We robustify a standard online dynamic programming algorithm by solving an additional optimization over possible transition functions. In Section 3.5, we extend the solution to the case where the agent's observation of rewards is limited to its state-action trajectory. In Section 3.6, we employ a simulation-based method that eliminates the need to estimate the uncertainty set of the transition probabilities. We take an approach that combines an online learning solution for reward-uncertainty and a simulation-based (Q-learning) solution that deals implicitly with the transition uncertainty. The resulting algorithm has the benefit of being computationally efficient, and requires neither the knowledge nor the estimation of the transition probabilities. We give both a probabilistic bound on the regret and a bound on its expected value. These bounds hold for finite time horizons and quantify the dependence on the range of uncertainty in the transition probabilities. We give comparisons and pose open problems in Section 3.7.

3.2 Setting

A Markov decision process is a standard sequential decision-making problem ([108, 20]). At each time step $t \in \{1, 2, \ldots\}$, an agent takes an action a_t from a finite set A. Starting from a fixed state s_1, the agent occupies at each time step t a state s_t belonging to a finite set S. The sets S and A are known to the agent. Given the current state-action pair (s_t, a_t), the probability that the next state s_{t+1} will be some state $s' \in S$ is $P_t(s' \mid s_t, a_t)$. The function P_t that determines these transition probabilities is the *transition function*. At every time step, the agent also receives a reward $r_t(s_t, a_t)$ depending on the reward function r_t, and its state s_t and action a_t. The agent's objective is to maximize a function of the rewards.

In this chapter, we consider a general version of MDPs where the reward functions r_t and transition functions P_t are a priori *unknown* to the agent. Furthermore, we do not assume—as is usually the case—that the sequences $r_1, r_2, \ldots$ and $P_1, P_2, \ldots$ are stationary. We treat these as individual sequences that change arbitrarily (*e.g.*, in a nonstationary manner) over time. This model generalizes the models of [101] and Chapter 2. In turn, we look for solutions and guarantees that hold universally against *every* pair of individual sequences $r_1, r_2, \ldots$ and $P_1, P_2, \ldots$.

Remark 21. Another possible model for our setting is a one-sided stochastic game with two players [116]. The opponent's action b_t at time t determines r_t and P_t, but the actions b_t are chosen arbitrarily—without regard to optimizing a fixed payoff function.

We assume that every reward function $r_t : S \times A \to \mathbb{R}$ takes values in the interval $[0, 1]$. Let C denote a finite set of indices. Let each $P^c : S \times S \times A \to [0, 1]$, for $c \in C$, denote a fixed transition function, where $P^c(s' \mid s, a)$ is the probability that the next state is s' given that the current state is s and that the agent takes action a. The transition functions $\{P^c\}_{c \in C}$ form a basis for the sequence of transition functions $P_1, P_2, \ldots$ as follows:

$$P_t(s' \mid s, a) = \sum_{c \in C} \delta_t(c) P^c(s' \mid s, a), \quad t = 1, 2, \ldots,$$

where $\delta_1, \delta_2, \ldots$ is the *modulating sequence*, and each element δ_t lies in an uncertainty set $\mathcal{D}$ that is a subset of the simplex of C—denoted $\Delta(C)$. Hence, $\mathcal{D}$ is a set of probability distributions over the basic transition functions $\{P^c\}_{c \in C}$. For a fixed $\delta \in \Delta(C)$, we also define the transition function P^δ as follows: for $(s, a) \in S \times A$ and $s' \in S$,

$$P^\delta(s' \mid s, a) = \sum_{c \in C} \delta(c) P^c(s' \mid s, a).$$

Observe that P^δ is a convex combination of the base transition functions $\{P^c\}_{c\in C}$. We assume that the uncertainty set $\mathcal{D}$ and the basis $\{P^c\}_{c\in C}$ are known, whereas the sequences of reward functions $r_1, r_2, \ldots$ and transition functions $P_1, P_2, \ldots$ are unknown but do *not* depend on the history of states or agent actions. This last point is in contrast to the model of two-player stochastic games [116].

We assume that the agent observes the states, its actions, and the reward functions. The exception is Section 3.5, where we consider the case where reward observations are limited to the agent's state-action trajectory. Accordingly, the agent's action a_t at time t is a function of the *history*

$$(s_1, a_1, r_1, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$$

up to time t, and, possibly, the agent's individual randomization. In effect, the agent adapts its policy in an *online* fashion. An *algorithm* for the agent is a function that maps, at every time t, every history up to time t to an action $a \in A$. A *stationary* policy $\sigma : S \to A$ for the agent assigns an action to each state, irrespective of the time step and history.

3.2.1 Regret

To measure the performance of an algorithm, we shall compare its *average* undiscounted reward to the reward that could have been obtained by following alternative algorithms. To obtain meaningful results that hold for every sequence of opponent actions, we restrict the class of comparison algorithms to the set of deterministic and stationary policies $\{\sigma : S \to A\}$ that assign actions based solely on the current state. We define the objective

$$\widehat{J}_T \triangleq \max_{\sigma:S\to A} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big[r_t(\widehat{s}_t, \sigma(\widehat{s}_t))\big],$$

where the initial state $\widehat{s}_1$ is fixed, each next state $\widehat{s}_{t+1}$ is distributed according to $P_t(\cdot \mid \widehat{s}_t, \sigma(\widehat{s}_t))$, and the expectation is taken with respect to the sequence $\widehat{s}_1, \widehat{s}_2, \ldots$.

Given the sequences $r_1, r_2, \ldots$, $P_1, P_2, \ldots$, and the sequence $a_1, a_2, \ldots$ of actions generated by the agent's algorithm, we define the *expected time-averaged regret*—or simply called the *regret*—at time T as

$$\widehat{L}_T \triangleq \widehat{J}_T - \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\big[r_t(s_t, a_t)\big], \tag{3.2.1}$$

where the initial state s_1 is assumed fixed throughout the paper, each next state s_{t+1} is distributed according to $P_t(\cdot \mid s_t, \sigma_t(s_t))$, and where the expectation is taken with respect to the probabilistic transitions and the agent's randomization. In this chapter, we seek a *universal* solution in the following sense: the objective is to minimize the regret with respect to every pair of *individual* sequences $\{r_t\}_{t=1,2,\ldots}$ and $\{P_t\}_{t=1,2,\ldots}$.

3.3 Examples and Motivation

Our model is motivated by the following problem.

Example 3.1 (Advertising Problem)**.** Consider the problem of deciding what advertisement to display on a webpage. There is a (finite) set C of types of viewers. Each type of viewers behaves differently. For instance, some viewers do not respond to advertisements, some respond to particular advertisements, etc. The operator receives a payoff when viewers are redirected to an advertiser. However, some of the redirected viewers may not return to the webpage. This trade-off between the number of viewers and the payoff can be modelled by an MDP. The instantaneous reward is the number of redirects. The state captures the budgets of the advertisers, and the composition of viewers. A similar problem is described in [1, 105].

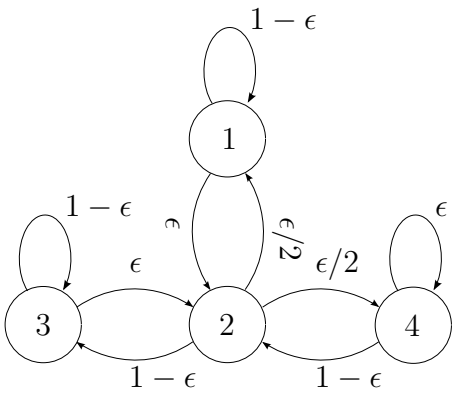
When there is a single viewer of type c, the state-transition function $P^c(\cdot \mid \cdot, a)$ depends on both the agent's action a and the type c. In general, however, the group of viewers is heterogeneous. The overall transition function is more accurately captured by a mixture of the functions $\{P^c\}_{c \in C}$; each function P^c being weighted by the proportion $\delta_t(c)$ of viewers of type c.

Furthermore, in real life, the composition of viewers depends on time of day, day of the week, and unpredictable events. Hence, we model the composition of viewers δ_t as a process that may change over time in an arbitrary fashion. The instantaneous payoff to the owner is also a quantity that is difficult to predict: it depends on external factors such as the quality of articles on the webpage, the relevance of the advertisement to the viewers, etc. Hence, we model the sequence of reward functions $r_1, r_2, \ldots$ as an arbitrary sequence as well.

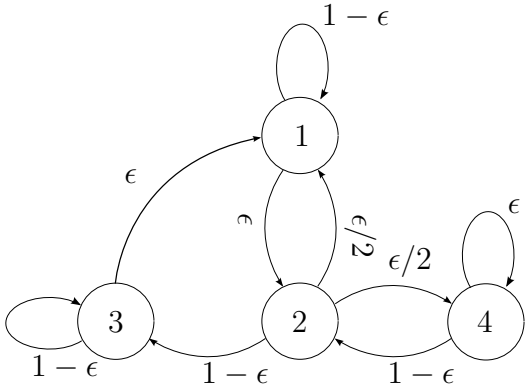
Our model generalizes that of prediction with expert advice by adding the notion of state. The state, which may specify the arm selected in the previous step, evolves according to Markovian dynamics, but the transition probabilities are uncertain, *e.g.*, the next-state distribution following each arm-selection is only specified within some range. This uncertainty may arise from inaccurate measurement or estimation, from an uncertainty principle [43], or from nonstationary variations in the probabilities. The latter occur, for instance, in systems that rely on replaceable components whose specifications fall within some ϵ-tolerance range (*e.g.*, battery life, physical dimensions).

Our model also encompasses MDPs where the transition function may change abruptly at unknown time instants [57], *e.g.*, as a result of a mechanical break-down, or triggered by an adversary. Consequently, the true state-transition model may deviate unpredictably from the nominal model.

In the next example, we show that arbitrary changes in the transition probabilities can cause large regret for online learning algorithms that compute policies based on a chosen transition function that differs—however slightly—from the true transition function. Protection against such breakdowns motivates the introduction of the algorithms presented in the following sections.



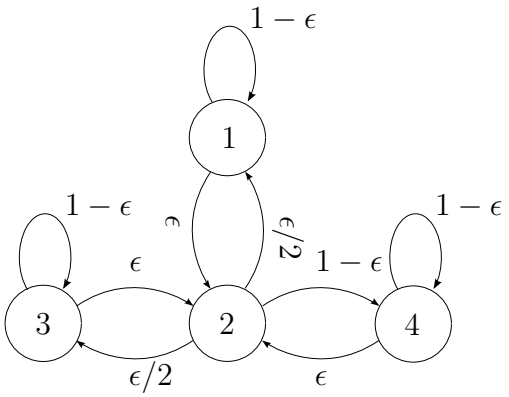
(a) Transition model $P_t(\cdot \mid \cdot, \mathcal{L})$ at even time steps.



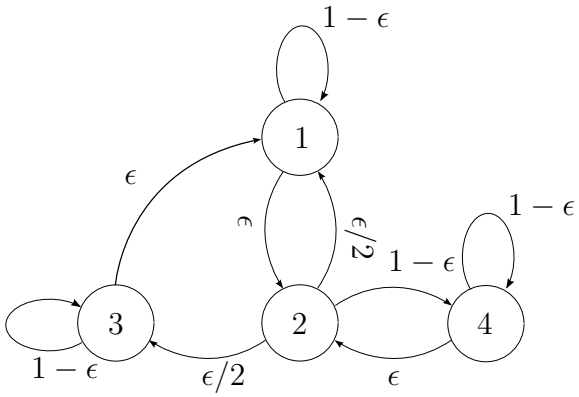
(b) Transition model $P_t(\cdot \mid \cdot, \mathcal{L})$ at odd time steps.

Figure 3–1: Transition models for Example 3.2 when agent chooses the action $\mathcal{L}$. Edges are labelled with transition probabilities.

Example 3.2. Consider an MDP with states $\{1, 2, 3, 4\}$. The choices of actions for the agent are $\mathcal{L}$ and $\mathcal{R}$. The rewards are constant over time. Taking action $\mathcal{L}$ at state 3 gives a reward of 1. Taking action $\mathcal{R}$ at state 4 gives a reward of 7/8. All actions in other states give zero reward. Figure 3–1(a) shows the nominal state-transition model when the agent takes action $\mathcal{L}$. However,



(a) Transition model $P_t(\cdot \mid \cdot, \mathcal{R})$ at even time steps.



(b) Transition model $P_t(\cdot \mid \cdot, \mathcal{R})$ at odd time steps.

Figure 3–2: Transition models for Example 3.2 when agent chooses the action $\mathcal{R}$. Edges are labelled with transition probabilities.

unknown to the agent, the transition model changes to that of Figure 3–1(b) at odd time steps. This change is small in the sense that the probabilities change by at most a fixed $\epsilon > 0$. When the agent chooses the action $\mathcal{R}$, the transitions toward state 4 are more likely, as shown in Figure 3–2.

Suppose that the agent takes the transition model of Figure 3–1(a) as the nominal model for all time steps in order to compute a regret-minimizing policy by perturbing the optimal policy, *e.g.*, using the Lazy FPL algorithm of Chapter 2. The unperturbed optimal policy in this case is to take the $\mathcal{L}$ action at every state. Since the random perturbation tends to zero, the regret-minimizing policy also chooses the $\mathcal{L}$ action with arbitrarily high probability

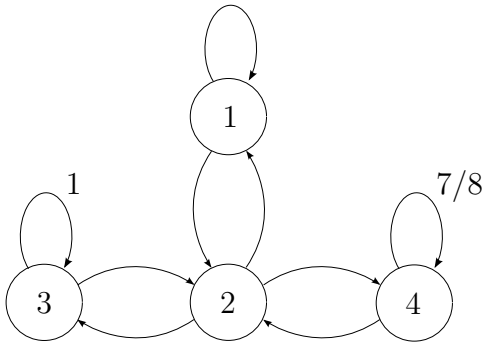


Figure 3–3: Rewards for Example 3.2.

after enough time. However, choosing the $\mathcal{L}$ action realizes an expected reward of at most $1/2$ per step when the actual transition model is that of Figure 3–1(b), *i.e.*, at the odd time steps. Since the expected average reward at even time steps is at most 1, the overall expected average reward of this policy is at most $3/4$. In contrast, a policy that is robust with respect to the two possible transition models (Figure 3–1(a) and 3–1(b)) chooses the action $\mathcal{R}$ at all times; hence, generating an expected average reward greater or equal to $7/8-\epsilon$. The difference in average reward between the regret minimizing policy and the robust policy can be made arbitrarily large by tuning the rewards.

One approach of minimizing the regret in this example is to combine robust dynamic programming and online learning (Section 3.4). Another approach uses a simulation-based algorithm (Section 3.6), which eliminates the need to observe or estimate the transition probabilities $\{P_t\}$ and the modulating sequence $\{\delta_t\}$.

3.3.1 Assumptions

First, we make a standard ergodicity assumption for Markov decision processes. This assumption excludes sequences of transition functions that create periodicity and prevent vanishing regret, as shown in Chapter 2. Recall

that for a fixed Markov chain $x_1, x_2, \ldots$ with state space S, the cover time is

$$T_{\text{cov}} = \min\{t : \text{for every } s \in S, \text{ there exists } j < t \text{ such that } x_j = s\},$$

i.e., the smallest integer t such that for every $s \in S$, there exists $j < t$ with $x_j = s$. In turn, the expected cover time is

$$\max_{s \in S} \mathbb{E}_{x_1 = s} [T_{\text{cov}}],$$

where the maximum is over all possible values of the initial state x_1. Let d_t denote the distribution of the state x_t at time t, and π denote the stationary distribution. The mixing time of the Markov chain $x_1, x_2, \ldots$ is

$$\min \left\{ t : \frac{1}{2} \sum_{s \in S} |d_t(s) - \pi(s)| \leq 1/4 \right\}.$$

Assumption 3.1 (Uniform Ergodicity)**.** The sequence of transition functions $P_1, P_2, \ldots$ is non-periodic. There exist constants τ and τ_{cov} such that for every (mixed) policy $\sigma : S \to \Delta(A)$ and every modulating vector $\delta \in \Delta(C)$, the Markov chain induced by the transition function $P^{\delta}(s' \mid s, \sigma(s))$ is ergodic with expected mixing time at most τ and expected cover time at most τ_{cov}.

Assumption 3.1 ensures that the Markov chain induced by every sequence of transition functions $P_1, P_2, \ldots$ and every policy $\sigma : S \to A$ is ergodic: all states in the induced Markov chain communicate and are aperiodic; moreover, the Markov chain admits a unique stationary distribution.

In general, the state transition functions and the reward functions vary quite differently. It may be appropriate to assume that the agent has more control over the system's state than the source of rewards (*e.g.*, Nature). This gives rise to some asymmetry in the extent of arbitrary uncertainty in the rewards and transitions, as illustrated by the following game.

Example 3.3 (AR-AT Games)**.** Additive-reward additive-transition stochastic games (cf. [50]) can provide natural examples where two players have non-symmetric influence on the transition probabilities. In such games, the agent's reward and transition functions at each time t can be decomposed as:

$$P_t(s' \mid s, a) = (1-\lambda)G_1(s' \mid s, a) + \lambda G_2(s' \mid s, b_t),$$
$$r_t(s, a) = (1-\nu)f_1(s, a) + \nu f_2(s, b_t),$$

where G_1 and f_1 are single-controller transition and reward functions in the absence of an opponent. However, the opponent takes the action b_t, and influences the reward and state-transition through f_2 and G_2. Hence, the functions P_t and r_t depend on the opponent action b_t, where the scalars λ and ν control the extent of this dependence. An example of such a stochastic game is the fishery game described in [50].

In light of the hardness of obtaining vanishing regret, we set out to quantify the dependence of the regret on the variability in the transition probabilities. To this effect, similarly to the role of the scalars λ and ν in Example 3.3, we introduce the following assumption to limit the extent to which the transition functions $\{P_i : i = 1, 2, \ldots\}$ may vary when modulated by the arbitrary individual sequence $\{\delta_i \in \Delta(C) : i = 1, 2, \ldots\}$.

Assumption 3.2 (ϵ-arbitrary Transition Functions)**.** Let $\mathbf{P}^\delta$ denote the matrix whose (s', s)-element is $P^\delta(s' \mid s, \sigma(s))$. Let $\mathbf{Z}^\delta$ denote the fundamental matrix (cf. [115]) associated with the same transition function P^δ, *i.e.*,

$$\mathbf{Z}^\delta \triangleq [\mathbf{I} - \mathbf{P}^\delta + \mathbf{P}^\delta_\infty]^{-1}, \quad \text{where} \quad \mathbf{P}^\delta_\infty \triangleq \lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} (\mathbf{P}^\delta)^k.$$

Further, let the norm $\|\mathbf{M}\|_\infty$ of a matrix $\mathbf{M}$ denote its maximum absolute row-sum. We assume that there exists a finite constant Z such that $\left\|\mathbf{Z}^\delta\right\|_\infty \leq Z$

for all $\delta \in \mathcal{D}$, and that there exists $\epsilon > 0$ such that

$$\left\| \mathbf{P}^{\delta} - \mathbf{P}^{\delta'} \right\|_{\infty} < \epsilon$$

for every $\delta \in \mathcal{D}$ and $\delta' \in \mathcal{D}$.

3.4 Full Observation Case

In this section, we present a solution that combines robust optimization and existing minimum-regret algorithms for single-controller stochastic games (*e.g.*, [50, 44]). In particular, we employ implementations of these algorithms that follow the best policy with respect to the empirical observations thus far, while subject to small perturbations. Our approach uses robust control of MDPs with infinite-horizon average-reward objectives, whereas robust control of MDPs with finite horizons and discounted rewards may be solved by the robust dynamic programming method of [101]. This robust dynamic programming method may be adapted to MDPs with the infinite-horizon average-reward objective by using appropriate Bellman equations (cf. [19]).

For this section, we begin by making some additional assumptions that will be relaxed in the following sections. First, we assume that the agent observes the reward function r_t at the end of every time step t. Second, we assume that the set of basic transition functions $\{P^c : c \in C\}$ and the modulating sequence uncertainty set $\mathcal{D}$ are known to the agent. For ease of exposition, we define the empirical average of reward functions observed from time 1 to $t-1$:

$$\widehat{r}_t(s,a) \triangleq \frac{1}{t-1} \sum_{i=1}^{t-1} r_i(s,a), \quad \text{for all } (s,a) \in S \times A.$$

Consider the online robust dynamic programming (ORDP) algorithm of Algorithm 3–1. This algorithm combines robust control of MDPs [101] with regret-minimizing policies for single-controller stochastic games (*e.g.*, [50, 44]).

At every time step, the ORDP algorithm solves a robust dynamic program with respect to the uncertainty set $\mathcal{D}$, and with the average reward function $\widehat{r}_t$. The algorithm then acts according to a randomly perturbed version of the optimal policy.

The ORDP algorithm uses the same ideas as the Lazy FPL algorithm of Chapter 2, with the addition of a robust optimization step and the exception of the time phases. However, the analysis differs significantly due to the changing transition functions.

(*Initialize.*) At time step 1, output an arbitrary action a_1.
At every time step $t = 2, 3, \ldots$:

1. (*Robust dynamic program.*) Solve the robust MDP with state space S, action space A, an uncertainty set $\mathcal{D} \subseteq \Delta(C)$, and with reward function $\widehat{r}_t$. In other words, solve the following Bellman equations for MDPs with infinite-horizon average-reward objective [19] (via linear programming or otherwise):

$$V_t(s) = \max_{a \in A} \Big(\widehat{r}_t(s,a) + \inf_{\delta \in \mathcal{D}} \sum_{s' \in S} V_t(s') \sum_{c \in C} \delta(c) P^c(s' \mid s_t, a) - V_t(s_0) \Big), \quad s \in S, \tag{3.4.2}$$

   where $s_0 \in S$ is a fixed state and $V_t(s_0)$ is a normalization term.
2. (*Sample.*) Sample a random variable $\mathbf{n}_t$ uniformly over the support $[-t^{-1/2}, t^{-1/2}]^{|A|}$.
3. (*Follow the perturbed leader.*) Output the action

$$a_t(s_t) = \arg\max_{a \in A} \Big(\widehat{r}_t(s_t,a) + \mathbf{n}_t(a) + \inf_{\delta \in \mathcal{D}} \sum_{s' \in S} V_t(s') \sum_{c \in C} \delta(c) P^c(s' \mid s_t, a) \Big). \tag{3.4.3}$$

Algorithm 3–1: Online robust dynamic programming (ORDP) algorithm

Theorem 3.1. *Suppose that Assumptions 3.1 and 3.2 hold. Then, for $T \geq 4e\tau/\epsilon$, the regret of the ORDP algorithm with respect to every sequence $r_1, r_2, \ldots$*

and $P_1, P_2, \ldots$ is bounded as follows:

$$\widehat{L}_T \leq (Z+1)\epsilon + \sqrt{|S|\,|A|\,/T} + 4\tau^2\sqrt{\log(|A|)/T} + 4\tau/T.$$

3.4.1 Proof of Theorem 3.1

Let a stationary policy $\mu : S \to A$ and a modulating vector $\delta \in \Delta(C)$ be fixed, and let $\mathbf{u}$ be a state distributed according to the stationary state-distribution $\pi(\mu, \delta)$ associated with the transition function $P^{\delta}(s' \mid s, \mu(s))$. We first define a relaxed objective:

$$\widehat{I}_T \triangleq \inf_{\delta \in \Delta(C)} \max_{\mu : S \to A} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\pi(\mu,\delta)}\big[r_t(\mathbf{u}, \mu(\mathbf{u}))\big], \tag{3.4.4}$$

where the expectation $\mathbb{E}_{\pi(\mu,\delta)}$ makes explicit the dependence of the distribution of $\mathbf{u}$ on the modulating vector δ. Theorem 3.1 follows from two steps. First, using the following Lemma 3.2, we relate the relaxed objective $\widehat{I}_T$ and the true objective $\widehat{J}_T$. Then, through Lemma 3.3 and Proposition 3.4, we bound the regret with respect to the relaxed objective.

Lemma 3.2. *Suppose that Assumptions 3.1 and 3.2 hold. Then, for every sequence $r_1, r_2, \ldots$ and $P_1, P_2, \ldots$, we have $|\widehat{J}_T - \widehat{I}_T| \leq (Z+1)\epsilon$, for $T \geq 2e\tau/\epsilon$.*

Proof. Let σ^* denote a stationary policy that achieves the true objective:

$$\max_{\sigma : S \to A} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\big[r_t(\widehat{\mathbf{s}}_t, \sigma(\widehat{\mathbf{s}}_t))\big].$$

Let δ^* denote a modulating vector that achieves the following objective:

$$\inf_{\delta \in \Delta(C)} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\pi(\sigma^*,\delta)}\big[r_t(\mathbf{u}, \sigma^*(\mathbf{u}))\big].$$

Recall that $\pi(\mu, \delta)$ denotes the distribution of $\mathbf{u}$ induced by the policy μ and the transition function P^{δ}. Observe that the policy σ^* and P^{δ} induce a Markov chain with a unique stationary distribution $\pi(\sigma^*, \delta)$ by Assumption 3.1. Since the transition functions are ϵ-arbitrary by Assumption 3.2, every such induced

stationary distribution satisfies

$$\|\pi(\sigma^*, \delta) - \pi(\sigma^*, \delta^*)\|_\infty \leq Z\epsilon \tag{3.4.5}$$

by [115, Section 6]. Let $d_t(\sigma^*, \delta)$ denote the distribution of $\widehat{s}_t$ when the agent follows policy σ^* and the transition function is P^δ. By Assumption 3.1, every Markov chain induced by σ^* and P^δ has a mixing time bounded by τ. Hence, we have

$$\|d_t(\sigma^*, \delta) - \pi(\sigma^*, \delta)\|_\infty \leq 2e^{1-t/\tau}. \tag{3.4.6}$$

Let $d_t(\sigma^*)$ denote the distribution of state $\widehat{s}_t$ following the transition functions $P_1, \ldots, P_{t-1}$, while the agent follows policy σ^*. Observe that, by ergodicity assumption, every transition matrix P_i has the spectral decomposition $P_i = UD_iU^{-1}$ for $i = 1, \ldots, t-1$. Let $\delta \in \Delta(C)$ be such that $P^\delta = U(D_{t-1} \ldots D_1)^{1/(t-1)}U^{-1}$, then we have

$$\begin{aligned} \|d_t(\sigma^*) - \pi(\sigma^*, \delta)\|_\infty &\leq \|d_t(\sigma^*, \delta) - \pi(\sigma^*, \delta)\|_\infty \\ &\leq 2e^{1-t/\tau}, \end{aligned} \tag{3.4.7}$$

where the last inequality follows from (3.4.6).

By observing that the rewards take values in $[0, 1]$, it follows from (3.4.5) and (3.4.7) that

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\big[r_t(\widehat{s}_t, \sigma^*(\widehat{s}_t))\big] - \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\pi(\sigma^*, \delta^*)}\big[r_t(\mathbf{u}, \sigma^*(\mathbf{u}))\big] \leq Z\epsilon + 2e\tau/T.$$

In turn, by definition of σ^* and δ^*, and for $T \geq 2e\tau/\epsilon$, we obtain

$$\max_{\sigma: S \to A} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\big[r_t(\widehat{\mathbf{s}}_t, \sigma(\widehat{\mathbf{s}}_t))\big] - \inf_{\delta \in \Delta(C)} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\pi(\sigma^*, \delta)}\big[r_t(\mathbf{u}, \sigma^*(\mathbf{u}))\big] \leq (Z+1)\epsilon.$$

By replacing the policy σ^* in the negative term on the left-hand side by an optimal one, we obtain:

$$\max_{\sigma:S\to A}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[r_t(\widehat{\mathbf{s}}_t,\sigma(\widehat{\mathbf{s}}_t))\big]-\max_{\mu:S\to A}\inf_{\delta\in\Delta(C)}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\mu,\delta)}\big[r_t(\mathbf{u},\mu(\mathbf{u}))\big]\leq(Z+1)\epsilon.$$

The final result follows by observing that the relaxed objective is the value of a game where the players' actions are fixed policies. This value exists [101, Theorem 4]; and hence,

$$\inf_{\delta\in\Delta(C)}\max_{\mu:S\to A}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\mu,\delta)}\big[r_t(\mathbf{u},\mu(\mathbf{u}))\big]=\max_{\mu:S\to A}\inf_{\delta\in\Delta(C)}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\mu,\delta)}\big[r_t(\mathbf{u},\mu(\mathbf{u}))\big].$$

■

Lemma 3.3 ([45, Proposition 2])**.** *Let an arbitrary $\delta\in\Delta(C)$ be fixed. Suppose that the sequence of transition functions is $P_t=P^\delta$ for all t. Let the uncertainty set be reduced to the singleton: $\mathcal{D}=\{\delta\}$. Then, the regret of the policy sequence $\{\sigma_t\}$ induced by the ORDP algorithm is bounded as follows:*

$$\begin{aligned}&\max_{\mu:S\to A}\mathbb{E}_{\pi(\mu,\delta)}\big[\widehat{r}_T(\mathbf{u},\mu(\mathbf{u}))\big]-\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[r_t(s_t,\sigma_t(s_t))\big]\\&\leq\sqrt{|S|\,|A|\,/T}+4\tau^2\sqrt{\log(|A|)/T}+4\tau/T.\end{aligned}$$

Proof. By [44, Theorem 6.2], for every δ, we have

$$\begin{aligned}&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[r(s_t,\sigma_t(s_t))\big]-\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\sigma_t,\delta)}\big[r_t(\mathbf{v}_t,\sigma_t(\mathbf{v}_t))\big]\\&\leq 4\tau^2\sqrt{\log(|A|)/T}+2\tau/T,\end{aligned}$$

where σ_t is the agent's policy at time t, and $\mathbf{v}_t$ is distributed according to the stationary distribution $\pi(\sigma_t,\delta)$. By [44, Theorem 6.1], for every agent-policy

$\sigma : S \to A$, we have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[r_t(s_t,\sigma(s_t))\big] \le \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\sigma_t,\delta)}\big[r_t(\mathbf{v}_t,\sigma(\mathbf{v}_t))\big] + 2\tau/T.$$

By [44, Theorem 5.2], for every σ, we also have

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\sigma_t,\psi)}\big[r_t(\mathbf{v}_t,\sigma_t(\mathbf{v}_t))\big] \ge \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\sigma_t,\psi)}\big[r_t(\mathbf{v}_t,\sigma(\mathbf{v}_t))\big] - \sqrt{|S|\,|A|\,/T}.$$

The claim follows by combining these three inequalities, and recalling the definition:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\pi(\mu,\delta)}\big[r_t(\mathbf{u},\mu(\mathbf{u}))\big] = \mathbb{E}_{\pi(\mu,\delta)}\big[\widehat{r}_T(\mathbf{u},\mu(\mathbf{u}))\big].$$

■

Proposition 3.4. *Under the assumptions of Theorem 3.1, the regret of the ORDP algorithm with respect to the relaxed objective $\widehat{I}_T$ is*

$$\widehat{I}_T - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[r_t(s_t,a_t)\big] \le \sqrt{|S|\,|A|\,/T} + 4\tau^2\sqrt{\log(|A|)/T} + 4\tau/T, \quad \textit{for all } T. \tag{3.4.8}$$

Proof. Let $\{\rho_t\}$ denote the sequence of policies induced by the ORDP algorithm with $\mathbf{n}_t = 0$ for all t. The policy ρ_t solves a robust dynamic program with an undiscounted average-reward objective; it can be obtained as the solution of a robust linear program [127]. Hence, by an argument similar to [101, Theorem 3], the policy ρ_t inherits the following optimality property: at every step t, and for every $\delta \in \Delta(C)$,

$$\mathbb{E}_{\pi(\rho_t,\delta)}\big[\widehat{r}_t(\mathbf{v},\rho_t(\mathbf{v}))\big] \ge \max_{\mu:S\to A}\mathbb{E}_{\pi(\mu,\delta)}\big[\widehat{r}_t(\mathbf{u},\mu(\mathbf{u}))\big].$$

By replacing the optimality property for dynamic programming by the above optimality property for robust dynamic programming, the claimed result follows by Lemma 3.3. ■

Since the ORDP algorithm solves a robust MDP at every time step, its computational complexity increases linearly in time. However, the policies computed by the ORDP algorithm change slowly in time, especially after a long time period. This observation suggests the alternative of computing a new policy only once in a while and following this policy in the meantime. Choosing the length of the intervening intervals provides a mean of trade-off between the regret bound and the computational complexity of the solution. This approach is similar to other lazy algorithms [7].

The trade-off between regret and complexity can be achieved by modifying the ORDP algorithm as follows. First, we partition the time horizon $1, 2, \ldots$ into intervals of time steps, whose lengths increase over time. Next, we compute a single stationary policy in each interval and follow it throughout the interval. For instance, if only $T^{3/4}$ different policies are computed up to time T, the corresponding regret bound is of the order of $O(T^{-1/4})$ (cf. Chapter 2).

3.5 Limited Observation of Rewards

In this section, we consider a limited-observation setting similar to the multi-armed bandit setting [10]. At the end of every time step, instead of observing the full reward function r_t, the agent observes only the reward $r_t(s_t, a_t)$ that was received. This setting is representative of applications in sequential clinical trials and sequential allocation of resources. Since the agent only observes the values of the reward functions on its state-action trajectory, our solution maps the observed rewards $r_t(s_0, a_0), \ldots, r_t(s_{t-1}, a_{t-1})$ and the current state s_t to an action a_t. We present an algorithm that uses estimates of the reward functions as in [10].

Our approach is to construct an interval estimate of $\widehat{r}_t$ at every t. To this end, we construct a *random* interval $[\mathbf{z}_t^-, \mathbf{z}_t^+]$ at every step t as follows.

Starting from the initial state s_1, at every subsequent step t, we compute the following lower and upper bounds for the state-distribution d_{s_t} of s_t:

$$d^-_{s_t}(s) = \inf_{\delta \in \mathcal{D}} P^\delta(s \mid s_{t-1}, a_{t-1}), \quad \text{for all } s \in S,$$
$$d^+_{s_t}(s) = \sup_{\delta \in \mathcal{D}} P^\delta(s \mid s_{t-1}, a_{t-1}), \quad \text{for all } s \in S. \tag{3.5.9}$$

Let σ_t denote the policy of the algorithm at time t, *i.e.*, $\sigma_t(a \mid s)$ is the probability that its algorithm outputs $a_t = a$ provided that $s_t = s$. Given a fixed pair $(s, a) \in S \times A$, the probability that $(s_t, a_t) = (s, a)$ is bounded from above by $\sigma_t(a \mid s)\, d^+_{s_t}(s)$. For time steps t such that $\sigma_t(a \mid s)\, d^+_{s_t}(s)$ is positive, we define

$$\mathbf{z}^-_t(s,a) = \begin{cases} \frac{r_t(s,a)}{\sigma_t(a|s) d^+_{s_t}(s)}, & \text{if } (s,a) = (s_t, a_t), \\ 0, & \text{otherwise.} \end{cases} \tag{3.5.10}$$

Likewise, we define

$$\mathbf{z}^+_t(s,a) = \begin{cases} \frac{r_t(s,a)}{\sigma_t(a|s) d^-_{s_t}(s)}, & \text{if } (s,a) = (s_t, a_t), \\ 0, & \text{otherwise.} \end{cases}$$

To ensure that the probabilities are non-zero, we explore the state-action space $S \times A$ during the N initial time steps, where N is the earliest time step such that $d^-_{s_t}(s) > 0$ and $d^+_{s_t}(s) > 0$ for all $s \in S$. Observe that only the value $r_t(s_t, a_t)$ of the reward at the traversed state-action profile (s_t, a_t) is required to evaluate $\mathbf{z}^-_j$ and $\mathbf{z}^+_j$. From the sequences $\mathbf{z}^-_j$ and $\mathbf{z}^+_j$, we construct $\widehat{\mathbf{z}}^-_t \triangleq \frac{1}{t-1} \sum_{j=1}^{t-1} \mathbf{z}^-_j$ and $\widehat{\mathbf{z}}^+_t \triangleq \frac{1}{t-1} \sum_{j=1}^{t-1} \mathbf{z}^+_j$, which are lower and upper estimates of the empirical-average reward function $\widehat{r}_t$.

Next, we present an algorithm and its corresponding regret bound in the limited-observation setting. Observe that since the algorithm follows a fixed policy for $t < N$, then N is bounded from above by the cover time of the

Markov chain induced by the fixed policy. Since the worst-case expected cover time is τ_{cov} by Assumption 3.1, we have $\mathbb{E}N \leq \tau_{\text{cov}}$.

(*Initialize.*) Fix $\phi \in (0,1)$. Let N denote the smallest t such that $d^+_{s_t}(s) > 0$ for all $s \in S$.

1. (*Estimate.*) At every time step $t > 1$, compute the reward function lower-estimate $\widehat{\mathbf{z}}^-_t$ recursively from (3.5.9) and (3.5.10).
2. At time steps $t < N$, output an action a_t chosen uniformly at random over A, *i.e.*, with distribution $\sigma_t(a \mid s) = |A|^{-1}$ for all $a \in A$.
3. At time steps $t \geq N$:
   (a) (*Sample.*) Sample an independent Bernoulli random variable $\mathbf{x}_t$ that takes value 1 with probability ϕ.
   (b) (*Minimize regret.*) If $\mathbf{x}_t = 0$, solve the robust MDP corresponding to the Bellman equations (3.4.2) of Algorithm 3–1 with the reward function $\widehat{\mathbf{z}}^-_t$ instead of $\widehat{r}_t$. Output the action a_t as in (3.4.3) of Algorithm 3–1 after replacing $\widehat{r}_t$ by $\widehat{\mathbf{z}}^-_t$.
   (c) (*Explore.*) If $\mathbf{x}_t = 1$, output an action a_t chosen uniformly at random over A.

Algorithm 3–2: Exploratory ORDP

Observe that the fixed parameter $\phi > 0$ ensures that each element of the state-action space $S \times A$ is explored infinitely often as $T \to \infty$. Similar results may be obtained by decreasing the probability of exploration ϕ over time (cf. [10]).

Theorem 3.5 (Exploratory ORDP)**.** *Suppose that Assumptions 3.1 and 3.2 hold. Let $P_{\min} > 0$ denote a positive constant such that $d^+_{s_t}(s) > P_{\min}$ and $d^-_{s_t}(s) > P_{\min}$ for every $s \in S$, initial state $s_0 \in S$, and $t \geq \tau_{\text{cov}}$. If the agent follows the Exploratory ORDP algorithm with a fixed $\phi \in (0,1)$, then, for every $T \geq \max\{\tau_{\text{cov}}\ ,\ 2e\tau/\epsilon\}$, the regret of the Exploratory ORDP algorithm is bounded as follows:*

$$\widehat{L}_T \leq \left(\frac{|A|}{P^2_{\min}\phi} + Z + 1\right)\epsilon + \phi + \sqrt{\frac{|S|\,|A|}{T}} + 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{4\tau}{T} + \frac{\tau_{\text{cov}}}{T}.$$

In particular, if $\phi = \frac{\sqrt{|A|}}{P_{\min}}\sqrt{\epsilon}$*, we obtain*

$$\widehat{L}_T \leq (Z+1)\epsilon + 2\frac{\sqrt{|A|}}{P_{\min}}\sqrt{\epsilon} + \sqrt{\frac{|S|\,|A|}{T}} + 4\tau^2\sqrt{\frac{\log|A|}{T}} + \frac{4\tau+\tau_{\text{cov}}}{T}.$$

Remark 22. Compared to the full observation case (cf. Theorem 3.1), the above regret bound for the limited observation model contains an additional component of $2\frac{\sqrt{|A|}}{P_{\min}}\sqrt{\epsilon}$.

Proof outline. First, observe that, for all $(s,a) \in S \times A$, the true unobserved average reward function $\widehat{r}_t$ is bounded as follows:

$$\mathbb{E}\left\{\widehat{\mathbf{z}}_t^-(s,a) \mid s_{t-1}, a_{t-1}\right\} \leq \widehat{r}_t(s,a) \leq \mathbb{E}\left\{\widehat{\mathbf{z}}_t^+(s,a) \mid s_{t-1}, a_{t-1}\right\}.$$

Next, observe that $\left\|d_{s_t}^- - d_{s_t}^+\right\|_\infty \leq \epsilon$ by definition and Assumption 3.2. By definition, we have

$$\mathbb{E}\left\{\left|\mathbf{z}_t^-(s,a) - \mathbf{z}_t^+(s,a)\right| \mid s_{t-1}, a_{t-1}\right\} \leq \frac{r_t(s,a)}{\sigma_t(a \mid s) d_{s_t}^-(s) d_{s_t}^+(s)}(d_{s_t}^-(s) - d_{s_t}^+(s)),$$
$$\text{for all } (s,a) \in S \times A.$$

By definition of the Exploratory ORDP algorithm, we have $\sigma_t(a \mid s) \geq \phi/|A|$. Also, for $t \geq \tau_{\text{cov}}$, we have the bounds $d_{s_t}^-(s) \geq P_{\min}$ and $d_{s_t}^+(s) \geq P_{\min}$. It follows that

$$\mathbb{E}\left\{\left|\widehat{\mathbf{z}}_t^-(s,a) - \widehat{\mathbf{z}}_t^+(s,a)\right| \mid s_{t-1}, a_{t-1}\right\} \leq \frac{|A|\,\epsilon}{P_{\min}^2 \phi}$$

for every $(s,a) \in S \times A$ and $t \geq N$. Thus, by replacing $\widehat{r}_t$ by $\widehat{\mathbf{z}}_t^-$—or an arbitrary convex combination of $\widehat{\mathbf{z}}_t^-$ and $\widehat{\mathbf{z}}_t^+$, the Exploratory ORDP algorithm incurs at most an additional regret of $|A|\,\epsilon/(P_{\min}^2\phi)$ over the ORDP algorithm. Further, observe that the first N initialization steps incur an overhead regret of at most N, whereas the recurrent exploration steps incur an expected overhead

of ϕ T. Finally, we obtain the bound of Theorem 3.5 by combining these additional terms with the bound of Theorem 3.1. ■

3.6 Unknown Transition Probabilities

In this section, we consider the case where the agent does not observe the state-transition probabilities, but only the state trajectory. As in the previous sections, we assume that the changes in the transition function, although arbitrary, are limited. One method is to estimate the transition probabilities (as in [30]), define an appropriate uncertainty set, and employ the methods of Section 3.4. We adopt a different method in this section. By using the Q-learning approach [126], we eliminate the need to know a priori or to estimate the transition probabilities.

Recall that in Example 3.2, we show that any discrepancy, no matter how small, between transition function used to compute the policy and the true transition function can cause arbitrarily high regret. Unlike the dynamic programming approach of the previous sections however, the Q-learning approach is simulation-based: the policy is generated according to the true state-action trajectory, which is induced by the true transition function. Hence, as a second advantage, this approach does not require an additional robust optimization over the uncertainty set $\mathcal{D}$ of the transition functions. As in the previous sections, the regret is proportional to the variability of the uncertainty.

For simplicity, we assume in this section that the reward function is fully observed at each time step, as in Section 3.4. However, the method for limited observation described in the previous section can be readily applied here as well.

3.6.1 Algorithm

Consider the Q-FPL algorithm of Algorithm 3–3, where an appropriate Q-function recursion replaces the Bellman equations (3.4.2) of the ORDP algorithm. We partition the time horizon $1, 2, \ldots$ into *intervals* of time steps, denoted by $\alpha_1, \alpha_2, \ldots$. We denote by K_m the length of each interval α_m. Within each interval α_m, we use a fixed learning rate γ_m for the Q-function iterations. Let t_m denote the first step of the interval α_m.

(*Initialize.*) For $t \in \alpha_1$, choose the action $a_t(s_t)$ according to an arbitrary deterministic policy $\mu : S \to A$.
For every phase α_m, where $m = 2, 3, \ldots$:

1. (*Q-learning.*) At the first step t of interval α_m, reset the Q-function Q_t to a vector of zeroes. At every step $t+1 \in \alpha_m$, recursively update the Q-function:

$$Q_{t+1}(s,a) = Q_t(s,a) + \gamma_m \Big(\widehat{r}_{t_m}(s,a) + \max_{a' \in A} Q_t(s_{t+1}, a') - Q_t(s,a) - Q_t(s_0,a_0) \Big), \quad \text{if } (s,a) = (s_t, a_t),$$
$$Q_{t+1}(s,a) = Q_t(s,a) \quad \text{otherwise,}$$

   where $Q_t(s_0, a_0)$ is a normalizing term, and the learning rate is $\gamma_m \in (0, 1/4)$.
2. (*Follow the perturbed leader.*) Let $Q^{\alpha_{m-1}}$ denote the Q-function computed at the end of the interval α_{m-1}. At every step $t \in \alpha_m$, choose the action

$$a_t(s_t) = \arg\max_{a \in A} \Big\{ Q^{\alpha_{m-1}}(s_t, a) + \mathbf{n}_t(a) \Big\}, \tag{3.6.11}$$

   where the element of A with the lowest index is taken if the max is not unique, and where the random variable $\mathbf{n}_t$ is sampled uniformly over the support

$$\left[-\left(\sum_{i=1}^{m} K_i \right)^{-1/2}, \left(\sum_{i=1}^{m} K_i \right)^{-1/2} \right]^{|A|}.$$

Algorithm 3–3: Q-Follow the Perturbed Leader (Q-FPL) algorithm

The main feature of the Q-FPL algorithm is that it uses the policy learnt in the previous interval while learning a new policy in the current interval.

The details are as follows. Observe the sequence of Q-functions $\{Q_t\}_{t\in\alpha_m}$ is determined by the fixed reward function $\widehat{r}_{t_m}$ and the sequence of transition functions $\{P_t\}_{t\in\alpha_m}$. The Q-FPL algorithm essentially performs standard Q-learning within each interval. The Q-function computed at the final step of the interval α_{m-1} is denoted by $Q^{\alpha_{m-1}}$ and used to derive a new policy (cf. (3.6.11)) for use throughout the next interval α_m. As in the ORDP algorithm, this policy is computed according to the concept of "follow the perturbed leader" ([64, 72]), where continuity between successive policies is ensured by adding a vanishing noise term $\mathbf{n}_t$ to the Q-function $Q^{\alpha_{m-1}}$. However, the Q-FPL algorithm is computationally more efficient: at each time step, we only iterate the Q-function instead of solving a robust dynamic program as in the ORDP algorithm.

For the purpose of analysis, we consider the synchronous version of the Q-learning iteration in the Q-FPL algorithm and write it in the form of a stochastic approximation algorithm (cf. [79]):

$$
\begin{aligned}
Q_{t+1} &= Q_t + \gamma_m(h_t(Q_t) + M_{t+1}), \quad t \in \alpha_m, \\
h_t(Q)(s,a) &= \widehat{r}_{t_m}(s,a) + \sum_{s'\in S} P_t(s' \mid s,a) \max_{a''\in A} Q(s',a'') - Q(s,a) - Q(s_0,a_0), \\
M_{t+1}(s,a) &= \max_{a'\in A} Q_t(s_{t+1},a') - \sum_{s'\in S} P_t(s' \mid s,a) \max_{a''\in A} Q_t(s',a''),
\end{aligned}
$$

where the state s_{t+1} is distributed according to $P_t(\cdot \mid s,a)$. Our analysis of synchronous Q-learning can be extended to the asynchronous version, as done in [29]. In contrast to the convergence analysis for average-reward Q-learning ([29, 2]), the function h_t is not fixed, but may change arbitrarily in our setting.

We also define the sequence Q_t^δ with the fixed transition function P^δ and the following recursion:

$$Q_{t+1}^\delta = Q_t^\delta + \gamma_m(h^\delta(Q_t^\delta) + N_{t+1}), \quad t \in \alpha_m,$$
$$h^\delta(Q)(s,a) = \widehat{r}_{t_m}(s,a) + \sum_{s' \in S} P^\delta(s' \mid s,a) \max_{a'' \in A} Q(s',a'') - Q(s,a) - Q(s_0,a_0),$$
$$N_{t+1}(s,a) = \max_{a' \in A} Q_t^\delta(\tilde{s}_{t+1},a') - \sum_{s' \in S} P^\delta(s' \mid s,a) \max_{a'' \in A} Q_t^\delta(s',a''),$$

where the state $\tilde{s}_{t+1}$ is distributed according to $P^\delta(\cdot \mid s,a)$. Observe that, for every fixed $\delta \in \Delta(C)$, the function h^δ satisfies the sufficient conditions[2] for the convergence of Q_t^δ. Furthermore, $\{N_t\}$ is a martingale difference sequence. By [29, Theorem 2.2], the sequence Q_t^δ converges almost surely; we denote this limit by Q^δ.

In contrast to $\{Q_t^\delta\}$, the sequence $\{Q_t\}$ does not converge to a single limit point due to the arbitrary sequence of transition functions $P_1, P_2, \ldots$. In the following section, we describe its convergence to a limit set.

3.6.2 PAC Regret Bound

In this section, we present a probably approximately-correct (PAC) bound on the regret of the Q-FPL algorithm.

For every basic transition function P^c, for $c \in C$, we define

$$h^c(Q)(s,a) = \widehat{r}_{t_m}(s,a) + \sum_{s' \in S} P^c(s' \mid s,a) \max_{a'' \in A} Q(s',a'') - Q(s,a) - Q(s_0,a_0).$$

Next, we define the convex hull in $\mathbb{R}^{S \times A}$:

$$G(Q) \triangleq \mathrm{conv}(\{h^c(Q) : c \in C\}).$$

[2] The sufficient conditions are that h^δ is Lipschitz continuous and that there exits a function h_∞ such that $\lim_{j \to \infty} h^\delta(jx)/j = h_\infty(x)$ for all x [29].

We denote by $\operatorname{dist}(x, Y)$ the distance of the point x to the set Y: $\inf_{y \in Y} \|x - y\|_\infty$.

Suppose that we write $\gamma(t)$ to denote γ_m if t is a time step belonging to α_m. For every sequence of Q-functions $\{Q_i\}$, we construct an interpolation function $q : \mathbb{R} \to \mathbb{R}^{|S \times A|}$ such that for every integer t, we have

$$q\left(\sum_{i=1}^{t} \gamma(t)\right) = Q_t.$$

Finally, we denote the invariant set of the differential inclusion $\frac{\mathrm{d}q(t)}{\mathrm{d}t} \in G(q(t))$ by

$$\Psi \triangleq \left\{ q : \frac{\mathrm{d}q(t)}{\mathrm{d}t} \in G(q(t)) \text{ for } t \geq 0 \right\}.$$

The following lemma due to [29] asserts that every function belonging to the set Ψ of possible limit points is exponentially stable.

Lemma 3.6 ([29, Lemma 4.1])**.** *Suppose that Assumption 3.1 holds. There exist constants C_1 and C_2 such that for every solution $q^* : \mathbb{R} \to \mathbb{R}^{|S \times A|}$ belonging to the set Ψ, we have*

$$|q^*(z)| \leq C_1 e^{-C_2 z} |q^*(0)|, \quad \textit{for } z \geq 0.$$

The following intermediary proposition describes the convergence of the Q-FPL algorithm during a single interval. This is not the focal point of this work, but a detailed account appears in ([29, 2]).

Proposition 3.7 (Q-learning PAC Bound)**.** *Suppose that Assumption 3.1 holds. Consider an arbitrary interval α_m during which the learning rate is γ_m. Let $1, \ldots, K_m$ denote the time steps of α_m. Suppose that C_1 and C_2 are as defined in Lemma 3.6. If $\gamma_m \in (0, 1/4)$ and $K_m \geq \frac{\log(2C_1)}{C_2 \gamma_m}$, then for every $\xi > 0$,*

$$\Pr\left(\inf_{Q^* \in \Psi} \|Q_{K_m} - Q^*\|_\infty > \xi\right) \leq \frac{16}{\xi^2} \gamma_m.$$

Remark 23. Proposition 3.7 requires that the length K_m of each interval α_m increases as $\gamma_m \searrow 0$. A small value of γ_m entails a tighter bound, but also slower convergence.

Proof. First, observe that since the functions h_t are continuous (in the argument Q) and non-expansive for all t:

$$\|h_t(Q) - h_t(Q')\|_\infty \leq \|Q - Q'\|_\infty .$$

Hence, the stochastic approximation sequence Q_t is stable without additional constraints.

Observe that $h_t(Q) \in G(Q)$ for all t, so that we have

$$\lim_{n,m\to\infty} \operatorname{dist}\left(\frac{1}{m}\sum_{i=n}^{n+m-1} h_i(Q)\ ,\ G(Q)\right) = 0.$$

Therefore, by [79, Theorem 5.6.3], with probability 1, the limit points of Q_t are contained in the invariant set Ψ.

By [29, Theorem 2.1], there exists a constant $\gamma^* = 1/4$ such that if $\gamma_m \in (0, \gamma^*)$, then

$$\limsup_{K\to\infty} \mathbb{E}\,\|Q_K\|_\infty^2 < \infty.$$

Let $\tau^*(K) = \sum_{j=1}^K \gamma_m = K\gamma_m$. Since $K \geq \frac{\log(2C_1)}{C_2\gamma_m}$ by assumption, we obtain

$$C_1 e^{-C_2\tau^*(K)} \leq 1/2.$$

Finally, by taking $\gamma_m \in (0, \gamma^*)$, the assumptions of [29, Theorem 2.3] are satisfied, and the claimed result follows from it. ∎

The following theorem is the main result of this section.

Theorem 3.8 (PAC regret bound of Q-FPL)**.** *Suppose that Assumptions 3.1 and 3.2, and the the assumptions of Proposition 3.7 hold. Suppose that there*

exists $x \in (0, 1/3)$ *such that* $K_m = O(m^x)$ *for every* m. *Let* $M(T)$ *denote the number of intervals up to time step* T. *Then, for every* $\xi > 0$ *and every* $T \geq 2e\tau/\epsilon$, *the regret of the Q-FPL algorithm with respect to every sequence* $r_1, r_2, \ldots$ *and every sequence* $P_1, P_2, \ldots$ *is bounded as follows:*

$$\widehat{J}_T - \frac{1}{T}\sum_{t=1}^{T} r_t(s_t, a_t) \leq (Z+2)\epsilon + \xi + R(T) \quad \textit{with probability } 1 - \frac{16}{\xi^2}\sum_{m=1}^{M(T)} \gamma_m,$$

where $R(T)$ *is of the order of* $O(T^{-1/4}\log(T))$.

Remark 24. By setting $\xi = \epsilon$ and $\gamma_m = \frac{\epsilon^3}{16M(T)}$ for all m, Theorem 3.8 gives the following PAC bound:

$$\widehat{J}_T - \frac{1}{T}\sum_{t=1}^{T} r_t(s_t, a_t) \leq (Z+3)\epsilon + O\left(\frac{\log(T)}{T^{1/4}}\right) \quad \text{with probability } 1 - \epsilon.$$

Observe that we may also let $\gamma_m \searrow 0$ and $K_m \to \infty$ as $m \to \infty$.

Proof. Using Lemma 3.2, our task is to bound the regret with respect to the relaxed objective $\widehat{I}_T$. We first bound the distance, at the end of each interval α_m, between the Q-function Q_t generated by the Q-FPL algorithm and the limit Q^δ of the sequence $\{Q_t^\delta\}$. The claimed result then follows from the bound on the regret associated with the policy induced by Q^δ.

Observe that Q^δ is a limit point belonging to this invariant set Ψ. Observe also that by definition and Assumption 3.2, for every pair $h(Q) \in G(Q)$ and $h'(Q) \in G(Q)$, we have

$$\|h(Q) - h'(Q)\|_\infty \leq \epsilon.$$

Hence, for every limit point Q^* belonging to the invariant set Ψ, we have

$$\left\|Q^\delta - Q^*\right\|_\infty \leq \epsilon.$$

By Proposition 3.7, we obtain

$$\Pr\left(\left\|Q_{t_m+K_m} - Q^\delta\right\|_\infty > \epsilon + \xi\right) \leq \frac{16}{\xi^2}\gamma_m.$$

Let σ_m denote the policy generated by $Q_{t_m+K_m}$, and σ^δ denote the policy induced by Q^δ—the optimal policy in a standard MDP with transition function P^δ. By the definition of the Q-functions, it follows that, with probability $1 - \frac{16}{\xi^2}\gamma_m$,

$$\begin{aligned}\frac{1}{K_m}\sum_{t\in\alpha_m} \widehat{r}_{t_m}(s_t, \sigma_m(s_t)) &\geq \frac{1}{K_m}\sum_{t\in\alpha_m} \widehat{r}_{t_m}(\widehat{s}_t, \sigma^\delta(\widehat{s}_t)) - \epsilon - \xi, \\ &\geq \max_{\mu:S\to A} \frac{1}{K_m}\sum_{t\in\alpha_m} \widehat{r}_{t_m}(\tilde{s}_t, \mu(\tilde{s}_t)) - \epsilon - \xi, \qquad (3.6.12)\end{aligned}$$

where s_t is the true state trajectory, whereas $\widehat{s}_t$ and $\tilde{s}_t$ are induced by agent policies σ^δ and μ with the fixed transition function P^δ. The second inequality is due to the fact that σ^δ is an optimal policy when the transition functions are fixed to $P_t = P^\delta$ for all t during interval α_m.

Finally, by Proposition 2.3, while the transition function is fixed, the expected average regret due to the arbitrary reward sequence r_t is bounded by

$$R(T) = [16 + 3(|S| + 7)\,|A|^2\,\tau\log(T)]T^{-1/4}.$$

The claim follows by combining the above bound with Lemma 3.2 and (3.6.12).

■

3.6.3 Expected Regret Bound

In this section, we show a bound on the expected regret of the Q-FPL algorithm.

Theorem 3.9 (Expected regret of Q-FPL)**.** *Suppose that Assumptions 3.1 and 3.2 hold. Further, suppose that* $0 \leq \gamma_m \leq \min(1/4, \epsilon^3/16)$ *and* $K_m =$

$\frac{\log(2C_1)}{C_2\gamma_m}$ *for all* m*. Then, for* $T \geq 2e\tau/\epsilon$*, the regret of the Q-FPL algorithm with respect to every sequence* $r_1, r_2, \ldots$ *and every sequence* $P_1, P_2, \ldots$ *is bounded as follows:*

$$\widehat{J}_T - \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\big[r_t(s_t, a_t)\big] \leq \big(Z + \log(2C_1)/C_2 + 3\big)\epsilon + R(T),$$

where $R(T)$ *is of the order of* $O(T^{-1/4}\log(T))$*.*

Remark 25. Compared to the regret bound for the ORDP algorithm (cf. Theorem 3.1), the asymptotic regret has a larger leading factor.

Proof. Using Lemma 3.2, our task is to bound the regret with respect to the relaxed objective $\widehat{I}_T$. In Step 1, we show that, at the end of each interval α_m, the Q-function generated by the Q-FPL algorithm is within a distance of ϵ (in expectation) from the limit Q^δ of the sequence $\{Q_t^\delta\}$. In Step 2, we apply a bound on the regret associated with the policy induced by Q^δ.

(Step 1.) We define $\mathbf{P}_j^a$ as the matrix whose (s', s)-element is $P_j(s' \mid s, a)$, and $\mathbf{P}^{\delta,a}$ as the matrix whose (s', s)-element is $P^\delta(s' \mid s, a)$. Let the norm $\|M\|_\infty$ of a matrix M denote its maximum absolute row-sum. For simplicity, let $1, \ldots, K+1$ denote the set of time steps of interval α_m. Let $(s, a) \in S \times A$

be an arbitrary state-action pair. Observe that

$$
\begin{aligned}
&\left|\mathbb{E}\, Q_{K+1}(s,a) - \mathbb{E}\, Q^{\delta}_{K+1}(s,a)\right| \\
&\leq (1-\gamma_m)\left|\mathbb{E}\, Q_K(s,a) - \mathbb{E}\, Q^{\delta}_K(s,a)\right| + \gamma_m \left|\mathbb{E}\, Q_K(s_0,a_0) - \mathbb{E}\, Q^{\delta}_K(s_0,a_0)\right| \\
&+ \gamma_m \left| \mathbb{E} \sum_{s' \in S} P_K(s' \mid s,a) \max_{a' \in A} Q_K(s',a') - \mathbb{E} \sum_{s'' \in S} P^{\delta}(s'' \mid s,a) \max_{a'' \in A} Q^{\delta}_K(s'',a'') \right| \\
&\leq (1-\gamma_m)\left|\mathbb{E}\, Q_K(s,a) - \mathbb{E}\, Q^{\delta}_K(s,a)\right| + \gamma_m \left\|\mathbf{P}^{\delta,a} - \mathbf{P}^{a}_K\right\|_\infty \left\|\mathbb{E}\, Q^{\delta}_K\right\|_\infty \\
&\quad + \gamma_m \max_{s' \in S} \left| \mathbb{E} \max_{a' \in A} Q_K(s',a') - \mathbb{E} \max_{a'' \in A} Q^{\delta}_K(s',a'') \right| \left\|\mathbf{P}^{\delta,a}\right\|_\infty \\
&\leq (1-\gamma_m)\left|\mathbb{E}\, Q_K(s,a) - \mathbb{E}\, Q^{\delta}_K(s,a)\right| + \gamma_m \left\|\mathbf{P}^{\delta,a} - \mathbf{P}^{a}_K\right\|_\infty 1 \\
&\quad + \gamma_m \max_{s' \in S} \max_{a' \in A} \left|\mathbb{E}\, Q_K(s',a') - \mathbb{E}\, Q^{\delta}_K(s',a')\right| \qquad (3.6.13) \\
&\leq (1-\gamma_m)\left|\mathbb{E}\, Q_K(s,a) - \mathbb{E}\, Q^{\delta}_K(s,a)\right| + \gamma_m \left\|\mathbf{P}^{\delta,a} - \mathbf{P}^{a}_K\right\|_\infty \\
&\quad + \gamma_m \left|\mathbb{E}\, Q_K(s^*,a^*) - \mathbb{E}\, Q^{\delta}_K(s^*,a^*)\right| \\
&\leq \left|\mathbb{E}\, Q_K(s^*,a^*) - \mathbb{E}\, Q^{\delta}_K(s^*,a^*)\right| + \gamma_m \left\|\mathbf{P}^{\delta,a} - \mathbf{P}^{a}_K\right\|_\infty, \qquad (3.6.14)
\end{aligned}
$$

where s^* and a^* denote the solutions to the maximizations of (3.6.13). The first inequality is due to taking expectation over the next-state variables. For the second inequality, we use the triangle inequality and assume without loss of generality that there exists a state-action pair (s_0, a_0) such that $\mathbb{E}\, Q_t(s_0,a_0) = \mathbb{E}\, Q^{\delta}_t(s_0,a_0)$ for all t. The third inequality follows by algebra and the fact that $\left\|\mathbb{E}\, Q^{\delta}_K\right\|_\infty \leq 1$ since the rewards take values in the interval $[0,1]$. For the final inequality, we assume without loss of generality that $\left|\mathbb{E}\, Q_K(s,a) - \mathbb{E}\, Q^{\delta}_K(s,a)\right| \leq \left|\mathbb{E}\, Q_K(s^*,a^*) - \mathbb{E}\, Q^{\delta}_K(s^*,a^*)\right|$; otherwise, we interchange (s,a) and (s^*,a^*). By carrying on the recursion of (3.6.14), we obtain

$$
\left|\mathbb{E}\, Q_{K+1}(s,a) - \mathbb{E}\, Q^{\delta}_{K+1}(s,a)\right| \leq \sum_{j \in \alpha_m} \gamma_m \left\|\mathbf{P}^{\delta,a} - \mathbf{P}^{a}_j\right\|_\infty.
$$

By the ϵ-arbitrariness assumption (Assumption 3.2), there exists a fixed $\epsilon > 0$ such that $\|\mathbf{P}_t^a - \mathbf{P}_{t'}^a\|_\infty < \epsilon$ for every pair of time instants t and t'. Therefore, there exists a $\delta \in \Delta(C)$ such that $\left\|\mathbf{P}^{\delta,a} - \mathbf{P}_j^a\right\|_\infty < \epsilon$, for all $j \in \alpha_m$. Moreover, we have $\sup_{m=1,2,\ldots} K_m \gamma_m \geq \sum_{j \in \alpha_m} \gamma_m$ by definition. Hence, we obtain

$$\left\|\mathbb{E}\, Q_{K+1} - \mathbb{E}\, Q_{K+1}^\delta\right\|_\infty \leq \frac{\log(2C_1)}{C_2}\epsilon.$$

The sequence Q_t^δ converges to a limit that we denote Q^δ. By Jensen's Inequality, Chebyshev's Inequality and Proposition 3.7, we obtain

$$\begin{aligned}
\left\|\mathbb{E}\, Q_{K+1}^\delta - Q^\delta\right\|_\infty &\leq \mathbb{E}\left\|Q_{K+1}^\delta - Q^\delta\right\|_\infty \\
&\leq \epsilon \Pr(\left\|Q_{K+1}^\delta - Q^\delta\right\|_\infty \leq \epsilon) + \Pr(\left\|Q_{K+1}^\delta - Q^\delta\right\|_\infty > \epsilon) \\
&\leq \epsilon + \frac{16}{\epsilon^2}\gamma_m \leq 2\epsilon,
\end{aligned}$$

where the final equality follows by the assumption that $\gamma_m \leq \epsilon^3/16$. It follows that

$$\begin{aligned}
\left\|\mathbb{E}\, Q_{K+1} - Q^\delta\right\|_\infty &\leq \left\|\mathbb{E}\, Q_{K+1} - \mathbb{E}\, Q_{K+1}^\delta\right\|_\infty + \left\|\mathbb{E}\, Q_{K+1}^\delta - Q^\delta\right\|_\infty \\
&\leq (\log(2C_1)/C_2 + 2)\epsilon. \qquad (3.6.15)
\end{aligned}$$

(Step 2.) Let σ_K denote the policy generated by Q_{K+1}, and σ^δ denote the policy generated by Q^δ. As shown in [29], σ^δ is an optimal policy when the transition functions are fixed to $P_t = P^\delta$ for all t during interval α_m. From (3.6.15), it follows that the average reward of policy σ_K differs from the

optimal average reward as follows:

$$\begin{aligned}
&\frac{1}{K_m}\sum_{t\in\alpha_m}\mathbb{E}\,\widehat{r}_{t_m}(s_t,\sigma_K(s_t))\\
&\geq \frac{1}{K_m}\sum_{t\in\alpha_m}\mathbb{E}\,\widehat{r}_{t_m}(\widehat{s}_t,\sigma^{\delta}(\widehat{s}_t)) - (\log(2C_1)/C_2+2)\epsilon,\\
&\geq \max_{\mu:S\to A}\frac{1}{K_m}\sum_{t\in\alpha_m}\mathbb{E}\,\widehat{r}_{t_m}(\tilde{s}_t,\mu(\tilde{s}_t)) - (\log(2C_1)/C_2+2)\epsilon,
\end{aligned}$$

where s_t is the true state trajectory, whereas $\widehat{s}_t$ and $\tilde{s}_t$ are induced by agent policies σ^{δ} and μ with the fixed transition function P^{δ}. The rest of the proof follows the same line as the proof of Theorem 3.8 ■

3.7 Discussion

It is useful to compare the ORDP and Q-FPL algorithms. The ORDP algorithm requires not only knowledge of the transition probabilities $P^c(s' \mid s,a)$—for all c, s', s, and a, but also of the uncertainty set $\mathcal{D}$ of the modulating sequence $\{\delta_t\}$. The Q-FPL algorithm requires only knowledge of the bound ϵ of Assumption 3.2. Moreover, the Q-function iteration of the Q-FPL algorithm is computationally more efficient than solving the robust dynamic program of the ORDP algorithm. The only drawback is in the weaker bound on the regret.

Our results clearly extend to the case where the transition functions are ϵ-arbitrary over subintervals of the entire time horizon, provided that the agent estimates the modulating term δ_t at each time t and resets the Q-learning sub-algorithm when a sufficiently large shift in the modulating sequence is detected. The uncertainty set $\mathcal{D}$ of the ORDP algorithm can also be regularly updated to reflect the estimates $\{\widehat{\delta}_t\}$. Effectively, this allows us to replace the ϵ-arbitrariness assumption of Theorem 3.1 by a more general assumption on the accuracy $|\widehat{\delta}_t - \delta_t|$ of each estimate $\widehat{\delta}_t$.

CHAPTER 4
Online Learning for Sequential Optimization with Constraints

4.1 Introduction

In this chapter, we study online learning when the objective of the decision maker is to maximize her long-term average reward subject to certain sample path average constraints. We define the reward-in-hindsight as the highest reward the decision maker could have achieved, while satisfying the constraints, had she known Nature's choices in advance. We show that in general the reward-in-hindsight is *not* attainable. The convex hull of the reward-in-hindsight function is, however, attainable. For the important case of a single constraint, the convex hull turns out to be the highest attainable function. Using a calibrated forecasting rule, we provide an explicit strategy that attains this convex hull. We also measure the performance of heuristic methods based on non-calibrated forecasters in experiments involving a CPU power management problem.

The model of this chapter is a repeated game from the viewpoint of a decision maker (player P1) who plays against Nature (player P2). The opponent (Nature) is "arbitrary" in the sense that player P1 has no prediction, statistical or strategic, of the opponent's choice of actions. This setting was considered by [64], in the context of repeated matrix games. Hannan introduced the Bayes utility with respect to the current empirical distribution of the opponent's actions, as a performance goal for adaptive play. This quantity, defined as the highest average reward that player P1 could achieve, in hindsight, by playing some fixed action against the observed action sequence of player P2. Player P1's *regret* is defined as the difference between the highest average

reward-in-hindsight that player P1 could have hypothetically achieved, and the actual average reward obtained by player P1. It was established in [64] that there exist strategies whose regret converges to zero as the number of stages increases, even in the absence of any prior knowledge on the strategy of player P2. For recent advances on online learning, see [34].

The objective in this model is to minimize the regret under sample-path constraints. That is, in addition to maximizing the average reward, or more precisely, minimizing the regret, the decision maker has some side constraints that need to be satisfied on the average. In particular, for every joint action of the players, there is an additional penalty vector that is accumulated by the decision maker. The decision maker has a predefined set in the space of penalty vectors, which represents the acceptable trade-offs between the different components of the penalty vector. An important special case arises when the decision maker wishes to keep some constrained resource below a certain threshold. Consider, for example, a wireless communication system where the decision maker can adjust the transmission power to improve the probability that a message is received successfully. Of course, the decision maker does not know a priori how much power will be needed (this depends on the behaviour of other users, the channel conditions, etc.). Still, a decision maker is usually interested in both the rate of successful transmissions, and in the average power consumption. In an often considered variation of this problem, the decision maker wishes to maximize the transmission rate, while keeping the average power consumption below some predefined threshold. We refer the reader to [92] and references therein for a discussion of constrained average cost stochastic games and to [5] for constrained Markov decision problems.

This chapter is organized as follows. In Section 4.2, we present formally the basic model, and provide a result that relates attainability with the value

of the game. In Section 4.3, we provide an example where the reward-in-hindsight cannot be attained. In light of this negative result, in Section 4.4 we define the closed convex hull of the reward-in-hindsight, and show that it is attainable. Furthermore, in Section 4.5, we show that when there is a single constraint, this is the maximal attainable objective. In Section 4.6, we provide a simple strategy, based on calibrated forecasting, that attains the closed convex hull. Section 4.7 presents heuristic algorithms derived from an online forecaster, while incorporating strictly enforced constraints. The application of the algorithms of Section 4.7 to a power management domain is presented in Section 4.8.

4.2 Problem Definition

We consider a repeated game against Nature, in which a decision maker tries to maximize her reward, while satisfying some constraints on certain time-averages. The underlying stage game is a game with two players: P1 (the decision maker of interest) and P2 (who represents Nature and is assumed arbitrary). For our purposes, we only need to define rewards and constraints for P1.

A constrained game with respect to a set $\mathcal{T}$ is defined by a tuple $(A, B, \mathbf{R}, \mathbf{C}, \mathcal{T})$ where:

1. A is the set of actions of P1; we will assume $A = \{1, 2, \ldots, |A|\}$.
2. B is the set of actions of P2; we will assume $B = \{1, 2, \ldots, |B|\}$.
3. $\mathbf{R}$ is an $|A| \times |B|$ matrix where the entry $\mathbf{R}(a, b)$ denotes the expected reward obtained by P1, when P1 plays action $a \in A$ and P2 action $b \in B$. The actual rewards obtained at each play of actions a and b are assumed to be i.i.d. random variables, with finite second moments, distributed according to a probability law $\Pr_{\mathbf{R}}(\cdot \mid a, b)$. Furthermore, the reward streams for different pairs (a, b) are statistically independent.

4. $\mathbf{C}$ is an $|A| \times |B|$ matrix, where the entry $\mathbf{C}(a, b)$ denotes the expected d-dimensional penalty vector incurred by P1, when P1 plays action $a \in A$ and P2 action $b \in B$. The actual penalty vectors obtained at each play of actions a and b are assumed to be i.i.d. random variables, with finite second moments, distributed according to a probability law $\Pr_{\mathbf{C}}(\cdot \,|\, a, b)$. Furthermore, the penalty vector streams for different pairs (a, b) are statistically independent.
5. $\mathcal{T}$ is a set in $\mathbb{R}^d$ within which we wish the average of the penalty vectors to lie. We assume that $\mathcal{T}$ is convex and closed. Since the entries of $\mathbf{C}$ are bounded, we will also assume, without loss of generality, that $\mathcal{T}$ is bounded.

The game is played in stages. At each stage t, P1 and P2 simultaneously choose actions $a_t \in A$ and $b_t \in B$, respectively. Player P1 obtains a reward r_t, distributed according to $\Pr_{\mathbf{R}}(\cdot \,|\, a_t, b_t)$, and a penalty c_t, distributed according to $\Pr_{\mathbf{C}}(\cdot \,|\, a_t, b_t)$. We define P1's average reward by time t to be

$$\widehat{r}_t = \frac{1}{t} \sum_{j=1}^{t} r_j, \tag{4.2.1}$$

and P1's average penalty vector by time t to be

$$\widehat{c}_t = \frac{1}{t} \sum_{j=1}^{t} c_j. \tag{4.2.2}$$

A *strategy* for P1 (respectively P2) is a mapping from the set of all possible past histories to the set of mixed actions on A (respectively B), which prescribes the (mixed) action of that player at each time t, as a function of the history in the first $t-1$ stages. Loosely, P1's goal is to maximize the average reward while keeping the average penalty vector in $\mathcal{T}$, path-wise:

$$\limsup_{t \to \infty} \quad \mathrm{dist}(\widehat{c}_t, \mathcal{T}) \to 0, \qquad \text{a.s.}, \tag{4.2.3}$$

where $\text{dist}(\cdot)$ is the point-to-set Euclidean distance, *i.e.*, $\text{dist}(x, \mathcal{T}) = \inf_{y \in \mathcal{T}} \|y - x\|_2$, and the probability measure is the one induced by the policy of P1, the policy of P2, and the randomness in the rewards and penalties.

We will often consider the important special case where $\mathcal{T} = \{c \in \mathbb{R}^d : c \leq c_0\}$, for some given $c_0 \in \mathbb{R}^d$, with the inequality interpreted component-wise. We simply call such a game a constrained game with respect to (a vector) c_0. For that special case, the requirement (4.2.3) is equivalent to:

$$\limsup_{t \to \infty} \quad \widehat{c}_t \leq c_0, \quad \text{a.s..} \tag{4.2.4}$$

For a set D, we will use the notation $\Delta(D)$ to denote the set of all probability measures on D. If D is finite, we will identify $\Delta(D)$ with the set of probability vectors of the same size as D. If D is a subset of Euclidean space, we will assume that it is endowed with the Borel σ-field.

4.2.1 Reward-in-Hindsight

We define $\widehat{q}_t \in \Delta(B)$ as the empirical distribution of P2's actions by time t, that is,

$$\widehat{q}_t(b) = \frac{1}{t} \sum_{j=1}^{t} 1_{\{b_t = b\}}, \quad b \in B. \tag{4.2.5}$$

If P1 knew in advance that $\widehat{q}_t$ will equal q, and if P1 were restricted to using a fixed action, then P1 would pick an optimal response (generally a mixed action) to the mixed action q, subject to the constraints specified by $\mathcal{T}$. In

particular, P1 would solve the convex program[1]

$$\begin{aligned} \max_{p\in\Delta(A)} \quad & \sum_{a,b} p(a)q(b)\mathbf{R}(a,b), \\ \text{s.t.} \quad & \sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \in \mathcal{T}. \end{aligned} \tag{4.2.6}$$

By playing a p that solves this convex program, P1 would meet the constraints (up to small fluctuations that are a result of the randomness and the finiteness of t), and would obtain the maximal average reward. We are thus led to define P1's reward-in-hindsight, which we denote by $r^* : \Delta(B) \mapsto \mathbb{R}$, as the optimal objective value in the program (4.2.6), as a function of q.

For the special case of a constrained game with respect to a vector c_0, the convex constraint $\sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \in \mathcal{T}$ is replaced by $\sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \leq c_0$ (the inequality is to be interpreted component-wise).

4.2.2 The Objective

Formally, our goal is to attain a function r in the sense of the following definition. Naturally, the higher the function r, the better.

Definition 4.1. A function $r : \Delta(B) \mapsto \mathbb{R}$ is *attainable* by P1 in a constrained game with respect to a set $\mathcal{T}$ if there exists a strategy σ of P1 such that for every strategy ρ of P2:

(i) $\liminf_{t\to\infty} \quad (\widehat{r}_t - r(\widehat{q}_t)) \geq 0, \quad$ a.s., and

(ii) $\limsup_{t\to\infty} \quad \mathrm{dist}(\widehat{c}_t, \mathcal{T}) \to 0, \quad$ a.s.,

where the almost sure convergence is with respect to the probability measure induced by σ and ρ.

[1] If $\mathcal{T}$ is a polyhedron (specified by finitely many linear inequalities), then the optimization problem is a linear program.

In constrained games with respect to a vector c_0 we can replace (ii) in the definition with

$$\limsup_{t\to\infty} \quad \widehat{c}_t \leq c_0, \qquad \text{a.s.}$$

4.2.3 The Value of the Game

In this section, we consider the attainability of a constant function $r : \Delta(B) \mapsto \mathbb{R}$, *i.e.*, $r(q) = \alpha$, for all q. We will establish that attainability is equivalent to having $\alpha \leq \mathrm{v}$, where v is a naturally defined "value of the constrained game."

We first introduce the assumption that P1 is always able to satisfy the constraint.

Assumption 4.1. For every mixed action $q \in \Delta(B)$ of P2, there exists a mixed action $p \in \Delta(A)$ of P1, such that:

$$\sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \in \mathcal{T}. \tag{4.2.7}$$

For constrained games with respect to a vector c_0, the condition (4.2.7) reduces to the inequality $\sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \leq c_0$.

If Assumption 4.1 is not satisfied, then P2 can choose a q such that for every (mixed) action of P1, the constraint is violated in expectation. By repeatedly playing this q, P1's average penalty vector will be outside $\mathcal{T}$, and the objectives of P1 will be impossible to meet.

The following result deals with the attainability of the value, v, of an average reward repeated constrained game, defined by

$$\mathrm{v} = \inf_{q\in\Delta(B)} \quad \sup_{p\in\Delta(A):\sum_{a,b} p(a)q(b)\mathbf{C}(a,b)\in\mathcal{T}} \sum_{a,b} p(a)q(b)\mathbf{R}(a,b). \tag{4.2.8}$$

The existence of a strategy for P1 that attains the value was proved in [117] in the broader context of stochastic games.

Proposition 4.1. *Suppose that Assumption 4.1 holds. Then,*

(i) P1 has a strategy that guarantees that the constant function $r(q) \equiv \mathrm{v}$ *is attained with respect to* $\mathcal{T}$*.*

(ii) For every number $\mathrm{v}' > \mathrm{v}$ *there exists* $\delta > 0$ *such that P2 has a strategy that guarantees that either* $\liminf_{t\to\infty} \widehat{r}_t < \mathrm{v}' - \delta$ *or* $\limsup_{t\to\infty} \mathrm{dist}(\widehat{c}_t, \mathcal{T}) > \delta$*, almost surely. (In particular, the constant function* v' *is not attainable.)*

Proof. The proof relies on Blackwell's approachability theory [24]. We construct a nested family of convex sets in $\mathbb{R}^{d+1}$ defined by $S_\alpha = \{(r, c) \in \mathbb{R} \times \mathbb{R}^d : r \geq \alpha, c \in \mathcal{T}\}$. Obviously, $S_\alpha \subset S_\beta$ for $\alpha > \beta$. Consider the vector-valued game in $\mathbb{R}^{d+1}$ associated with the constrained game. In this game, P1's vector-valued payoff at time t is the $d+1$ dimensional vector $m_t = (r_t, c_t)$ and P1's average vector-valued payoff is $\widehat{m}_t = (\widehat{r}_t, \widehat{c}_t)$. Since S_α is convex, it follows from approachability theory for convex sets [24] that each S_α is either approachable or excludable. If S_α is approachable, then S_β is approachable for every $\beta < \alpha$. We define $\mathrm{v}_0 = \sup\{\beta \mid S_\beta \text{ is approachable}\}$. It follows that S_{v_0} is approachable (as the limit of approachable sets; see [120]). By Blackwell's theorem, for every $q \in \Delta(B)$, an approachable convex set must intersect the set of feasible payoff vectors when P2 plays q. Using this fact, it is easily shown that v_0 equals v, as defined by (4.2.8)), and part (i) follows. Part (ii) follows because a convex set which is not approachable is excludable. ∎

Note that part (ii) of the proposition implies that, essentially, v is the highest average reward P1 can attain while satisfying the constraints, if P2 plays an adversarial strategy. By comparing (4.2.8) with (4.2.6), we see that $\mathrm{v} = \inf_q r^*(q)$. On the other hand, if P2 does not play adversarially, P1 may

be able to do better, perhaps attaining $r^*(q)$. Our subsequent results address the question whether this is indeed the case.

Remark 26. It can be shown that in order to attain the value of the game, P1 may have to use a non-stationary strategy. This is in contrast to standard (non-constrained) games, in which P1 always has a stationary strategy that attains the value of the game.

Remark 27. In general, the infimum and supremum in (4.2.8) *cannot* be interchanged. This is because the set of feasible p in the inner maximization depends on the value of q. Moreover, it can be shown that the set of (p, q) pairs that satisfy the constraint $\sum_{a,b} p(a)q(b)\mathbf{C}(a, b) \in \mathcal{T}$ is not necessarily convex.

4.2.4 Related Works

Notwithstanding the apparent similarity, the problem that we consider is not an instance of online convex optimization [133, 67]. In the latter setting, there is a convex feasible domain $\mathcal{F} \subset \mathbb{R}^n$, and an arbitrary sequence of convex functions $f_j : \mathcal{F} \to \mathbb{R}$. At every step j, the decision maker picks $x_j \in \mathcal{F}$ based on the past history, without knowledge of the future functions f_j, and with the objective of minimizing the regret

$$\sum_{j=1}^{t} f_j(x_j) - \min_{y \in \mathcal{F}} \sum_{j=1}^{t} f_j(y). \tag{4.2.9}$$

An analogy with our setting might be possible, by identifying x_j and f_j with a_j and b_j, respectively, and by somehow relating the feasibility constraints described by $\mathcal{F}$ to our constraints. However, this attempt seems to run into some fundamental obstacles. In particular, in our setting, feasibility is affected by the opponent's actions, whereas in online convex optimization, the set $\mathcal{F}$ is fixed a priori. For this reason, we do not see a way to reduce the problem

of online learning with constraints to an online convex optimization problem, and given the results below, it is unlikely that such a reduction is possible.

4.3 Reward-in-Hindsight Is Not Attainable

As it turns out, the reward-in-hindsight cannot be attained in general. This is demonstrated by the following simple 2×2 matrix game, with just a single constraint.

Consider a 2×2 constrained game specified by

$$\begin{pmatrix} (1,-1) & (1,1) \\ (0,-1) & (-1,-1) \end{pmatrix},$$

where each entry (pair) corresponds to $(\mathbf{R}(a,b), \mathbf{C}(a,b))$ for a pair of actions a and b. At a typical stage, P1 chooses a row, and P2 chooses a column. We set $c_0 = 0$. Let q denote the frequency with which P2 chooses the second column. The reward of the first row dominates the reward of the second one, so if the constraint can be satisfied, P1 would prefer to choose the first row. This can be done as long as $0 \leq q \leq 1/2$, in which case $r^*(q) = 1$. For $1/2 \leq q \leq 1$, player P1 needs to optimize the reward subject to the constraint. Given a specific q, P1 will try to choose a mixed action that satisfies the constraint (on the average) while maximizing the reward. If we let α denote the frequency of choosing the first row, we see that the reward and penalty are:

$$r(\alpha, q) = \alpha - (1-\alpha)q\,, \quad c(\alpha, q) = 2\alpha q - 1,$$

respectively. We observe that for every q, $r(\alpha)$ and $c(\alpha)$ are monotonically increasing functions of α. As a result, P1 will choose the maximal α that satisfies $c(\alpha) \leq 0$, which is $\alpha(q) = 1/(2q)$, and the optimal reward is $1/2 +$

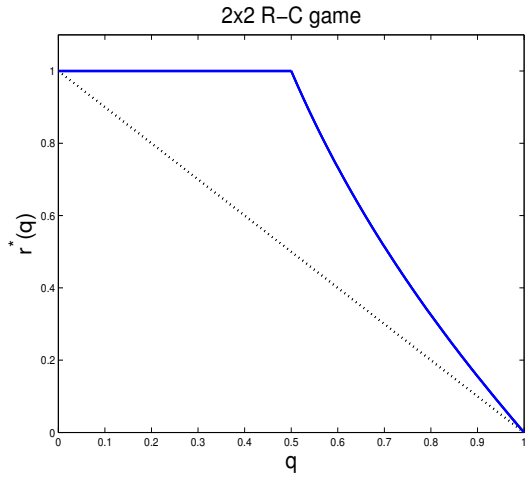


Figure 4–1: The reward-in-hindsight of the constrained game. Here, $r^*(q)$ is the solid line, and the dotted line connects the two extreme values, for $q = 0$ and $q = 1$.

$1/(2q) - q$. We conclude that the reward-in-hindsight is:

$$r^*(q) = \begin{cases} 1, & \text{if } 0 \le q \le 1/2, \\ \dfrac{1}{2} + \dfrac{1}{2q} - q, & \text{if } 1/2 \le q \le 1. \end{cases}$$

The graph of $r^*(q)$ is the solid line in Figure 4–1.

We now claim that P2 can make sure that P1 does not attain r^*.

Proposition 4.2. *If $c_0 = 0$, then there exists a strategy for P2 such that r^* cannot be attained.*

Proof. Suppose that the opponent, P2, plays according to the following strategy. Initialize k to 1. Set $\epsilon > 0$ be a fixed small constant. Let $\widehat{\alpha}_t$ be the empirical frequency with which P1 chooses the *first* row during the first t time steps. Similarly, let $\widehat{q}_t$ be the empirical frequency with which P2 chooses the *second* column during the first t time steps.

1. While $k \le 1/\epsilon$ or $\widehat{\alpha}_{t-1} > 3/4$, P2 chooses the second column. Set $k := k + 1$.
2. For the next k times, P2 chooses the first column.

3. Reset $k := 1$, and go back to Step 1.

We now show that if

$$\limsup_{t\to\infty} \quad \widehat{c}_t \leq 0, \qquad \text{a.s.}, \tag{4.3.10}$$

then a strict inequality holds for the regret:

$$\liminf_{t\to\infty} \quad (\widehat{r}_t - r^*(\widehat{q}_t)) < 0, \qquad \text{a.s.}$$

Suppose that Step 2 is entered only a finite number of times. Then, after some finite time, P2 keeps choosing the second column, and $\widehat{q}_t$ converges to 1. For P1 to satisfy the constraint $\limsup_{t\to\infty} \widehat{c}_t \leq 0$, we must have $\lim \widehat{\alpha}_t \leq 1/2$. But then, the condition $\widehat{\alpha}_{t-1} > 3/4$ will be eventually violated. This shows that Step 2 is entered an infinite number of times. In particular, there exist infinite sequences t_i and t'_i such that $t_i < t'_i < t_{i+1}$ and (i) if $t_i < t \leq t'_i$, P2 chooses the second column (Step 1); (ii) if $t'_i < t \leq t_{i+1}$, P2 chooses the first column (Step 2).

Note that Steps 1 and 2 last for an equal number of time steps. Thus, we have $\widehat{q}_{t_i} = 1/2$, and $r^*(\widehat{q}_{t_i}) = 1$, for all i. Furthermore, $t_{i+1} - t'_i \leq t'_i$, or $t'_i \geq t_{i+1}/2$. Note that $\widehat{\alpha}_{t'_i} \leq 3/4$, because otherwise P2 would still in Step 1 at time $t'_i + 1$. Thus, during the first t_{i+1} time steps, P1 has played the first row at most

$$3t'_i/4 + (t_{i+1} - t'_i) = t_{i+1} - t'_i/4 \leq 7t_{i+1}/8.$$

This implies that $\widehat{r}_{t_{i+1}} \leq 7/8$, and $\liminf_{t\to\infty}(\widehat{r}_t - r^*(\widehat{q}_t)) \leq 7/8 - 1 < 0$. ■

Intuitively, the strategy that was described above allows P2 to force P1 to move, back and forth, between the extreme points ($q = 0$ and $q = 1$) that are linked by the dotted line in Figure 4–1. Since $r^*(q)$ is not convex, and since the dotted line is strictly below $r^*(q)$ for $q = 1/2$, this strategy precludes P1 from attaining $r^*(q)$. We note that the choice of c_0 is critical in this example.

With other choices of c_0 (for example, $c_0 = -1$), the reward-in-hindsight may be attainable.

4.4 Attainability of the Convex Hull

Since the reward-in-hindsight is not attainable in general, we have to settle for a more modest objective. More specifically, we are interested in functions $f : \Delta(B) \to \mathbb{R}$ that are attainable with respect to a given constraint set $\mathcal{T}$. As a target we suggest the closed convex hull of the reward-in-hindsight, r^*. After defining it, we prove that it is indeed attainable. In the next section, we will also show that it is the highest possible attainable function, when there is a single constraint.

Given a function $f : X \mapsto \mathbb{R}$, its *closed convex hull* is the function whose epigraph is

$$\overline{\text{conv}}(\{(x, r) : r \geq f(x)\}),$$

where conv(D) is the convex hull, and $\overline{D}$ is the closure of a set D. We denote the closed convex hull of r^* by r^c.

We will make use of the following facts. Forming the convex hull and then the closure results in a larger epigraph, hence a smaller function. In particular, $r^c(q) \leq r^*(q)$, for all q. Furthermore, the closed convex hull is guaranteed to be continuous on $\Delta(B)$. (This would not be true if we had considered the convex hull, without forming its closure.) Finally, for every q in the interior of $\Delta(B)$, we have:

$$\begin{aligned} r^c(q) \;=\; & \inf_{q_1,q_2,\ldots,q_k \in \Delta(B), \alpha_1,\ldots,\alpha_k} \sum_{i=1}^{k} \alpha_i r^*(q_i) \\ & \text{s.t.} \quad \sum_{i=1}^{k} \alpha_i q_i(b) = q(b), \quad b \in B, \\ & \qquad \alpha_i \geq 0, \quad i = 1, 2, \ldots, k, \\ & \qquad \sum_{i=1}^{k} \alpha_i = 1, \end{aligned} \tag{4.4.11}$$

where k can be taken equal to $|B|+2$ by Caratheodory's Theorem.

The following result is proved using Blackwell's approachability theory. The technique is similar to that used in other no-regret proofs (e.g., [23, 91]), and is based on the convexity of a target set in an appropriately defined space.

Theorem 4.3. *Let Assumption 4.1 hold for a given convex set $\mathcal{T} \subset \mathbb{R}^d$. Then r^c is attainable with respect to $\mathcal{T}$.*

Proof. Define the following game with vector-valued payoffs, where the payoffs belong to $\mathbb{R} \times \mathbb{R}^d \times \Delta(B)$ (a $|B|+d+1$ dimensional space, which we denote by $\mathcal{M}$). Suppose that P1 plays a_t, P2 plays b_t, P1 obtains an immediate reward of r_t and an immediate penalty vector of c_t. Then, the vector-valued payoff obtained by P1 is

$$m_t = (r_t, c_t, e(b_t)),$$

where $e(b)$ is a vector of zeroes, except for a 1 in its bth component. It follows that the average vector-valued reward at time t, which we define as $\widehat{m}_t = \frac{1}{t}\sum_{j=1}^{t} m_j$, satisfies: $\widehat{m}_t = (\widehat{r}_t, \widehat{c}_t, \widehat{q}_t)$, where $\widehat{r}_t$, $\widehat{c}_t$, and $\widehat{q}_t$ were defined in Equations (4.2.1), (4.2.2), and (4.2.5), respectively. Consider the sets:

$$\mathcal{B}_1 = \{(r, c, q) \in \mathcal{M} : \; r \geq r^c(q)\}, \qquad \mathcal{B}_2 = \{(r, c, q) \in \mathcal{M} : \; c \in \mathcal{T}\},$$

and let $\mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2$. Note that $\mathcal{B}$ is a convex set. We claim that $\mathcal{B}$ is approachable. Let $m : \Delta(A) \times \Delta(B) \to \mathcal{M}$ describe the expected payoff in a single stage game, when P1 and P2 choose actions p and q, respectively. That is,

$$m(p, q) = \Big(\sum_{a,b} p(a)q(b)\mathbf{R}(a, b), \; \sum_{a,b} p(a)q(b)\mathbf{C}(a, b), \; q \Big).$$

Using the sufficient condition for approachability of convex sets [24], it suffices to show that for every q there exists a p such that $m(p, q) \in \mathcal{B}$. Fix $q \in \Delta(B)$. By Assumption 4.1, the constraint $\sum_{a,b} p(a)q(b)\mathbf{C}(a, b) \in \mathcal{T}$ is feasible, which

implies that the program (4.2.6) has an optimal solution p^*. It follows that $m(p^*, q) \in \mathcal{B}$. We now claim that a strategy that approaches $\mathcal{B}$ also attains r^c in the sense of Definition 4.1. Indeed, since $\mathcal{B} \subseteq \mathcal{B}_2$ we have that $\Pr(d(c_t, \mathcal{T}) > \epsilon \text{ infinitely often}) = 0$ for every $\epsilon > 0$. Since $\mathcal{B} \subseteq \mathcal{B}_1$ and using the continuity of r^c, we obtain $\liminf (\widehat{r}_t - r^c(\widehat{q}_t)) \geq 0$. ■

Remark 28. Convergence rate results also follow from general approachability theory, and are generally of the order of $t^{-1/3}$; see [99]. It may be possible, perhaps, to improve upon this rate and obtain $t^{-1/2}$, which is the best possible convergence rate for the unconstrained case.

Remark 29. For every $q \in \Delta(B)$, we have $r^*(q) \geq \text{v}$, which implies that $r^c(q) \geq \text{v}$. Thus, attaining r^c guarantees an average reward at least as high as the value of the game.

4.4.1 Degenerate Cases

In this section, we consider the degenerate cases where the penalty vector is affected by only one of the players. We start with the case where P1 alone affects the penalty vector, and then discuss the case where P2 alone affects the penalty vector.

If P1 alone affects the penalty vector, that is, if $\mathbf{C}(a, b) = \mathbf{C}(a, b')$ for all $a \in A$ and $b, b' \in B$, then $r^*(q)$ is convex. Indeed, in this case, (4.2.6) becomes (writing $\mathbf{C}(a)$ for $\mathbf{C}(a, b)$)

$$r^*(q) = \max_{p \in \Delta(A): \sum_a p(a)\mathbf{C}(a) \in \mathcal{T}} \sum_{a,b} p(a)q(b)\mathbf{R}(a, b),$$

which is the maximum of a collection of linear functions of q (one function for each feasible p), and is therefore convex.

If P2 alone affects the penalty vector, that is, if $c(a, b) = c(a', b)$ for all $b \in B$ and $a, a' \in A$, then Assumption 4.1 implies that the constraint is always

satisfied. Therefore,

$$r^*(q) = \max_{p \in \Delta(A)} \sum_{a,b} p(a)q(b)\mathbf{R}(a,b),$$

which is again a maximum of linear functions, hence convex.

We conclude that in both degenerate cases, if Assumption 4.1 holds, then the reward-in-hindsight is attainable.

4.5 Tightness of the Convex Hull

We now show that r^c is the maximal attainable function, for the case of a single constraint.

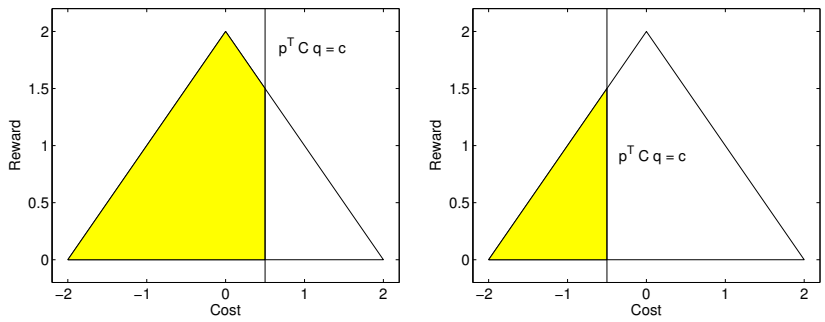
Theorem 4.4. *Suppose that $d = 1$, $\mathcal{T}$ is of the form $\mathcal{T} = \{c \mid c \leq c_0\}$, where c_0 is a given scalar, and that Assumption 4.1 is satisfied. Let $\tilde{r} : \Delta(B) \mapsto \mathbb{R}$ be a continuous attainable function with respect to the scalar c_0. Then, $r^c(q) \geq \tilde{r}(q)$ for all $q \in \Delta(B)$.*

Proof. The proof is constructive, as it provides a concrete strategy for P2 that prevents P1 from attaining $\tilde{r}$, unless $r^c(q) \geq \tilde{r}(q)$ for every q. Assume, in order to derive a contradiction, that there exists some $\tilde{r}$ that violates the theorem. Since $\tilde{r}$ and r^c are continuous, there exists some $q^0 \in \Delta(B)$ and some $\epsilon > 0$ such that $\tilde{r}(q) > r^c(q) + \epsilon$ for all q in an open neighbourhood of q^0. In particular, q^0 can be taken to lie in the interior of $\Delta(B)$. Using (4.4.11), it follows that there exist $q^1, \ldots, q^k \in \Delta(B)$ and $\alpha_1, \ldots, \alpha_k$ (with $k \leq |B| + 2$, due to Caratheodory's Theorem) such that

$$\sum_{i=1}^{k} \alpha_i r^*(q^i) \leq r^c(q^0) + \frac{\epsilon}{2} < \tilde{r}(q^0) - \frac{\epsilon}{2};$$

$$\sum_{i=1}^{k} \alpha_i q^i(b) = q^0(b), \quad \forall\, b \in B; \qquad \sum_{i=1}^{k} \alpha_i = 1; \qquad \alpha_i \geq 0,\ \forall\, i.$$

Let j be a large positive integer (j is to be chosen large enough to ensure that the events of interest occur with high probability, etc.). We will show



(a) (b)

Figure 4–2: For a fixed q, the triangular area shown is the set $M(q)$. Furthermore, the shaded region corresponds to reward and cost pairs (r, c) associated with feasible vectors p; cf. Equations (4.5.12) and (4.5.13).

that if P2 plays each q^i for $\alpha_i j$ time steps, in an appropriate order, then either P1 does not satisfy the constraint along the way or $\widehat{r}_j \leq \tilde{r}(\widehat{q}_j) - \epsilon/2$.

We let q^i, $i = 1, \ldots, k$, be fixed, as above, and define a function $f_i : \mathbb{R}^d \to \mathbb{R} \cup \{-\infty\}$ as:

$$f_i(c) = \max_{p \in \Delta(A)} \sum_{a,b} p(a) q^i(b) \mathbf{R}(a,b), \tag{4.5.12}$$

$$\text{subject to} \qquad \sum_{a,b} p(a) q^i(b) \mathbf{C}(a,b) \leq c, \tag{4.5.13}$$

where the maximum over an empty set is defined to equal $-\infty$. Observe that the feasible set (and hence, optimal value) of the above linear program depends on c, as illustrated in Figure 4–2. By viewing Equations (4.5.12)-(4.5.13) as a parametric linear program, with a varying right-hand side parameter c, we see that $f_i(c)$ is piecewise linear, concave, and non-decreasing in c [22]. Furthermore, $f_i(c_0) = r^*(q^i)$. Let f_i^+ be the right directional derivative of f_i at $c = c_0$, note that $f_i^+ \geq 0$. From now on, we assume that the q^i have been ordered so that the sequence f_i^+ is non-increasing (e.g., as in Figure 4–3). To visualize the ordering that we have introduced, consider the set of possible pairs (r, c), given a fixed q. That is, consider the set $M(q^i) = \{(r, c) : \exists p \in \Delta(A) \text{ s.t. } r = \sum_{a,b} p(a) q^i(b) \mathbf{R}(a,b),\ c = \sum_{a,b} p(a) q^i(b) \mathbf{C}(a,b)\}$. The

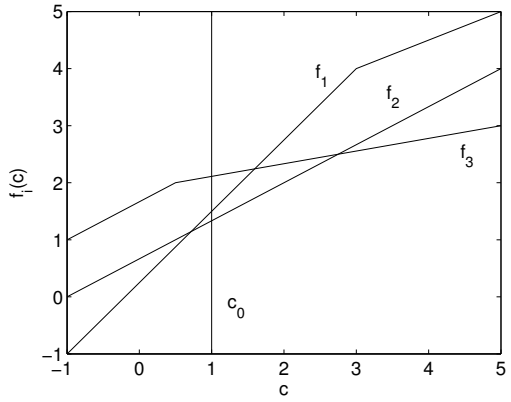


Figure 4–3: An example of functions f_1, f_2, and f_3 ordered according to f_i^+.

set $M(q^i)$ is the image of the simplex under a linear transformation, and is therefore a polytope. (In Figure 4–2 we show two such sets.) The strategy of P2 is to first play q^i such that the p that maximizes the reward (4.5.12)) satisfies (4.5.13) with equality. (Such a q^i results in a set $M(q^i)$ like the one shown in Figure 4–2(b).) The ordering of such q^i is set according to their slope at c (*i.e.*, f_i^+). After all these q^i are played, P2 plays those q^i for which the p that maximizes the reward (4.5.12)) satisfies (4.5.13) with strict inequality (and $f_i^+ = 0$). (Such a q^i results in a set $M(q^i)$ like the one shown in Figure 4–2(a).)

Suppose that P1 knows the sequence $q^1, \ldots, q^k$ (ordered as above) in advance, and that P2 follows the strategy described earlier. We assume that j is large enough so that we can ignore the effects of dealing with a finite sample, or of $\alpha_i j$ not being an integer. Let p^i be the average of the mixed actions chosen by P1 while player P2 plays q^i. We introduce the constraints

$$\sum_{i=1}^{\ell} \alpha_i \sum_{a,b} p^i(a) q^i(b) \mathbf{C}(a,b) \leq c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1, 2, \ldots, k.$$

These constraints must be satisfied in order to guarantee that $\widehat{c}_t$ has negligible probability of substantially exceeding c_0, at the "switching" times from one mixed action to another. If P1 exploits the knowledge of P2's strategy to

maximize her average reward at time j, the resulting expected average reward at time j will be the optimal value of the objective function in the following linear programming problem:

$$\begin{aligned} \max_{p^1,p^2,\ldots,p^k} \quad & \sum_{i=1}^{k} \alpha_i \sum_{a,b} p^i(a)q^i(b)\mathbf{R}(a,b) \\ \text{s.t.} \quad & \sum_{i=1}^{\ell} \alpha_i \sum_{a,b} p^i(a)q^i(b)\mathbf{C}(a,b) \le c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1,2,\ldots,k, \\ & p^\ell \in \Delta(A), \quad \ell = 1,2,\ldots,k. \end{aligned} \tag{4.5.14}$$

Of course, given the value of $\sum_{a,b} p^i(a)q^i(b)\mathbf{C}(a,b)$, to be denoted by c_i, player P1 should choose a p^i that maximizes rewards, resulting in $\sum_{a,b} p^i(a)q^i(b)\mathbf{R}(a,b) = f_i(c_i)$. Thus, the above problem can be rewritten as

$$\begin{aligned} \max_{c_1,\ldots,c_k} \quad & \sum \alpha_i f_i(c_i) \\ \text{s.t.} \quad & \sum_{i=1}^{\ell} \alpha_i c_i \le c_0 \sum_{i=1}^{\ell} \alpha_i, \quad \ell = 1,2,\ldots,k. \end{aligned} \tag{4.5.15}$$

We claim that letting $c_i = c_0$, for all i, is an optimal solution to the problem (4.5.15). This will then imply that the optimal value of the objective function for the problem (4.5.14) is $\sum_{i=1}^{k} \alpha_i f_i(c_0)$, which equals $\sum_{i=1}^{k} \alpha_i r^*(q^i)$, which in turn, is bounded above by $\tilde{r}(q^0) - \epsilon/2$. Thus, $\widehat{r}_j < \tilde{r}(q^0) - \epsilon/2 + \delta(j)$, where the term $\delta(j)$ incorporates the effects due to the randomness in the process. By repeating this argument with ever increasing values of j (so that the stochastic term $\delta(j)$ is averaged out and becomes negligible), we obtain that the event $\widehat{r}_t < \tilde{r}(q^0) - \epsilon/2$ will occur infinitely often, and therefore $\tilde{r}$ is not attainable.

It remains to establish the claimed optimality of $(c_0, \ldots, c_0)$. Suppose that $(\overline{c}_1, \ldots, \overline{c}_k) \neq (c_0, \ldots, c_0)$ is an optimal solution of the problem (4.5.15). If $\overline{c}_i \le c_0$ for all i, the monotonicity of the f_i implies that $(c_0, \ldots, c_0)$ is also an optimal

solution. Otherwise, let j be the smallest index for which $\overline{c}_j > c_0$. If $f_j^+ = 0$ (as in the case shown in Figure 4–2(b)) we have that $f_i(c)$ is maximized at c_0 for all $i \geq j$ and $(c_0, \ldots, c_0)$ optimal. Suppose that $f_j^+ > 0$. In order for the constraint (4.5.15) to be satisfied, there must exist some index $s < j$ such that $\overline{c}_s < c_0$. Let us perturb this solution by setting $\delta = \min\{\alpha_s(c_0 - \overline{c}_s), \alpha_j(\overline{c}_j - c_0)\}$, increasing $\overline{c}_s$ to $\tilde{c}_s = \overline{c}_s + \delta/\alpha_s$, and decreasing $\overline{c}_j$ to $\tilde{c}_j = \overline{c}_j - \delta/\alpha_j$. This new solution is clearly feasible. Let $f_s^- = \lim_{\epsilon \downarrow 0}(f_s(c_0) - f_s(c_0 - \epsilon))$, which is the left derivative of f_s at c_0. Using the concavity of f_s, and the earlier introduced ordering, we have $f_s^- \geq f_s^+ \geq f_j^+$, from which it follows easily (the detailed argument is omitted) that $f_s(\tilde{c}_s) + f_j(\tilde{c}_j) \geq f_s(\overline{c}_s) + f_j(\overline{c}_j)$. Therefore, the new solution must also be optimal, but has fewer components that differ from c_0. By repeating this process, we eventually conclude that $(c_0, \ldots, c_0)$ is an optimal solution of (4.5.15). ■

To the best of our knowledge, this is the first tightness result for a performance envelope (the reward-in-hindsight) different than the Bayes envelope, for repeated games. On the other hand, we note that our proof relies crucially on the assumption of a single constraint ($d = 1$), which allows us to order the f_i^+.

4.6 Attaining the Convex Hull Using Calibrated Forecasts

In this section, we consider a specific strategy that attains the convex hull, thus strengthening Theorem 4.3. The strategy is based on forecasting P2's action, and playing a best response (in the sense of (4.2.6)) against the forecast. The quality of the resulting strategy depends, of course, on the quality of the forecast; it is well known that using *calibrated* forecasts leads to no-regret strategies in standard repeated matrix games. See [51, 34] for a discussion of calibration and its implications in learning in games. In this

section we consider the consequences of calibrated play for repeated games with constraints.

We start with a formal definition of calibrated forecasts and calibrated play, and then show that calibrated play attains r^c in the sense of Definition 4.1.

A forecasting scheme specifies at each stage k a probabilistic forecast $q_k \in \Delta(B)$ of P2's action b_k. More precisely a (randomized) forecasting scheme is a sequence of maps that associate with each possible history h_{k-1} during the first $k-1$ stages a probability measure μ_k over $\Delta(B)$. The forecast $q_k \in \Delta(B)$ is then selected at random according to the distribution μ_k. Let us clarify that for the purposes of this section, the history is defined to include the realized past forecasts.

We shall use the following definition of calibrated forecasts.

Definition 4.2 (Calibrated forecasts)**.** A forecasting scheme is *calibrated* if for every (Borel measurable) set $Q \subset \Delta(B)$ and every strategy of P1 and P2

$$\lim_{t\to\infty} \frac{1}{t} \sum_{j=1}^{t} 1_{\{q_j \in Q\}} (e(b_j) - q_j) = 0, \quad \text{a.s.}, \tag{4.6.16}$$

where $e(b)$ is a vector of zeroes, except for a 1 in its bth component.

Calibrated forecasts, as defined above, have been introduced into game theory in [51], and several algorithms have been devised to achieve them (see [34] and references therein). These algorithms typically start with predictions that are restricted to a finite grid, and gradually increase the number of grid points.

The proposed strategy is to let P1 play a best response against P2's forecast play while still satisfying the constraints (in expectation, for the single

stage game). Formally, we let:

$$p^*(q) = \arg\max_{p\in\Delta(A)} \quad \sum_{a,b} p(a)q(b)\mathbf{R}(a,b) \qquad (4.6.17)$$
$$\text{s.t.} \quad \sum_{a,b} p(a)q(b)\mathbf{C}(a,b) \in \mathcal{T},$$

where in the case of a non-unique maximum we assume that $p^*(q)$ is uniquely determined by some tie-breaking rule; this is easily done, while keeping $p^*(\cdot)$ a measurable function. The strategy is to play $p_t = p^*(q_t)$, where q_t is a calibrated forecast of P2's actions[2] . We call such a strategy a *calibrated strategy.*

The following theorem states that a calibrated strategy attains the convex hull.

Theorem 4.5. *Let Assumption 4.1 hold, and suppose that P1 uses a calibrated strategy. Then, r^c is attainable with respect to $\mathcal{T}$.*

Proof. Fix $\epsilon > 0$. We need to show that by playing the calibrated strategy, P1 obtains $\liminf_{t\to\infty}(\widehat{r}_t - r^c(\widehat{q}_t)) \geq 0$ and $\limsup_{t\to\infty} \mathrm{dist}(\widehat{c}_t, \mathcal{T}) \leq 0$, almost surely.

Fix some $\epsilon > 0$. Consider a partition of the simplex $\Delta(B)$ to finitely many measurable sets $Q_1, Q_2, \ldots, Q_\ell$ such that $q, q' \in Q_i$ implies that $\|q - q'\| \leq \epsilon$ and $\|p^*(q) - p^*(q')\| \leq \epsilon$. (Such a partition exists by the compactness of $\Delta(B)$ and $\Delta(A)$. The measurability of the sets Q_i can be guaranteed because the mapping $p^*(\cdot)$ is measurable.) For each i, let us fix a representative element $q^i \in Q_i$, and let $p^i = p^*(q^i)$.

Since we have a calibrated forecast, (4.6.16) holds for every Q_i, $1 \leq i \leq \ell$. Define $\Gamma_t(i) = \sum_{j=1}^{t} 1_{\{q_j \in Q_i\}}$ and assume without loss of generality that

[2] When the forecast μ_t is mixed, q_t is the realization of the mixed rule.

$\Gamma_t(i) > 0$ for large t (otherwise, eliminate those i for which $\Gamma_t(i) = 0$ for all t, and renumber the Q_i). To simplify the presentation, we assume that for every i, and for large enough t, we have $\Gamma_t(i) \geq \epsilon t$. (If for some i, and t this condition is violated, the contribution of such an i in the expressions that follow will be $O(\epsilon)$.)

By the strong law of large numbers, we have

$$\lim_{t\to\infty}\left(\widehat{c}_t - \frac{1}{t}\sum_{j=1}^{t}\mathbf{C}(a_j,b_j)\right) = 0, \qquad \text{a.s.} \tag{4.6.18}$$

By definition, we have

$$\frac{1}{t}\sum_{j=1}^{t}\mathbf{C}(a_j,b_j) = \sum_i \frac{\Gamma_t(i)}{t}\sum_{a,b}\mathbf{C}(a,b)\frac{1}{\Gamma_t(i)}\sum_{j=1}^{t}1_{\{q_j\in Q_i\}}1_{\{a_j=a\}}1_{\{b_j=b\}}.$$

Observe that whenever $q_t \in Q_i$, we have $\|p_j - p^i\| \leq \epsilon$, where $p_j = p^*(q_j)$ and $p^i = p^*(q^i)$ because of the way the sets Q_i were constructed. By the strong law of large numbers, the frequency with which a will be selected whenever $q_j \in Q_i$ and $b_j = b$, will be approximately $p^i(a)$. Hence, for all b,

$$\limsup_{t\to\infty}\left|\frac{1}{\Gamma_t(i)}\sum_{j=1}^{t}1_{\{q_j\in Q_i\}}1_{\{a_j=a\}}1_{\{b_j=b\}} - p^i(a)\frac{1}{\Gamma_t(i)}\sum_{j=1}^{t}1_{\{q_j\in Q_i\}}1_{\{b_j=b\}}\right| \leq \epsilon,$$

a.s. By the calibration property (4.6.16) for $Q = Q_i$, and the fact that whenever $q, q' \in Q_i$, we have $\|q - q'\| \leq \epsilon$, we obtain

$$\limsup_{t\to\infty}\left|\frac{1}{\Gamma_t(i)}\sum_{j=1}^{t}1_{\{q_j\in Q_i\}}1_{\{b_j=b\}} - q^i(b)\right| \leq \epsilon, \qquad \text{a.s.}$$

By combining the above results, we obtain

$$\lim_{t\to\infty}\left|\widehat{c}_t - \sum_i \frac{\Gamma_t(i)}{t}\sum_{a,b}\mathbf{C}(a,b)p^i(a)q^i(b)\right| \leq K\epsilon, \qquad \text{a.s.,} \tag{4.6.19}$$

for some absolute constant K.

Note that the sum over index i in (4.6.19) is a convex combination (because the coefficients $\Gamma_t(i)/t$ sum to 1) of elements of $\mathcal{T}$ (because of the definition of p^i), and is therefore an element of $\mathcal{T}$ (because $\mathcal{T}$ is convex). This establishes that the constraint is asymptotically satisfied within $O(\epsilon)$. Note that in this argument, whenever $\Gamma_t(i)/t < \epsilon$, the summand corresponding to i is indeed of order $O(\epsilon)$ and can be safely ignored, as stated earlier.

Regarding the average reward, an argument similar to the above yields

$$\liminf_{t\to\infty} \widehat{r}_t \geq \liminf_{t\to\infty} \sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} \mathbf{R}(a,b) p^i(a) q^i(b) - K'\epsilon, \qquad \text{a.s.},$$

for some absolute constant K'. Next, observe that

$$\sum_i \frac{\Gamma_t(i)}{t} \sum_{a,b} \mathbf{R}(a,b) p^i(a) q^i(b) = \sum_i \frac{\Gamma_t(i)}{t} r^*(q^i) \geq r^c\Big(\sum_i \frac{\Gamma_t(i)}{t} q^i\Big),$$

where the equality is a consequence of the definition of p^i, and the inequality follows by the definition of r^c as the closed convex hull of r^*. Observe also that the calibration property (4.6.16), with $Q = \Delta(B)$, implies that

$$\lim_{t\to\infty} \left\| \widehat{q}_t - \frac{1}{t} \sum_{j=1}^{t} q_j \right\| = 0, \qquad \text{a.s.}$$

In turn, since $\|q_j - q^i\| \leq \epsilon$ for a fraction $\Gamma_t(i)/t$ of the time,

$$\limsup_{t\to\infty} \left\| \widehat{q}_t - \sum_i \frac{\Gamma_t(i)}{t} q^i \right\| = \limsup_{t\to\infty} \left\| \frac{1}{t} \sum_{j=1}^{t} q_j - \sum_i \frac{\Gamma_t(i)}{t} q^i \right\| \leq \epsilon, \qquad \text{a.s.}$$

Recall that the function r^c is continuous, hence uniformly continuous. Thus, there exists some function g, with $\lim_{\epsilon\downarrow 0} g(\epsilon) = 0$, such that when the argument of r^c changes by at most ϵ, the value of r^c changes by at most $g(\epsilon)$. By combining the preceding results, we obtain

$$\liminf_{t\to\infty} \widehat{r}_t \geq r^c(\widehat{q}_t) - K'\epsilon - g(\epsilon), \qquad \text{a.s.}$$

The above argument involves a fixed ϵ, and a fixed number ℓ of sets Q_i, and lets t increase to infinity. As such, it establishes that for any $\epsilon > 0$ the function $r^c - K'\epsilon - g(\epsilon)$ is attainable with respect to the set $\mathcal{T}^\epsilon$ defined by $\mathcal{T}^\epsilon = \{x \mid \text{dist}(x, \mathcal{T}) \leq K\epsilon\}$. Since this is true for every $\epsilon > 0$, we conclude that the calibrated strategy attains r^c as claimed. ■

4.7 Algorithms

The results in the previous section motivate us to develop algorithms for online learning with constraints, perhaps based on calibrated forecasts. For practical reasons, we are interested in computationally efficient methods, but there are no known computationally efficient calibrated forecasting algorithms. For this reason, we will consider related heuristics that are similar in spirit, even if they do not have all the desired guarantees.

We first consider a method based on the weighted average predictor [34]. The algorithm in Algorithm 4–1 keeps track of the performance of the different actions in the set A, updating a corresponding set of weights accordingly at each step. The main idea is to quantify "performance" by a linear combination of the total reward and the magnitude of the constraint violation. The parameter $\lambda > 0$ of the algorithm, which acts similar to a Lagrange multiplier, determines the trade-off between these two objectives. When the average penalty is higher than c_0 (*i.e.*, there is a violation), the weight of the cost term increases. When the average penalty is lower than c_0, the weight of the cost term decreases. The parameters $\overline{M}$ and $\underline{M}$ are used to bound the magnitude of the weight of the cost term; in the experiments reported in Section 4.8, they were set to 1000 and 0.001, respectively.

The second algorithm uses the tracking forecaster [90] as the forecasting method. This forecaster predicts that the distribution of the next action as a weighted average of previous actions, weighing recent actions more than less

1. Set $\lambda > 0$, w_0, $\overline{M}$, and $\underline{M}$.
2. For $t = 1, 2, \ldots$, compute

$$w_t(a) = w_{t-1}(a)\ \exp\left(\eta\ \left(\mathbf{R}(a, b_t) - \lambda \mathbf{C}(a, b_t)\right)\right), \quad a \in A, \tag{4.7.20}$$

sample an independent random variable a_t distributed so that

$$a_t = a, \quad \text{with probability } \frac{w_t(a)}{\sum_{a \in A} w_t(a)} \text{ for } a \in A. \tag{4.7.21}$$

3. For $t = 1, 2, \ldots$, update λ as follows:

$$\lambda := \begin{cases} \min(2\lambda, \overline{M}), & \text{if } \widehat{c}_t > c_0, \\ \max(\lambda/2, \underline{M}), & \text{otherwise.} \end{cases}$$

Algorithm 4–1: Exponentially weighted average predictor.

recent ones others. For the special case of only two actions, it is calibrated, but *not* calibrated in general. There are, however, some special cases where it is calibrated, in particular if the sequence it tries to calibrate comes from a source with some specific properties; see [90] for details. The algorithm is presented in Algorithm 4–2. If there is a current violation, it selects an action that minimizes the immediate forecast cost. If the current average penalty does not violate the constraint, it selects a best response to the forecast action of P2, while satisfying the constraints.

4.8 Experimental Setup

Our experimental testbed addresses the problem of minimizing power consumption minimization in a computer with a human user. The agent is a low-level software controller that decides when to put the central processor (CPU) into a low-power state, thereby reducing power expenditures during periods when the user is idle. The system is driven by a human user, as well as different hardware processes, and can be realistically assumed to be non-stationary. The actions of the system correspond to hardware interrupts (most interrupts are generated by hardware controllers on the motherboard such as

1. Set $\gamma \in (0,1)$, c_0, and $f_0 = (1/|B|)\vec{1}$.
2. For $t = 1, 2, \ldots$:
   - If $\widehat{c}_t > c_0$, choose an action that minimizes the worst-case cost:

$$a_t \in \arg\min_{a \in A} \left(\mathbf{C}(a,b) f_t(b) \right),$$

   - Otherwise (if $\widehat{c}_t \leq c_0$), solve

$$\begin{aligned} \max_{p \in \Delta(A)} \quad & \sum_{a,b} p(a) \mathbf{R}(a,b) f_t(b), \\ \text{subject to} \quad & \sum_{a,b} p(a) \mathbf{C}(a,b) f_t(b) \leq c_0. \end{aligned}$$

     and choose a random action distributed according to the solution to the above linear program.
3. For $t = 1, 2, \ldots$, update the forecast f_t on the probability distribution of the next opponent action b_{t+1} by letting

$$f_{t+1} = f_t + \left(\frac{1}{t+1} \right)^{\gamma} (e_{b_t} - f_t),$$

   where e_b is a unit vector in $\mathbb{R}^{|B|}$ with the element 1 in the component corresponding to $b \in B$.

Algorithm 4–2: Tracking forecaster.

direct memory access, hard disk interrupts and networking interrupts) and the ongoing running processes. In the particular application at hand, there is a software interrupt (generated by the Windows operating system) every 16 milliseconds. The times of these interrupts are the decision epochs, at which the software controller can decide if and when to put the CPU to sleep before the next scheduled periodic interrupt.

However, saving energy by putting the processor in the low-power state comes at a cost. In the low-power state, a delay is incurred each time that the processor moves back into the high-power state in response to user-generated interrupts. We wish to limit the delay perceived by the human user. For this purpose, we assign a cost to the event that an interrupt arrives while

the processor is in the low-power state, and impose a constraint on the time average of these costs. A similar model was used in [81], and we refer the reader to that work for further details.

We formulate the problem as follows. We divide a typical 16 millisecond interval into ten intervals. We let P1's action set be $A = \{0, 0.1, 0.2, \ldots, 1\}$, where action a corresponds to turning off the CPU after $16a$ milliseconds (the action $a = 1$ means the CPU is not turned off during the interval while the action $a = 0$ means it is turned off for the whole interval). Similarly, the action set of P2 is $B = \{0, 0.1, 0.2, \ldots, 0.9\}$, where action b corresponds to an interrupt after $16 \times b$ milliseconds. (Note that the action $b = 0$ means there is no interrupt and that there is no point in including an action $b = 1$ in B since it would coincide with the known periodic interrupt.) The assumption is that an interrupt is handled instantaneously so if the CPU chooses a slightly larger than b it maximizes the power savings while incurring no penalty for observed delay (it is assumed for the sake of discussion that only a single interrupt is possible in each 16 millisecond interval). We define the reward at each stage as follows:

$$\mathbf{R}(a,b) = \begin{cases} 1-a, & \text{if } a > b, \\ 1, & \text{if } a = b = 0, \\ b-a, & \text{if } b \geq a > 0. \end{cases}$$

The cost is:

$$\mathbf{C}(a,b) = \begin{cases} 1, & \text{if } a \leq b \text{ and } b > 0, \\ 0, & \text{otherwise.} \end{cases}$$

In "normal" operation where the CPU is powered throughout, the action is $a = 1$ and in that case there is no reward (no power saving) and no cost (no perceived delay). When $a = 0$ the CPU is turned off immediately and in this case the reward will be proportional to the amount of time until an interrupt

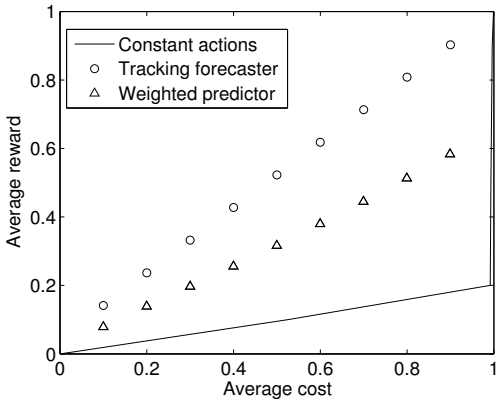
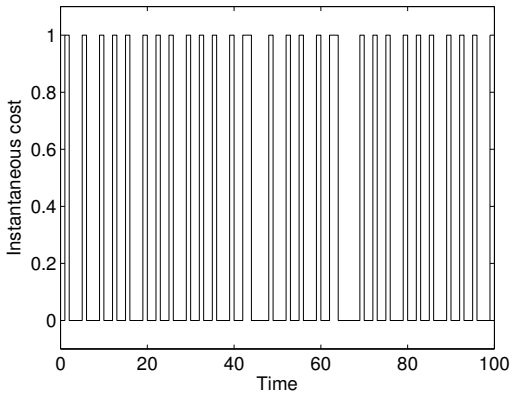


Figure 4–4: Plot of average reward against constraint violation frequency from experiments in power management for the MM05 data.

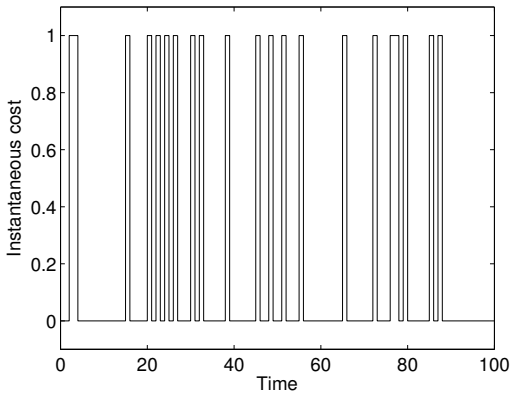
(or until the next decision). The cost in the case $a = 0$ is zero only is there is no interrupt ($b = 0$).

We used the real data trace obtained from what is known as MobileMark 2005 (MM05), a performance benchmark that simulates the activity of an average Microsoft Windows user. This CPU activity trace is 90 minutes long and contains more than 500000 interrupts, including the periodic scheduled interrupts mentioned earlier. The exponentially weighted algorithm (Algorithm 4–1) and the tracking forecaster (Algorithm 4–2) were run on this data set. Figure 4–4 shows the performance of the two algorithms. The straight line shows the trade-off between constraint violation and average reward by picking a fixed action over the entire time horizon. The different points for the exponential weighted predictor (Algorithm 4–1) or the tracking forecaster (Algorithm 4–2) correspond to different values of c_0. We observe that for the same average cost, the tracking forecast performs better (*i.e.*, gets higher reward).

We selected $c_0 = 0.3$ and used both algorithms for the MM05 trace. Figures 4–5(a) and 4–5(b) show the instantaneous cost incurred by the tracking forecaster and the weighted average forecaster over the same short period. It



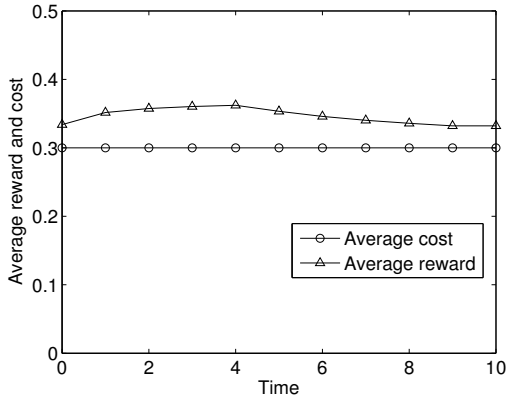
(a) Tracking forecaster



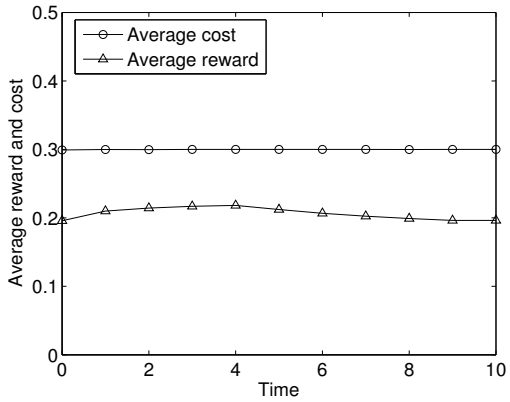
(b) Weighted average predictor

Figure 4–5: Instantaneous cost incurred by the tracking forecaster and weighted average predictor with target constraint $c_0 = 0.3$ for the MM05 data.

should be observed that the cost of the algorithms is different, reflecting the fact that different policies are employed. Figures 4–6(a) and 4–6(b) show the time evolution of the average reward and average cost for the same experiment. In spite of not being calibrated, the tracking forecast based algorithm outperforms the exponentially weighted based algorithm.



(a) Tracking forecaster



(b) Weighted average predictor

Figure 4–6: Time evolution of average reward and average cost for the tracking forecaster and weighted average forecaster with $c_0 = 0.3$ for the MM05 data.

CHAPTER 5
Constrained Online Learning for Power Management

5.1 Introduction

In this chapter, we demonstrate an application of a model of online optimization with side constraints similar to that of Chapter 4. The objective is to maximize the average reward accumulated over time subject to side constraints. Side constraints are common in real-world problems. For instance, power management problems are often formulated as maximizing power savings subject to an average performance criterion. This criterion restrict the rate of actions that impede the intended purpose of the system. In a computer system for example, power savings can be obtained by turning down the processor at appropriate moments. This also has the negative side effect of introducing delays in the performance of useful computations. A performance criterion in this example can be represented by a constraint on the average delay.

Online learning methods [34] have been studied extensively and have also been successfully used to solve many real-world problems, such as adaptive caching [62] and power management [69, 42, 82]. The novelty of our model is that in addition to nonstationary reward functions, the constraints are also nonstationary, *e.g.*, controlled by an opponent.

Our goal is to solve our problem in an efficient manner. To achieve this, we consider a solution concept different from the notion of attainability of Chapter 4. We present an efficient expert-algorithm with a guarantee that the average regret vanishes over time while the frequency of constraint violations remains bounded. The performance of this algorithm is further demonstrated

on a real-world power management problem. Our results suggest that non-stationary online optimization with constraints can be done successfully in practice.

This work makes two contributions. First, we apply prediction with expert advice to solve online optimization problems with nonstationary rewards and constraints efficiently. Our solution is based on mixing policies with a bounded number of constraint violations. Based on our knowledge, this is the first online solution to our problem that is efficient and has performance guarantees. This solution is suitable for real-world optimization problems, where we need to adapt to the environment over time without making any statistical assumptions on the environment. To support this claim, we demonstrate the performance of our solution in a real-world power management domain. This is our second contribution.

This chapter is structured as follows. First, we formulate our optimization problem and relate it to existing works. Second, we propose and analyze a practical solution to the problem based on prediction with expert advice. Third, we evaluate our solution on a real-world power management problem.

5.2 Online Constrained Optimization

In this chapter, we study an online learning problem, where an agent wants to maximize its total reward subject to temporal constraints. At every time instant t, the agent takes some action θ_t from the action set $\mathcal{A}$, and then receives a reward $r_t(\theta_t) \in [0, 1]$ and a cost $c_t(\theta_t) \in [0, 1]$. We assume that our agent has no prior knowledge on the sequence of reward and cost functions $r_1, r_2, \ldots$ and $c_1, c_2, \ldots$ except that they are bounded. Therefore, they can be generated in a non-stationary or even adversarial way. The agent may consider only the past reward functions $r_1, \ldots, r_{t-1}$ and the past cost functions $c_1, \ldots, c_{t-1}$ when deciding what action θ_t to take.

To situate our online learning problem and its challenges, we first define an *offline* version of the problem. This offline version simply assumes that our agents knows all reward and cost terms in advance. In such a setting, the optimal strategy of the agent is a solution to the optimization problem:

$$\max_{\boldsymbol{\theta}} \quad \frac{1}{T}\sum_{t=1}^{T} r_t(\theta_t) \tag{5.2.1}$$

$$\text{subject to:} \quad g_t(\boldsymbol{\theta}) \triangleq \frac{1}{\tau}\sum_{\ell=t-\tau+1}^{t} c_\ell(\theta_\ell) \leq c_0 \quad \text{for } t \in \mathcal{G}, \tag{5.2.2}$$

where we seek a sequence of actions $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_T)$ that maximizes the average reward over T time steps subject to a set of constraints indexed by a subset $\mathcal{G}$ of the time horizon $1, \ldots, T$. Observe that each constraint function $g_t(\boldsymbol{\theta})$ is an average of the instantaneous cost functions over a window of τ time instants ending at time t. Note that τ, c_0 and $\mathcal{G}$ are given parameters of this problem.

Instances of the above problem are common in the field of engineering. For example, most power management problems can be formulated as maximizing power saving subject to some performance criteria. These criteria are represented by the sequence of constraints

$$g_t(\boldsymbol{\theta}) \leq c_0, \quad \text{for } t \in \mathcal{G}.$$

The scalar c_0 is a strict bound on the tolerable range of average cost.

The offline version of our online optimization problem (5.2.1) can be solved by standard optimization techniques [21]. In this chapter, however, we solve this problem in an online fashion. Similarly to the settings of the previous chapters, we assume that the time horizon T is unknown and we make no statistical assumption about the reward functions $r_1, r_2, \ldots$ and cost functions $c_1, c_2, \ldots$. As a result, we can not expect to learn a policy that achieves

the optimal solution of the offline setting. Our objective is more modest: We want to learn a policy that performs over the long-term as well as the best policy from a limited set of alternative policies. Moreover, at the same time, our policy must also satisfy a given fraction of the constraints.

5.2.1 Objective

Suppose that we have access to set of experts $\xi_1, \ldots, \xi_N$. Each expert ξ_n is a policy that outputs an action $\xi_n(t)$ at every time instant t (based on the same observations as the agent). We seek an online algorithm that generates a sequence of actions $\theta_1, \theta_2, \ldots$ such that the *regret* is sub-linear, *i.e.*,

$$\limsup_{T\to\infty} \left\{ \max_{n=1,\ldots,N} \frac{1}{T} \sum_{t=1}^{T} r_t(\xi_n(t)) - \frac{1}{T} \sum_{t=1}^{T} \mathbb{E} r_t(\theta_t) \right\} \leq 0; \tag{5.2.3}$$

moreover, the frequency of *constraint violations*[1] is bounded:

$$\limsup_{T\to\infty} \frac{1}{|\mathcal{G}|} \sum_{t\in\mathcal{G}} \mathbf{1}_{[g_t(\boldsymbol{\theta})>c_0]} \leq \epsilon, \quad \text{a.s.} \tag{5.2.4}$$

In other words, we want to achieve close-to-optimal rewards as $T \to \infty$ while violating a vanishing number of constraints. These two objectives are optimized independently instead of being combined. This allows us to provide separate guarantees on the regret and constraint violation of learned policies. Therefore, our approach is suitable for constrained optimization problems, where the trade-off between the two objectives is hard to quantify, or it cannot be established at all.

5.2.2 Related Works

The model of this chapter is similar to that of Chapter 4, and hence share the same literature of related works. The difference of this chapter

[1] This work can be extended to other constraint violation metrics, such as the magnitude of violated constraints $\sum_{t\in\mathcal{G}} [g_t(\boldsymbol{\theta}) - c_0]^+$.

are the solution concept and the solution approach. Instead of the notion of attainability, we consider the solution concept of Section 5.2.1. Another difference is in terms of the constraints: our goal is to satisfy a sequence of finite-horizon constraints

$$\frac{1}{\tau}\sum_{\ell=t-\tau+1}^{t} c_\ell(\theta_\ell) \leq c_0 \quad \text{for } t \in \mathcal{G},$$

rather than a sequence of terminal constraints (cf. Section 4.2)

$$\frac{1}{t}\sum_{\ell=1}^{t} c_\ell(\theta_\ell) \leq c_0, \quad t = 1, 2, \ldots.$$

5.3 An Online Learning Solution

We propose a solution by modifying the exponentially weighted forecaster [34]. This solution is based on a set of experts $\xi_1, \ldots, \xi_N$, each of which is a policy that produces a solution to the constrained optimization problem (5.2.1) that satisfies a specified fraction of constraints. These experts are often available in practice. For instance, in the power management domain, these experts may represent heuristic policies that are conservative enough to satisfy the specified fraction of constraints.

5.3.1 Non-overlapping Constraints and Ideal Experts

First, let us assume that we have access to a pool of experts $\xi_1, \ldots, \xi_N$ that never violate the constraints (5.2.2). Moreover, we assume that the constraints of (5.2.2) are *non-overlapping*, *i.e.*, if $g_t(\boldsymbol{\theta}) \leq c_0$ and $g_{t'}(\boldsymbol{\theta}) \leq c_0$ are two constraints with t and t' belonging to $\mathcal{G}$, then $|t - t'| \geq \tau$. An example of such a sequence of constraints is shown in Figure 5–1a. Observe that each constraint span τ time steps and consecutive constraints occur at intervals of τ time steps (*i.e.*, at time instants $t = \tau, 2\tau, 3\tau, \ldots$. We shall forgo these two assumptions in later sections.

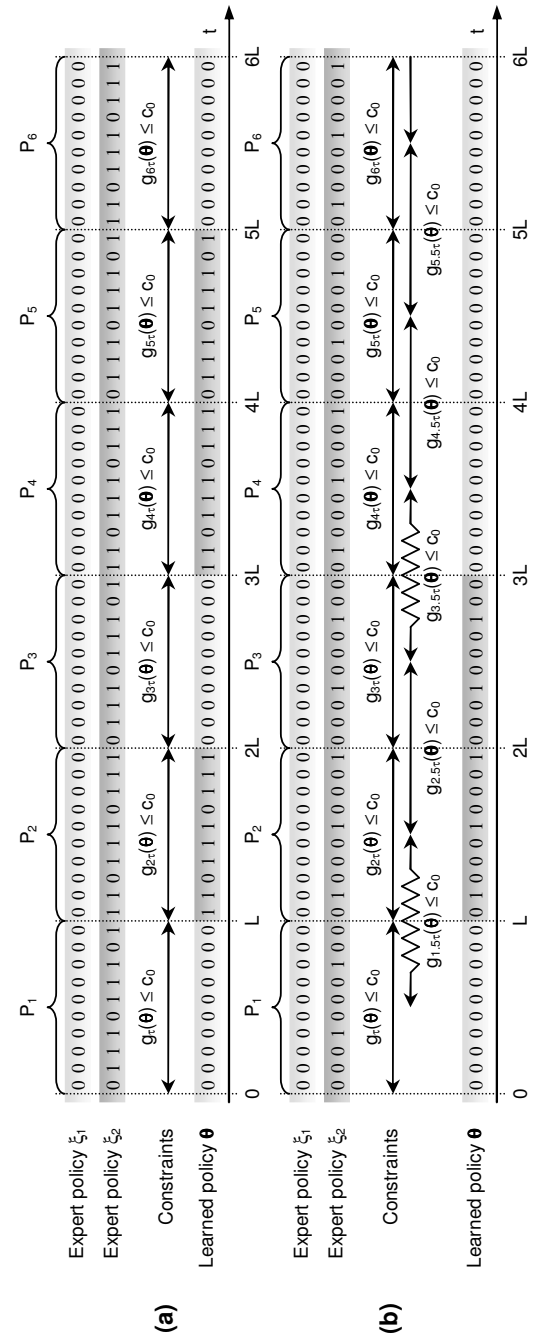


Figure 5–1: **a.** An online optimization problem with non-overlapping constraints. The example involves expert policies ξ_1 and ξ_2, side constraints, and the sequence of actions $\boldsymbol{\theta}$ generated by the lazy learner algorithm. The actions are listed chronologically. The temporal span of each constraint is illustrated by arrows. Time steps at which the lazy learner may switch between experts are denoted by dotted lines. **b.** An online optimization problem with overlapping constraints. Constraints that may be violated when switching between two experts at the instants depicted by jagged arrows.

(Initialize.) Fix the learning window L, learning rate η, expert policies $\xi_1, \ldots, \xi_N$. Initialize the expert weights $w_{t-1}(1), \ldots, w_{t-1}(N)$ to $1/N$. For $t = 1, 2, \ldots$:

1. If ($t \pmod L \equiv 1$), randomly choose an expert e_t according to the distribution

$$P(e_t = j) = \frac{w_{t-1}(j)}{\sum_{n=1}^{N} w_{t-1}(n)}, \quad j = 1, \ldots, N;$$

2. Otherwise, set $e_t = e_{t-1}$.
3. Play an action $\theta_t = \xi_{e_t}(t)$.
4. For every $n = 1, \ldots, N$, update:

$$w_t(n) = w_{t-1}(n) \exp[\eta r_t(\xi_n(t))].$$

Algorithm 5–1: The lazy learner algorithm.

Consider the lazy learner algorithm (Algorithm 5–1). We choose an integer L that is a multiple of the constraint span τ, then we partition the time steps $1, \ldots, T$ into *intervals* $\mathcal{P}_1, \ldots, \mathcal{P}_{T/L}$ of the length L each. Observe that, as a consequence, every constraint belongs to a single interval (cf. Figure 5–1a). The lazy learner algorithm is a modification of the exponentially weighted forecaster [34] where the algorithm may switch between experts at the beginning of each interval, *i.e.*, once every L time steps. The *learning window* L is the main parameter of the algorithm.

The following proposition bounds the regret and frequency of constraint violations for the lazy learner algorithm.

Proposition 5.1. *Suppose that the constraints do not overlap. Let $\xi_1, \ldots, \xi_N$ be expert policies that satisfy all constraints. Then the regret of the lazy learner (Algorithm 5–1) is bounded as follows:*

$$\max_{n=1,\ldots,N} \sum_{t=1}^{T} r_t(\xi_n(t)) - \sum_{t=1}^{T} \mathbb{E} r_t(\theta_t) \leq \frac{\log(N)}{\eta} + \frac{\eta T L}{2}. \tag{5.3.5}$$

Moreover, the lazy learner does not violate any constraint.

Proof. Our first claim is proved by an analysis similar to the exponentially weighted forecaster with a convex reward function $\mathbb{E}r_t(\theta_t) = \sum_{j=1,\dots,N} P(j) r_t(\xi_j(t))$:

$$\begin{aligned}
&\max_{n=1,\dots,N} \sum_{t=1}^{T} r_t(\xi_n(t)) - \sum_{t=1}^{T} \mathbb{E}r_t(\theta_t) \\
&= \max_{n=1,\dots,N} \sum_{m=0}^{T/L-1} \sum_{t=mL+1}^{(m+1)L} (r_t(\xi_n(t)) - \mathbb{E}r_t(\theta_t)) \\
&\leq \frac{\log(N)}{\eta} + \frac{\eta}{2} \sum_{m=0}^{T/L-1} \\
&\quad \max_{n=1,\dots,N} \left(\sum_{t=mL+1}^{(m+1)L} (r_t(\xi_n(t)) - \mathbb{E}r_t(\theta_t)) \right)^2 \\
&\leq \frac{\log(N)}{\eta} + \frac{\eta}{2} \frac{T}{L} L^2 \\
&= \frac{\log(N)}{\eta} + \frac{\eta T L}{2}.
\end{aligned}$$

The first step of the proof follows by algebra, the second step is based on Theorem 2.1 [34], and the third step results from the reward terms r_t being bounded on the interval $[0, 1]$.

Our second claim follows from two assumptions. First, no constraint is violated by the experts. Second, no constraint is violated due to switching between the experts. Since our constraints do not overlap, and every expert satisfies all the constraints, the lazy learner also satisfies all the constraints. ■

The parameters η and L can be set such that the regret bound in Proposition 5.1 is sub-linear in T. From now on, we set the learning rate to $\eta = \sqrt{2\log(N)/(TL)}$. For instance, if we set $L = T^{1/2}$ and $\eta = \sqrt{2\log(N)} T^{-3/4}$, the bound is of the order of $O(T^{3/4})$. Therefore, the average regret of the lazy learner vanishes as T increases.

In the rest of the chapter, we study the lazy learner in a more general context. Specifically, we relax the assumptions that constraints do not overlap and that the experts do not violate any constraint. It turns out that the parameter L has an important role in this new setting. It allows a trade-off between the regret and the frequency of constraint violation.

5.3.2 Experts with Bounded Constraint Violation Frequency

Building of expert policies that satisfy all constraints may be too difficult in practice. In this section, we relax this assumption.

We construct experts that violate a bounded fraction ϵ of constraints as follows. First, we assume that there exists an action θ^0 that guarantees $c_t(\theta^0) = 0$ for every time step t. This action is called the arbitration action. For a fixed $\epsilon > 0$, we can modify every expert so that it does not violate more than a fraction ϵ of the constraints by taking the action θ^0 when the empirical frequency of constraint violations is about to exceed ϵ. We refer to this process as ϵ-*arbitration*.

The existence of an arbitration action is standard in most real-world domains. In the power management domain, for instance, it corresponds to taking no power management action. It is conceptually equivalent to requiring that the feasible set be non-empty for an optimization problem. The drawback of the arbitration action θ^0 is that it yields low reward.

From the preceding discussion and Proposition 5.1, we obtain the following guarantees on the regret and the frequency of constraint violations for the lazy learner.

Corollary 5.2. *Suppose that the constraints do not overlap. Suppose that the experts $\xi_1, \ldots, \xi_N$ are ϵ-arbitrated. Then the regret of the lazy learner is*

bounded as in (5.3.5) *while its constraint violation frequency is bounded as*

$$\frac{1}{|\mathcal{G}|}\sum_{t\in\mathcal{G}}\mathbf{1}_{[g_t(\boldsymbol{\theta})>c_0]}\leq\epsilon,\quad a.s.$$

5.3.3 Overlapping Constraints

Constraints have been assumed to be non-overlapping up to this point. In this section, we remove this assumption. A simple example of constraints that overlap is depicted in Figure 5–1b. Observe that the constraints span τ time steps. When the lazy learner switches between two experts, it may violate a constraint because the corresponding costs depend on the policies of two different experts.

Fortunately, if the number of overlapping constraints at the boundary of two consecutive intervals $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell+1}$ is bounded, the following proposition holds. This is the case in application domains where the constraints occur periodically

Proposition 5.3. *Suppose that the experts $\xi_1,\ldots,\xi_N$ are ϵ-arbitrated. Then the lazy learner has regret bounded as*

$$\max_{n=1,\ldots,N}\sum_{t=1}^{T}r_t(\xi_n(t))-\sum_{t=1}^{T}\mathbb{E}r_t(\theta_t)\leq\frac{\log(N)}{\eta}+\frac{\eta TL}{2},$$

and constraint violation frequency

$$\frac{1}{|\mathcal{G}|}\sum_{t\in\mathcal{G}}\mathbf{1}_{[g_t(\boldsymbol{\theta})>c_0]}\leq\epsilon+\frac{T\tau}{L\,|\mathcal{G}|},\quad a.s.$$

Proof. The regret bound is proved as in Proposition 5.1. The first term in the constraint violation bound comes from Corollary 5.2. The second term accounts for constraint violations due to switching between experts. Observe that at most τ constraints overlap between any two intervals $\mathcal{P}_\ell$ and $\mathcal{P}_{\ell+1}$. ■

Observe that if we set $L = T^{1/2}$ and $\eta = \sqrt{2\log(N)}T^{-3/4}$, the regret bound is of the order of $O(T^{3/4})$. Moreover, if the number of constraints is $|\mathcal{G}| = T/C$ for some constant C, then the constraint violation frequency is bounded by $\epsilon + C\tau T^{-1/2}$, which converges to ϵ asymptotically. In addition, observe that the value of the learning window L allows a trade-off between the regret and the constraint violation frequency.

In the rest of the chapter, we evaluate the performance of the lazy learner in a power management problem. The nature of the problem is not adversarial as typically assumed in other works in online learning. Therefore, the actual performance of the lazy learner is better than the bounds in our guarantees.

5.4 CPU Power Management

We evaluate the proposed solution on the challenging real-world problem of power management in a central processing unit (CPU) including multi-core processors, L1 and L2 caches, and associated circuitry. Solving this power management problem is important because the CPU may account for as much as 40 percent of the power consumed in a mobile computer.

The primary goal of CPU power management is to minimize the power consumption without impacting the performance, *i.e.*, without increasing the perceived latency. This objective can be stated as maximizing the *power saving* (or *residency*) of the CPU in low power states subject to constraints on the *latency* in serving hardware interrupts. The latency is a delay incurred when waking up the CPU from low power states. Some interrupts are generated periodically by the OS. Hardware interrupts, *e.g.*, generated by key strokes or other hardware, are uncertain. If an unpredicted interrupt occurs when the CPU is in a low power state, the CPU must wake up in order to serve it; hence, incurring a latency cost. The power management predicts whether the next

interrupt will occur sufficiently far away to power down the CPU for an period without affecting the latency. This prediction is made at regular periods.

The existing solutions to CPU power management consist of static timeout policies. A *static timeout policy* [73] is a simple strategy parametrized by a timeout parameter: if the CPU remains idle for more than the timeout parameter, the policy puts the CPU into a low power state. In the following sections, we employ the lazy learner algorithm with a pool of experts composed of adaptive timeout policies.

5.5 Experiments

The main goal of this section is to demonstrate online learning with constraints in practice—specifically, in power management problem. First, we collect data, or *traces*, of CPU activity from two experiments. The power savings and the interrupts are recorded by external measurement devices connected to the CPU. Then, we evaluate our algorithm on these traces through simulation in MATLAB. It should be noted that the lazy learner algorithm keeps track of a single index for each expert. Hence, it can be efficiently implemented and run online without affecting the overall performance of the system.

5.5.1 Setting

The first trace is recorded by running MobileMark 2005 (MM05), which is a performance benchmark simulating the activity of a Microsoft Windows user. The corresponding CPU activity trace is 90 minutes long and contains more than 500000 operating system (OS) interrupts. The second trace is generated by running in real-time the following applications: Adobe Photoshop, Microsoft Windows Explorer, Microsoft WordPad, and Microsoft Media Player. We refer to this trace as the heavy workload trace; it reflects 30 minutes of human activity and contains over 200000 OS interrupts.

In the model of online constrained optimization, each action θ_t represents the time instant when the CPU is put in its low power state, the instantaneous reward $r_t(\theta_t)$ represents the power saving (or residency), whereas the cost $c_t(\theta_t)$ represents the latency. In order to guarantee an acceptable perceived latency, our latency constraints average the latency costs over periods of 10 seconds or $\tau = 640$ time steps.

We apply the lazy learner (Algorithm 5–1) to this online constrained optimization problem. For the component experts $\xi_1, \ldots, \xi_N$, we employ adaptive timeout policies [82]. Each such policy adapts its timeout parameter periodically (at every OS interrupt) based on the empirical frequency of interrupts and according to the fixed-share algorithm [71]. Moreover, these policies are arbitrated to satisfy the latency constraints.

Unless specified otherwise, our experiment settings are as follows. The side constraints are defined at intervals of $\tau = 640$ time instants, such that $\mathcal{G} = \{640, 1280, 1920, \ldots\}$. The latency threshold c_0 is set to 0.03 and the learning window L is set to the same value as τ. Since the lazy learner is a randomized algorithm, we report empirical results that correspond to averages over ten simulations.

5.5.2 Results

Figure 5–2 shows that the lazy learner performs almost as well as the best expert in hindsight. Here, we assume non-overlapping constraints (Figure 5–1a) that are satisfied by every expert. The expected regret of the lazy learner is less than 2 percent. The lazy learner clearly performs better static timeout policies, which are currently the only employed methods of CPU power management. On the heavy workload trace with $c_0 = 0.06$, for instance, none of the static timeout policies yield more than 5 percent power saving, whereas the lazy learner algorithm achieves over six times more power saving.

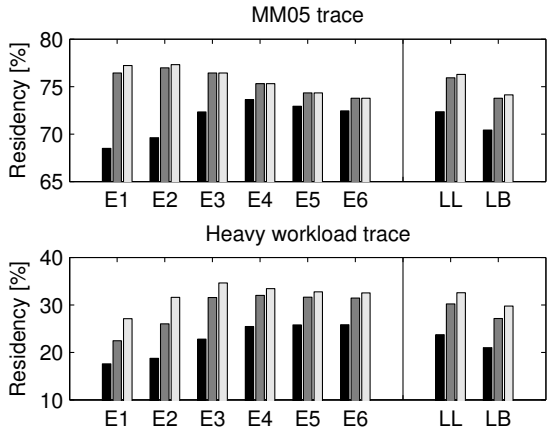


Figure 5–2: Comparison of the lazy learner and the pool of experts (E1, ..., E6). The power saving (residency) of the policies are plotted for different latency thresholds c_0: 0.02 (black bars), 0.04 (dark grey bars), and 0.06 (light grey bars). We also show the lower bounds (LB) on the power saving (residency) of the lazy learner according to Proposition 5.1.

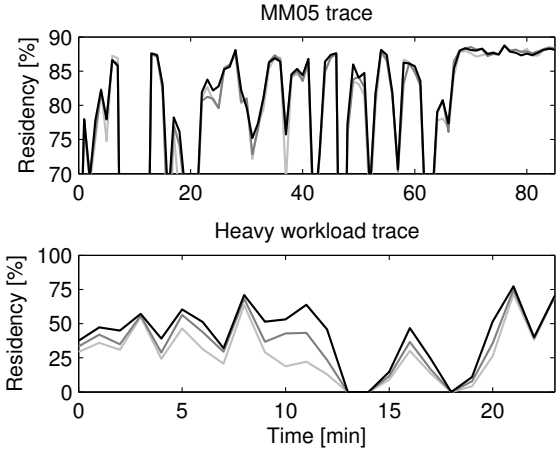


Figure 5–3: The instantaneous power saving (residency) of the lazy learner for three versions of ϵ-arbitration: $\epsilon = 0$ (light grey lines), $\epsilon = 0.5$ (dark grey lines), and $\epsilon = 0$ (black lines). The power saving is depicted as a function of the time in minutes.

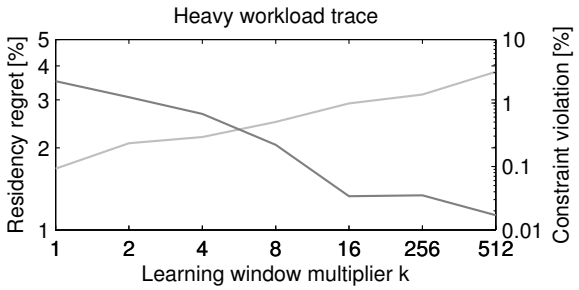


Figure 5–4: Constraint violation frequency (dark grey line) and power saving (residency) regret (light grey line) of the lazy learner as a function of the learning window $L = k\tau$ for different values of k.

Figure 5–3 shows the instantaneous power saving of the lazy learner with a pool of ϵ-arbitrated experts for different values of ϵ. We also assume here non-overlapping constraints (Figure 5–1a). We set the learning window to $L = 10\tau$. As expected, by relaxing the ϵ-arbitration process, the corresponding power saving increases.

Figure 5–4 shows the effect of the learning window L on the regret and constraint violation of the lazy learner algorithm. Here, we assume overlapping constraints (Figure 5–1b) defined at intervals of 1 second. As guaranteed by Proposition 5.3, the parameter L trades off regret and constraint violation. A higher value of L causes higher regret but lower constraint violation frequency.

CHAPTER 6
Piecewise-Stationary Bandit Problems with Side Observations

6.1 Introduction

In this chapter, we consider a sequential decision problem where the rewards are generated by a piecewise-stationary distribution. However, the different reward distributions are unknown and may change at unknown instants. Our approach uses a limited number of side observations on past rewards, but does not require prior knowledge of the frequency of changes. In spite of the adversarial nature of the reward process, we provide an algorithm whose regret, with respect to the baseline with perfect knowledge of the distributions and the changes, is $O(k \log(T))$, where k is the number of changes up to time T. The baseline of the notion of regret in this chapter is much more general: it including every sequence of policies.

In some learning scenarios, the agent is confronted with an adversarial opponent that can be very general and difficult to model, and is therefore modelled as an arbitrary non-stochastic process. In other scenarios, the opponent is stochastic, which may be characterized and adapted to. What about opponents that fall between these two extremes? An instance of the adversarial scenario is the expert problem [87], where the agent observes sequentially the performance of a number of experts, and (choosing one expert at each time step) tries to match the performance of the best expert in retrospect. An instance of the stochastic scenario is the multi-armed bandit problem [85], where each of n arms has a fixed reward distribution, and where the agent tries to obtain the performance of the best arm by picking and observing one arm each time step—without observing the reward of any other arm.

We consider a model that combines the bandit and expert models, and shall refer to the arms of the bandit and the experts interchangeably. The reward process of the arms is non-stationary on the whole, but stationary on intervals. This piecewise-stationary reward process is similar to that of the non-stationary bandit problem of [66, 58], or that of the multiple change-point detection problem of [4].

In our variant of the non-stationary bandit problem, the agent has the benefit of querying and observing some of the past outcomes of arms that have not been picked. This is the same benefit available to the agent in the expert problem (cf. [70]). The following examples motivate our model.

Example 6.1 (Investment options)**.** Consider the problem of choosing every day one of n investment options, say mutual funds. Our model assumes that the outcomes of these investments undergo changes reflecting changes in market conditions. Otherwise, the outcomes remains stationary over the periods between two changes, *e.g.*, they follow bearish or bullish trends. Suppose that the outcomes of the previous day's investment options are revealed today, *e.g.*, in the newspaper. Suppose that observing the outcome of each option requires a query (looking up a price history), which incur a querying cost. By limiting the number of queries allowed at each step, we can model the trade-off between the cost of observations and the regret due to insufficient observations.

Example 6.2 (Dynamic pricing with feedback)**.** As a second example, we consider a vendor whose task is to sell commodity X. Potential customers arrive sequentially, one after the other, and the demand for commodity X (for various prices) is modelled as a stationary process that may nonetheless change abruptly at unknown instants. To each customer, the vendor offers one of n possible prices. If the customer accepts, a corresponding profit is made. Bargaining is not an option, but after each transaction, the vendor has the

leisure to ask the customer if the outcome would have been different had a different price been offered (*e.g.*, through a short survey). A partial goal is to achieve as much profit as if the distribution of the demand were known at all times (even though unknown changes occur at unknown instants). A second goal is to minimize the cost associated with conducting surveys for feedback. A similar problem of dynamic pricing with partial-monitoring is also described in [34].

We present the setting in Section 6.2, followed by a survey of related works in Section 6.3. We present a solution and its guarantee in Section 6.4. In Section 6.6, we compare our solution with other solutions via simulation. In Section 6.7, we conclude with a discussion.

6.2 Setting

We consider the following sequential decision problem. Let $\{\alpha_1, \ldots, \alpha_n\}$ denote the n arms of a multi-armed bandit—or n experts of an online learning problem. Let $r_1, r_2, \ldots$ be a sequence of reward vectors in $\mathbb{R}^n$. The element $r_t(i)$ of r_t, for $i = 1, \ldots, n$ and $t = 1, 2, \ldots$, represents the reward associated with the i-th arm α_i at time t. For clarity of notation, we shall sometimes write $r_t(i)$ instead of $r_t(\alpha_i)$. We assume that the rewards take values in the unit interval $[0, 1]$, *i.e.*, $r_t(i) \in [0, 1]$ for all i and t.

6.2.1 Reward Process

In our model, the source of rewards is piecewise-stationary: *i.e.*, it changes its distribution arbitrarily and at arbitrary time instants, but otherwise remains stationary. The reward process $r_1, r_2, \ldots$ is an independent sequence of random variables that undergoes abrupt changes in distribution at unknown time instants $\nu_1, \nu_2, \ldots$, which are called *change-points*. By convention, we let $\nu_1 = 1$. Let f_t denote the distribution (probability density function) of r_t. Hence, $r_{\nu_1}, \ldots, r_{\nu_2 - 1}$ are i.i.d. with common distribution f_{ν_1}, as is the case

for stochastic learning problems (cf. [85]). Likewise, $r_{\nu_j}, r_{\nu_j+1}, \ldots, r_{\nu_{j+1}-1}$ are i.i.d. with distribution f_{ν_j}, for $j = 1, 2, \ldots$. The intervals are illustrated as follows:

$$\underbrace{r_1, r_2, \ldots, r_{\nu_2-1}}_{\text{distribution } f_{\nu_1}}, \underbrace{r_{\nu_2}, \ldots, r_{\nu_3-1}}_{\text{distribution } f_{\nu_2}}, \ldots, \underbrace{r_{\nu_j}, \ldots, r_{\nu_{j+1}-1}}_{\text{distribution } f_{\nu_j}} \cdots.$$

Similarly to adversarial learning problems (cf. [34]), both the change-points $\nu_1, \nu_2, \ldots$ and the distributions $f_{\nu_1}, f_{\nu_2}, \ldots$ are unknown. We can think of an opponent deciding the time instants (and frequency) of the changes, as well as the distribution after each change.

Remark 30. It is important that the changes occur at arbitrary instants. Otherwise, we only need to reset an algorithm for the multi-armed bandit problem at the appropriate instants.

The model of piecewise-stationary rewards combines two important models. If there are no changes, then we recover the stochastic source of the multi-armed bandit problem. If there is no constraint on the number of changes, we obtain the source of rewards adopted by the oblivious adversarial model of prediction with expert advice. We consider the interesting case where the frequency of changes is between these two extremes, *i.e.*, where the number of change-points

$$k \equiv k(T) \triangleq \sum_{t=1}^{T-1} \mathbf{1}_{[f_t \neq f_{t+1}]}$$

up to time T increases with T. To simplify notation, we shall simply write k in place of $k(T)$.

6.2.2 Decision-Maker

At each time step $t > 1$, the agent picks an arm $a_t \in \{\alpha_1, \ldots, \alpha_n\}$ and makes ℓ (where $1 \leq \ell \leq n$) observations on the individual arm-rewards

$r_{t-1}(1), \ldots, r_{t-1}(n)$ of the previous step. This is captured in the following assumption.

Assumption 6.1 (Partial observation)**.** At time 1, the agent chooses an action a_1 and an ℓ-subset $\mathcal{S}_1$ of the arms $\{\alpha_1, \ldots, \alpha_n\}$. At every time step $t > 1$, the agent chooses (deterministically) an ℓ-subset $\mathcal{S}_t$ and takes an action a_t that is a function of the reward observations

$$\{r_j(i) \mid j = 1, \ldots, t-1, \quad \alpha_i \in \mathcal{S}_j\}.$$

Partial observation allows us to capture querying costs associated with observations, and to quantify the total query budget.

6.2.3 Notion of Regret

At each time instant t, the agent chooses and activates an arm $a_t \in \{\alpha_1, \ldots, \alpha_n\}$ and receives the corresponding reward $r_t(a_t)$. Let $\rho_t = \mathbb{E}r_t$ denote the mean of the reward vector r_t. The agent's baseline—or objective—is the reward accumulated by picking at each instant t an arm with the maximal expected reward. Letting k denote the number of changes in reward distribution up to time T, the baseline is

$$\sum_{t=1}^{T} \max_{i=1,\ldots,n} \rho_t(i) = \max_{a'_1,\ldots,a'_T \,:\, k \text{ changes}} \sum_{t=1}^{T} \mathbb{E}[r_t(a'_t)],$$

where the maximum is taken over sequences of arms with only as many changes as change-points in the reward sequence $r_1, \ldots, r_T$, *i.e.*, over the set

$$\{(a'_1, \ldots, a'_T) \mid a'_{\nu_j} = \ldots = a'_{\nu_{j+1}-1} \quad \text{for } j = 1, \ldots, k\}.$$

Despite the appearance, this objective is reasonable when the number of changes k is small; it is also the same objective as in the classical stochastic multi-armed bandit problems. Hence, for a given reward process $r_1, r_2, \ldots$,

we define the expected regret of the agent at time T as

$$L_T \triangleq \sum_{t=1}^{T} \max_{i=1,\ldots,n} \rho_t(i) - \sum_{t=1}^{T} \mathbb{E}[r_t(a_t)], \tag{6.2.1}$$

where the expectation $\mathbb{E}$ is taken with respect to the sequence $r_1, r_2, \ldots$.

6.3 Related Works

In this section, we survey results concerning related models. The different models are distinguished by the source of the reward process, the observability of the rewards, and the baseline for the notion of regret.

6.3.1 Stochastic Multi-Armed Bandit

In stochastic multi-armed bandit problems [85, 9], the reward sequence $r_1, r_2, \ldots$ is a sequence of i.i.d. random vectors from a common unknown distribution $\rho_1 = \rho_2 = \ldots$. The reward observations are limited to rewards $r_1(a_1), r_2(a_2), \ldots$ corresponding to the arms chosen by the agent. This invites the agent to trade-off exploring the different arms to estimate their distributions and exploiting the arms with the highest empirical reward. The notion of regret is the same as ours (6.2.1). However, the optimal reward of the baseline can be obtained by a single fixed arm. In such problems, a number of algorithms, *e.g.*, [85, 9, 78], achieve the optimal expected regret of the order of $O(n \log(T)/\Delta)$, where Δ denotes the difference in mean between the best and second-best arms.

6.3.2 Adversarial Expert Problem

Many learning problems take the adversarial setting, *e.g.*, prediction with expert advice, etc.—see [34] for a comprehensive review. The sequence of rewards achieved by the experts is arbitrary; *i.e.*, no assumption is made regarding the joint distribution of $r_1, r_2, \ldots$. This approach essentially makes provisions for the worst-case sequence of reward. At time t, the past reward vectors $r_1, \ldots, r_{t-1}$ are observable by the agent. In this case, the notion of

adversarial regret is adopted, whose baseline is the reward accumulated by the best fixed expert, *i.e.*, $\max_{i=1,\ldots,n} \sum_{t=1}^{T} r_t(i)$. For every sequence $r_1, r_2, \ldots$, the (expected) adversarial regret

$$\max_{i=1,\ldots,n} \sum_{t=1}^{T} r_t(i) - \sum_{t=1}^{T} \mathbb{E}[r_t(a_t)]$$

is of the order of $O(\sqrt{T \log(n)})$—see [34] for a detailed account. A bound of $O(\sqrt{Tn \log(n)})$ holds when the observations are limited to the chosen arms: $r_1(a_1), r_2(a_2), \ldots$ [10].

The baseline in the adversarial case is limited to a single fixed expert, whereas our baseline in (6.2.1) is the optimal expected reward. Our baseline, which contains as many switches as changes in distribution, is similar to the baseline defined by appropriately chosen shifting policies in [70]. The fixed-share algorithm or one of its variants [70, 10] can be applied to our setting, if the number of changes k is given in advance, yielding a regret of $O(\sqrt{nkT \log(T)})$ [8]. We present an algorithm with a regret of $O(nk \log(T))$ without prior knowledge of k. It should be noted that when k is of the same order as T, it is hopeless to minimize the regret of (6.2.1): consider an adversary that picks the new distribution after each change-point.

6.3.3 Non-stationary Bandits

Our problem is reminiscent of the non-stationary bandit problem of [66, 58]. The reward process and the notion of regret are similarly defined, as in Section 6.2. However, in those works, observation of the past rewards is limited to the chosen arms; hence, at time t, the agent's choice a_t is a function of $r_1(a_1), r_2(a_2), \ldots$. Using a statistical change detection test, Hartland *et al.* present a partial solution for instances where the best arm is not superseded by another arm following a change. In the event that an oracle reveals a-priori

the number of changes k up to time T, Garivier and Moulines provide upper-confidence bound algorithms that achieve a regret of $O(n\sqrt{kT}\log(T)/\Delta)$, and show a lower-bound of $\Omega(\sqrt{T})$ for the regret.

With respect to the above non-stationary bandit model, the distinguishing feature of our model is that, in addition to activating an arm at each time instant, the agent may query the current reward of one or more arms. We show that with T queries in total, the regret is bounded by $O(nk\log(T)/\Delta)$. Hence, queries reduce significantly the regret with respect to the results of [58].

6.3.4 Change-Detection

A classical problem in statistics is that of detecting an abrupt change of distribution in an otherwise i.i.d. sequence of random variables (see [84] for a survey).

A standard assumption in change-detection problems is that the initial distribution is known, whereas the distribution after the change can be either known or not. Another common feature is the assumption that the distributions belong to one parametric family.

Methods based on CUSUM [103] and Shiryayev-Roberts (SR) rules solve the change-point detection problem optimally according to the expected run length criterion (in a minmax sense and in the asymptotic limit) [88]. The Shiryayev-Roberts method has the advantage that it can be used when the post-change distribution is not specified [107]. The unknown post-change distribution is usually specified up to an unknown parameter (parametric case). The method of generalized likelihood ratio with supremum over the unknown parameter θ, but no efficiency guarantee. If the unknown distribution after the change is from an exponential family with parameter θ, the method of assigning a distribution to θ [107] works. For nonparametric unknown post-change

distribution and the sequential nonparametric likelihood ratio approach, an additional assumption that the post-change distribution is stochastically larger than pre-change is required [60].

We consider a more general setting where in addition to the post-change distribution, the pre-change distribution (before every individual change) is also unknown. Moreover, both pre-change and post-change distributions are assumed to non-parametric. The case of pre-change and post-change distributions both unknown but parametric is considered in [95], however, with a special notion of optimality.

Another problem related to ours is that of fault detection-isolation [84]. In that problem, the goal is to detect the change, and classify the post-change distribution within a finite set of possibilities. In our problem, the distribution after the change can be arbitrarily different. Moreover, we do not classify; instead, we apply a minimum-regret learning algorithm. In contrast to the change-detection literature, we consider a more complex setting, where the sequence $r_1, r_2, \ldots$ may have multiple (arbitrarily many) changes. The problem of joint-detection of multiple change-points is addressed in [4] and references therein.

Since our notion of expected regret considers the expected rewards, our approach is to detect changes in the mean. Moreover, for some families of distributions (*e.g.*, Bernoulli, Geometric, etc.), a change in distribution results in a change in mean. In the special case of a sequence of normal random variables, changes in the mean can be detected using a moving-average method as in [83]. In nonparametric settings, limit theorems for detecting changes in the mean is studied in [40, Chapter 2].

6.4 Multi-armed Bandits with Queries

In this section, we present an algorithm for our setting and provide its performance guarantee. We begin with two assumptions. We shall use as a component of our solution a typical multi-armed bandit algorithm described in the first assumption. The second assumption describes a limitation of our algorithm.

Assumption 6.2 (MAB algorithm for $k = 1$)**.** Consider a multi-armed bandit where there are no distribution changes (except at time 1). Let the i.i.d. reward sequence $r_1, r_2, \ldots$ have mean $\bar{r}$. Let $\alpha_{i^{(1)}}$ and $\alpha_{i^{(2)}}$ denote, respectively, the arm with the highest and second-highest mean. Let Δ denote their mean difference:

$$\Delta = \bar{r}(i^{(1)}) - \bar{r}(i^{(2)}).$$

Let $\mathcal{A}$ be an algorithm that guarantees a regret of at most $Cn\log(T)/\Delta$, for some constant C. At each step $t > 1$, algorithm $\mathcal{A}$ receives as input the reward $r_{t-1}(a_{t-1})$ obtained in the previous step, and outputs a new arm choice a_t. Examples of candidate algorithms include those of [85, 9].

In this chapter, we are concerned with detecting abrupt changes bounded from below by some threshold; we exclude infinitesimal changes in the following assumption.

Assumption 6.3. Recall that $\rho_{\nu_j}(i)$ and $\rho_{\nu_{j+1}}(i)$ denote the pre-change and post-change means of the arm α_i at the change-point ν_{j+1}. There exists a known value $\epsilon > 0$ such that, for each $j = 1, 2, \ldots$, there exists an arm α_i such that

$$\left|\rho_{\nu_j}(i) - \rho_{\nu_{j+1}}(i)\right| > 2\epsilon.$$

6.4.1 The WMD Algorithm

Our algorithm (Algorithm 6–1) detects changes in the mean of a process, in the spirit of statistical methods for detecting an abrupt change of distribution in an otherwise i.i.d. sequence of random variables (see [84] for a survey). The algorithm partitions the time horizon into intervals of equal length τ. Hence, for $m = 1, 2, \ldots$, the m-th interval is comprised of the time instants $(m-1)\tau+1, (m-1)\tau+2, \ldots, m\tau$. The algorithm computes iteratively empirical mean vectors $\widehat{r}_1, \widehat{r}_2, \ldots$ over intervals (windows) of length τ, in the following fashion:

$$\underbrace{r_1, r_2, \ldots, r_\tau}_{\widehat{r}_1}, \underbrace{r_{\tau+1}, \ldots, r_{2\tau}}_{\widehat{r}_2}, \ldots, \underbrace{r_{(m-1)\tau+1}, \ldots, r_{m\tau}}_{\widehat{r}_m} \ldots.$$

The algorithm follows a multi-armed bandit algorithm $\mathcal{A}$ with a regret guarantee in the absence of changes (Assumption 6.2). When it detects a mean shift with respect to a threshold given by Assumption 6.3, it reset the sub-algorithm $\mathcal{A}$.

6.4.2 Regret Bound

The following theorem bounds the expected regret of the WMD algorithm.

Theorem 6.1 (WMD regret)**.** *Suppose that Assumption 6.1 holds. Suppose that the agent employs the WMD algorithm with a sub-algorithm satisfying Assumption 6.2, a threshold ϵ satisfying Assumption 6.3, and intervals of length $\tau = \lfloor \frac{n}{\ell} \rfloor \cdot \lfloor \frac{\log(T)}{2\epsilon^2} \rfloor$. Then, for every sequence of change-points $\nu_1, \nu_2, \ldots$ and every choice of post-change distributions $f_{\nu_1}, f_{\nu_2}, \ldots$, the expected regret is bounded as follows:*

$$L_T \leq \frac{7}{\epsilon^2} \frac{kn}{\ell} \log(T) + \frac{C}{\Delta} kn \log(T) + \frac{6C}{\Delta} n^2, \tag{6.4.2}$$

where C is the constant of Assumption 6.2.

Input: interval length $\tau > 0$, threshold $\epsilon > 0$, and ℓ queries per step.
Initialize $r := 1$.
At each step t:

1. (Follow $\mathcal{A}$.) Follow the action of an algorithm $\mathcal{A}$ satisfying Assumption 6.2.
2. (Querying policy.) If t belongs to the m-th interval except its first step, *i.e.*, if $t \in [(m-1)\tau+2, \ldots, m\tau]$, let $\Sigma_{t-1}(i)$ denote the number of queries arm α_i has received since the start of the m-th interval until step $t-1$. Order the arms $\{\alpha_1, \ldots, \alpha_n\}$ according to $\Sigma_{t-1}(1), \ldots, \Sigma_{t-1}(n)$. Query the set $\mathcal{S}_t$ of arms that received the fewest queries. Update the following elements of the empirical mean $\widehat{r}_m$:

$$\widehat{r}_m(i) := \frac{\Sigma_{t-1}(i)\,\widehat{r}_m(i) + r_{t-1}(i)}{\Sigma_{t-1}(i) + 1}, \quad \text{for every } i \in \mathcal{S}_t.$$

3. (Detect change.) At the start of the m-th interval, *i.e.*, if $t = (m-1)\tau + 1$ for some $m = 3, 4, \ldots$. If $\left\|\widehat{r}_m - \widehat{r}_r\right\|_\infty > \epsilon$, reset (*i.e.*, re-instantiate) algorithm $\mathcal{A}$ and set $r := m$. The index r denotes the last interval at which the algorithm $\mathcal{A}$ was reset.

Algorithm 6–1: Windowed mean-shift detection (WMD) algorithm

Remark 31. The WMD algorithm requires as input the threshold ϵ and the time horizon T, but it does not require prior knowledge of the number of distribution changes k.

Remark 32 (Query-regret trade-off). The bound of Theorem 6.1 indicates a way to trade-off the number of queries ℓ per step and the expected regret per step. Suppose that an increasing function C_Q assigns a cost, in the same unit as the rewards and the regret, to the rate of queries ℓ. The corresponding new objective thus becomes the sum of two components: query cost and regret. This overall expected cost-per-step at time T is $C_Q(\ell) + L_T/T$. With the implicit assumption that the bound (6.4.2) is tight in the duration T, the number of changes k, and the query rate ℓ, this cost can be optimized with respect to ℓ. If each query is assigned a constant cost c_q, *i.e.*, $C_Q(\ell) = c_q \cdot \ell$, then

the (non-discrete) optimal query rate is $\ell^* = \sqrt{(7kn/C_Q)\log(T)/(\epsilon^2 T)}$. This is the type of optimization problem that has to be resolved in Example 6.2.

Proof of Theorem 6.1. The proof is composed of five steps. In the first step, we identify the components of the regret. In the second step, we analyze the empirical means computed by the WMD algorithm. In the successive steps, we bound the components of the regret. The components are combined in the final step.

Step 1. Let $M(T)$ denote the expected number of intervals after a change-point ν_j occurs before it is detected by the WMD algorithm (*i.e.*, algorithm $\mathcal{A}$ is reset). Let $N(T)$ be the expected number of false detections up to T, *i.e.*, instances when the algorithm $\mathcal{A}$ resets when no change-point has occurred since the last time $\mathcal{A}$ was reset. Observe that the total number of times the algorithm resets is bounded from above by $k + N(T)$. Hence, over a T-step horizon, there are at most $k + N(T)$ interval-periods during which the algorithm resets once and the source distribution does not change. By Assumption 6.2, during each such period, the expected regret is of the order of $Cn\log(\Gamma)/\Delta$, where Γ is the length of the period. Since log is a concave function and $\Gamma \leq T$, the expected regret over all such periods is at most $(Cn/\Delta)(k + N(T))\log(T)$.

The algorithm may also incur regret during the delay between distribution change and its detection. Since there are k distribution changes, each occurring at most $\lceil M(T)\rceil\tau$ time steps before the algorithm $\mathcal{A}$ resets. Hence, the total regret of this algorithm is at most

$$k(M(T)+1)\tau + \frac{Cn}{\Delta}(k + N(T))\log(T). \tag{6.4.3}$$

Next, we bound $N(T)$ and $M(T)$, starting with $N(T)$. Observe that the term $k\tau$ accounts for the regret within intervals during which a change occurs.

Hence, for the remainder of the proof, we consider only the intervals that do not contain a distribution change.

Step 2 (Empirical means). Consider the empirical mean over each interval that does not contain a change. Let $\gamma = \tau/\lfloor \frac{n}{\ell} \rfloor = \lfloor \frac{\log(T)}{2\epsilon^2} \rfloor$. Observe that by the construction of the WMD algorithm, after the end of the m-th interval, spanning time steps $(m-1)\tau + 1, \ldots, m\tau$, every arm is queried either γ or $\gamma + 1$ times. In the former case, the empirical mean for an arm α_i is the mean of γ i.i.d. random variables

$$\widehat{r}_m(i) = \frac{1}{\gamma} \sum_{t=(m-1)\tau+1}^{m\tau} r_t(i) \cdot \mathbf{1}_{[i \in \mathcal{S}_t]}.$$

The expression for the latter case (of $\gamma + 1$ queries) is similar, and is omitted in this proof.

Step 3 (Number of false detections). Suppose that in the opponent action sequences during the m-th and r-th intervals are generated from the same distribution with expected value denoted, with an abuse of notation, by ρ_m. Observe that

$$\begin{aligned} N(T) &\leq \lceil T/\tau \rceil P\left(\|\widehat{r}_m - \widehat{r}_r\|_\infty > \epsilon \right) \\ &\leq n \lceil T/\tau \rceil \min_{i=1,\ldots,n} P\left(|\widehat{r}_m(i) - \widehat{r}_r(i)| > \epsilon \right), \end{aligned}$$

since there are at most $\lceil T/\tau \rceil$ intervals. Observe that, for every $i = 1, \ldots, n$,

$$\begin{aligned}
&P\left(|\widehat{r}_m(i) - \widehat{r}_r(i)| > \epsilon\right) \\
&= P\left(|\widehat{r}_m(i) - \widehat{r}_r(i)| > \epsilon, |\widehat{r}_m(i) - \rho_m(i)| \leq \epsilon\right) \\
&\quad + P\left(|\widehat{r}_m(i) - \widehat{r}_r(i)| > \epsilon, |\widehat{r}_m(i) - \rho_m(i)| > \epsilon\right) \\
&\leq P\left(|\widehat{r}_m(i) - \widehat{r}_r(i)| > \epsilon \mid |\widehat{r}_m(i) - \rho_m(i)| \leq \epsilon\right) \\
&+ P\left(|\widehat{r}_m(i) - \rho_m(i)| > \epsilon\right) \\
&\leq P\left(|\widehat{r}_r(i) - \rho_m(i)| > 2\epsilon\right) + P\left(|\widehat{r}_m(i) - \rho_m(i)| > \epsilon\right) \qquad (6.4.4) \\
&\leq \exp\left(-8\gamma\epsilon^2\right) + \exp\left(-2\gamma\epsilon^2\right),
\end{aligned}$$

where the last inequality follows by Step 2 and Hoeffding's Inequality (recall that $\widehat{r}_r(i)$ and $\widehat{r}_m(i)$ have the same distribution). Hence, we have

$$N(T) \leq 2n \exp\left(-2\gamma\epsilon^2\right)(T/\tau + 1). \qquad (6.4.5)$$

Step 4 (Delay in change detection). Next, we bound $M(T)$. Suppose that there is a reset at the s-th interval. The WMD algorithm compares successively the empirical means $\widehat{r}_{s+1}, \widehat{r}_{s+2}, \ldots$ to $\widehat{r}_s$. Suppose that the following change occurs during the $(m-1)$-th interval. Let ρ_m and ρ_s denote the expected reward vectors during m-th and s-th intervals, respectively. By the same argument as the first occurrence of an event in an i.i.d. random sequence, we have

$$M(T) \leq 1 \Big/ P\left(\|\widehat{r}_m - \widehat{r}_s\|_\infty > \epsilon\right). \qquad (6.4.6)$$

Observe that, for every $i = 1, \ldots, n$,

$$\begin{aligned}
&P\left(\|\widehat{r}_m - \widehat{r}_s\|_\infty > \epsilon\right) \\
&\geq P\left(|\widehat{r}_m(i) - \widehat{r}_s(i)| > \epsilon\right) \\
&\geq P\left(|\widehat{r}_s(i) - \rho_m(i)| > 3\epsilon/2, |\widehat{r}_m(i) - \rho_m(i)| \leq \epsilon/2\right) \\
&= P\left(|\widehat{r}_s(i) - \rho_m(i)| > \frac{3\epsilon}{2}\right) P\left(|\widehat{r}_m(i) - \rho_m(i)| \leq \frac{\epsilon}{2}\right) \\
&\geq P\left(|\widehat{r}_s(i) - \rho_m(i)| > \frac{3\epsilon}{2}\right)\left(1 - e^{-\gamma\epsilon^2/2}\right), \qquad (6.4.7)
\end{aligned}$$

where the equality is due to independence, and the final inequality follows by Hoeffding's Inequality. Next, we bound the first term of (6.4.7). Suppose, without loss of generality, that $\rho_s(i) > \rho_m(i)$; let δ denote $\rho_s(i) - \rho_m(i)$; we obtain, for some i:

$$\begin{aligned}
&P\left(|\widehat{r}_s(i) - \rho_m(i)| > 3\epsilon/2\right) \\
&= P\left(|\widehat{r}_s(i) - \rho_s(i) + \delta| > 3\epsilon/2\right) \\
&= 1 - P\left(\delta - 3\epsilon/2 \leq \widehat{r}_s(i) - \rho_s(i) \leq \delta + 3\epsilon/2\right) \\
&\geq 1 - P\left(\delta - 3\epsilon/2 \leq \widehat{r}_s(i) - \rho_s(i)\right) \\
&\geq 1 - \exp(-2\gamma(\delta - 3\epsilon/2)^2) \geq 1 - \exp(-\gamma\epsilon^2/2), \qquad (6.4.8)
\end{aligned}$$

where the last two inequalities follows from the fact that $\delta = \rho_s(i) - \rho_m(i) > 2\epsilon$ for some i by Assumption 6.3. Hence, (6.4.7), (6.4.8) and (6.4.6) give

$$M(T) \leq 2 \big/ \left(1 - \exp\left(-\gamma\epsilon^2/2\right)\right). \qquad (6.4.9)$$

Step 5 (Tying up). By combining (6.4.3) with (6.4.5) and (6.4.9), we find that expected regret is at most:

$$\frac{2k\tau}{(1 - \exp(-\gamma\epsilon^2/2))} + k\tau + \frac{Cn}{\Delta}\Big(k + 2n\exp\left(-2\gamma\epsilon^2\right)(T/\tau + 1)\Big)\log(T),$$

from which (6.4.2) follows by substituting the values $\tau = \lfloor \frac{n}{\ell} \rfloor \cdot \lfloor \frac{\log(T)}{2\epsilon^2} \rfloor$ and $\gamma = \lfloor \frac{\log(T)}{2\epsilon^2} \rfloor$. ■

6.5 A Lower Bound

In the previous section, we showed an upper bound on the regret using a simple mean-shift detection scheme. In this section, we present a lower bound on the regret for a class of algorithms. This class consists of algorithms where the expected number of changes in the sequence of selected arms is bounded by the number of distribution changes up to an additive constant. This class encompasses algorithms that detect changes and reset, such as the WMD algorithm. Another example of algorithm of this class uses a combination of optimal change-point detection (*e.g.*, the Shiryayev-Roberts scheme) and optimal sampling in multi-armed bandits. We show that the lower-bound for this class of algorithms matches the upper-bound for the WMD algorithm.

More precisely, we consider the class of algorithms that generate an action sequence $a_1, a_2, \ldots$ satisfying the following constraint:

$$\mathbb{E}\left[\sum_{t=1}^{T-1} 1_{[a_t \neq a_{t+1}]}\right] \leq k + c, \tag{6.5.10}$$

for all T and some constant c. This constraint is motivated by the fact that if the entire sequence of expected reward vectors $\rho_1, \rho_2, \ldots$ is known a priori, then the optimal algorithm also satisfies the constraint of (6.5.10).

6.5.1 Lower Bound for Change-Detection

In this section, we present a result (cf. [107]) on the expected run lengths of the SR method for change-detection that will be used subsequently. Consider a sequence of random variables $r_1, \ldots, r_{\nu-1}, r_\nu, r_{\nu+1}, \ldots$, where $r_1, \ldots, r_{\nu-1}$ are i.i.d. with distribution f_0, whereas $, r_\nu, r_{\nu+1}, \ldots$ are i.i.d. with distribution f_θ. We assume that ν is unknown, as well as f_θ. However, we assume that f_0 is known and that f_0 and f_θ belong to an exponential family of distributions.

Assumption 6.4. Every element of the random vectors $r_1, r_2, \ldots$ is distributed according to a distribution belonging to the exponential family of distributions:

$$f_y(x) = e^{yx-\psi(y)}, \quad y \in \Omega,$$

where Ω is some interval of $\mathbb{R}$, and $\psi(0) = \psi'(0) = 0$.

Examples of distribution families satisfying Assumption 6.4 include Bernoulli, geometric, and beta distributions.

The following theorem states that in order to have an average run-length (ARL) to a false detection of at least V, the ARL to detection is of the order of $\Theta(\log V)$. This result is a consequence of [107, Theorem 1 and 4] and [106].

Theorem 6.2 (Average run-lengths [106, 107])**.** *Suppose that a change-detection algorithm raises an alarm at time U and that $\mathbb{E}_\infty[U] \geq V$ when there is no change-point,* i.e., *$\nu = \infty$. Then, if $\nu = 1$,* i.e., *a change occurs at time 1, and θ is in the interior of Ω, then*

$$\mathbb{E}_1[U] \geq C_1[\log V + \log\log V + C_2], \tag{6.5.11}$$

where C_2 is a constant, $C_1 = 1/D(f_\theta; f_0)$, and D denotes the Kullback-Leibler divergence.

Let G be a probability measure on Ω such that $G(\{0\}) = 0$, and its first derivative G' exists, is positive and continuous in the interior of Ω. The following change-detection stopping-time satisfies the lower bound (6.5.11) with equality:

$$U(V, G) = \min\left\{t : \int \sum_{k=1}^{t} \exp\Big(y \sum_{i=k}^{t} r_i - (t-k+1)\psi(y)\Big) dG(y) \geq V\right\}, \quad V > 0.$$

6.5.2 Lower Bound for the Regret

Theorem 6.3 (Regret lower-bound)**.** *For every fixed algorithm of our class (fixed ℓ and query mechanism), there exists a piecewise-stationary source such*

that detect-and-react algorithm has a regret of at least

$$L_T \geq C_1 k \frac{n}{\ell}[\log T + \log\log T + C_2].$$

where C_1 and C_2 are the same constants as in Theorem 6.2.

Proof. We consider a Bernoulli-distributed piecewise-stationary sequences of reward vectors, so that detecting changes in the mean is equivalent to detecting changes in distribution. Hence, Theorem 6.2 holds.

Consider an arbitrary interval from ν_{j-1} to ν_{j+1} for $j = 1, \ldots, k-1$. We denote the length of this interval by $\tau_j = \nu_{j+1} - \nu_{j-1}$. Let X_j denote the expected delay in detecting the change-point ν_j. We can easily construct examples where the expected regret is at least

$$L_T \geq \sum_{j=1}^{k-1} X_j.$$

By Theorem 6.2, if an algorithm satisfies the constraint of (6.5.10) (a constant expected number of false detections over a horizon of T time steps), the expected delay to detection X_j is at least

$$X_j^2 \triangleq C_1 \frac{n}{\ell}[\log T + \log\log T + C_2].$$

where the factor n/ℓ comes from the fact that we sample only ℓ of the n arm rewards at each time step whereas Theorem 6.2 assumes that all n arms are observed at each time step. Hence, the result follows. ■

6.6 Simulations

In this section, we present an empirical comparison of the WMD algorithm with other algorithms for multi-armed bandit problems. As reference, we consider two algorithms based on upper confidence bounds: the UCB1

algorithm of [9], and the Discounted-UCB algorithm of [78, 58]. For comparison purpose, we employ the UCB1 algorithm as the component $\mathcal{A}$ of the WMD algorithm. The resulting combination is referred to as the WMD-UCB algorithm.

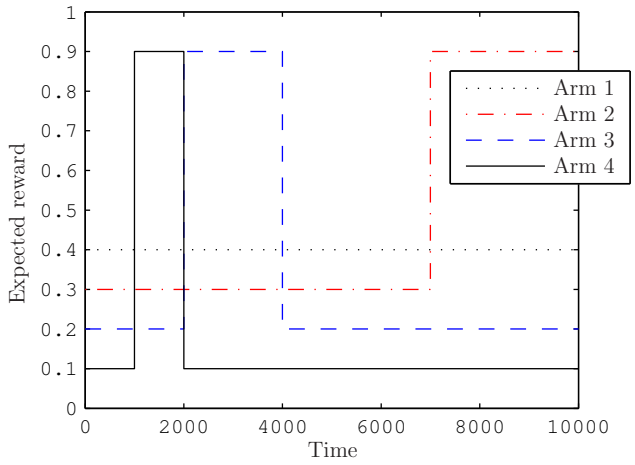


Figure 6–1: Expected reward of the arms of a 4-armed bandit.

For our setting, we take a bandit with 4 arms, whose rewards are piecewise-stationary with Bernoulli distributions. The sequence of expected rewards $\rho_t(i)$ for each arm α_i is illustrated in Figure 6–1. The Discounted-UCB algorithm is provided with prior knowledge of the number k of changes to come in the reward sequence, and its parameters are accordingly set to optimal values [58]. Neither the UCB1, nor the WMD-UCB algorithm require this prior information. However, the WMD-UCB algorithm has the privilege to query the previous rewards of some of the arms.

Figure 6–2 shows the evolution of the average reward of the three algorithms. In this experiment, the WMD-UCB algorithm queries only one arm per step; its average reward is close to optimal with respect to the baseline of (6.2.1). Figure 6–3 illustrates the benefit of increasing the number of queries per step of the WMD-UCB algorithm (with the interval length τ held fixed).

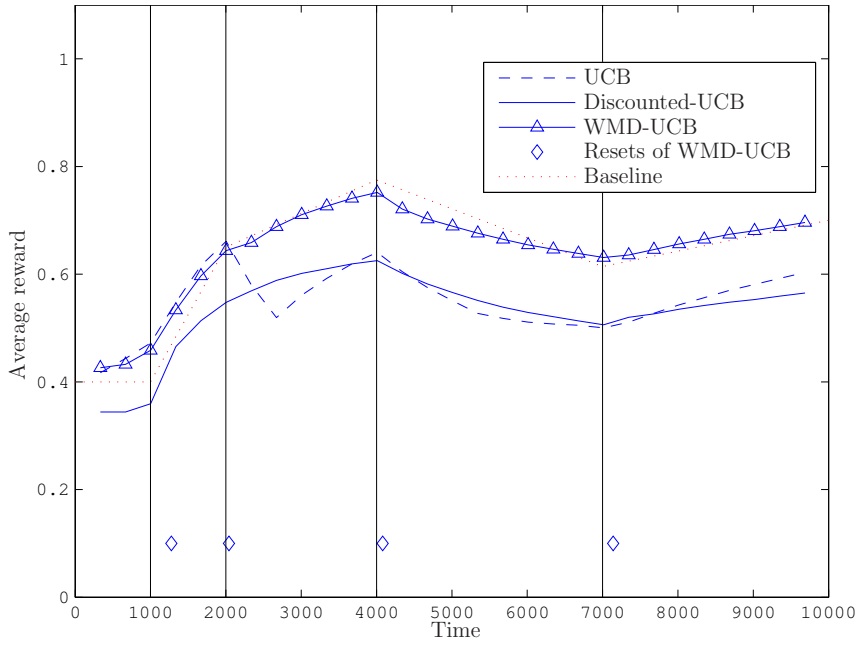


Figure 6–2: Average expected regret of UCB, Discounted-UCB, and WMD-UCB against the bandit of Figure 6–1. The WMD-UCB uses the threshold $\epsilon = 0.3$ and makes 1 query per step. Changes in the reward sequence distribution are indicated by vertical lines, whereas instants at which the WMD-UCB algorithm resets are indicated by diamonds. The baseline of our notion of regret (6.2.1) is also plotted.

6.7 Discussions

The WMD algorithm uses a very simple scheme to detect changes in the mean. In its place, we may employ more sophisticated change-detection schemes, *e.g.*, CUSUM [103] and the Shiryayev-Roberts rule [119]. Modifications are nonetheless required to make them applicable to our problem: the reward distributions must be parametrized; and the pre-change distribution is unknown and must be estimated (cf. [95]). There also exist schemes that detect changes when the reward process follows one of many Markovian processes [57], as is the case for restless bandit problems. Despite the drawback of complexity, these schemes detect changes with optimal delay, and do not require prior knowledge of the parameter ϵ of Assumption 6.3. Yet, they also incur a regret of the order of $\log(T)$ due to an inevitable logarithmic delay to

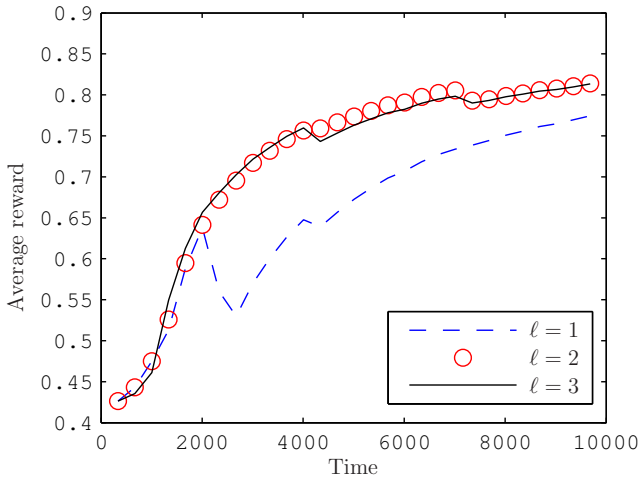


Figure 6–3: Average expected regret of the WMD-UCB algorithm with 1, 2, and 3 queries per step against a 4-armed bandit. The threshold parameter ϵ is 0.2.

detection [88, 107]. This is the basis of the lower-bound on the regret of the class of algorithms described in Section 6.5, which includes algorithms that detect the unknown changes and then react.

The side information obtained through queries can be applied to two purposes: detecting changes and improving the performance of the multi-armed bandit algorithm of Assumption 6.2. In this work, the queries serve only the purpose of change detection. Because of the aforementioned lower regret-bound intrinsic to change detection schemes, we have neglected the question of accelerating the exploration of the sub-algorithm of Assumption 6.2. The action elimination method of [47] presents another possible improvement to the sub-algorithm of Assumption 6.2. As a further improvement to the detection component of the WMD algorithm, it is sufficient, when the distribution changes are not adversarial, to limit detections to changes where the current best arm is no longer the best.

CHAPTER 7
Unimodal Piecewise-Stationary Bandits

7.1 Introduction

In this chapter, we consider another version of the piecewise-stationary bandit problem of Chapter 6. The main differences are the following. First, we allow the set of arms to be an interval of the real line. Second, we employ a new strategy of detecting change-points that does not require queries for side observations. The notion of regret, the objective, and the presence of abrupt changes in the reward distribution at unknown time instants, remain the same as in Chapter 6.

In order avoid a regret dependent on the number of arms, it is necessary to offset the infinite number of arms by an assumption of dependence between the reward distributions of the arms. In addition to a standard Lipschitz (or Hölder) continuity condition (*e.g.*, [3]), we also assume that the expected reward function is unimodal over the set of arms. However, we do not require convexity or differentiability as is the case in [75, 37]. In this setting, we present an algorithm with a finite-time regret bound that is tight up to a logarithmic factor of the time.

An additional consequence of the unimodality assumption is the possibility of detecting abrupt changes in the arms' reward distributions efficiently. We present a method that simultaneously detects change-points occurring at unknown instants and minimizes the regret. This method chooses arms in the neighbourhood of the optimal arm so as to trade-off the regret due to choosing suboptimal arms and the regret due to the delay in detecting abrupt changes.

We consider a model that is partly more general in the sense that the sequence of change-points dictates whether the environment is more stochastic or more adversarial. Our model is also partly more specific because we assume that the expected reward is unimodal on the continuum of arms. Beside bandit problems with intrinsic unimodality, this model also suggests a new solution method when the set of arms can be partitioned into subsets where the unimodal condition holds.

We organize this chapter as follows. In Section 7.2, we motivate our model and survey related works. We present the details of our model through assumptions and an example in Section 7.3. In Section 7.4, we present a solution for the setting of unimodal bandits without change-points, using a sampling scheme that finds an approximately correct arm with high probability. We give both a bound on the expected regret and a probably approximately correct (PAC) guarantee. In addition to a continuum set of arms, we also consider finitely many arms (Section 7.4.4). and arms represented by vertices in a graph (Section 7.4.5). We define in Section 7.5 the notion of changes in the reward distributions and present an algorithm that simultaneously minimizes regret and detects changes.

7.2 Related Works and Motivation

Multi-armed bandit problems are central to machine learning and have numerous applications. With a finite set of arms, index-based solutions exist, whether the rewards are stochastic or adversarial [9, 10]. The case of a continuum of arms has become the object of great interest due to applications, such as the design of auction mechanisms [27, 76] and routing [14]. The one-dimensional case was first studied in [3] with a Hölder condition. With similar conditions, Kleinberg presents an algorithm based on discretization that achieves a regret of $O(T^{2/3})$, which is optimal up to a logarithmic factor

[75]. In this work, we obtain a regret of the order of $O(\sqrt{T}\log(T))$ under an additional unimodality assumption. This regret is of the same order as [11], which also considers the model of [75] with additional assumptions. The case of arms in more general metric spaces and topological spaces is studied in [77, 31]. The critical assumption in bandit problems with an infinite number of arms is dependence between the rewards of nearby arms. The notion of dependence between arm rewards also helps in the case of bandits with finitely many arms, as shown in [104, 98].

Our model is a special case of the model of [75], where we assume that the expected reward is unimodal over the set of arms. This is similar to one of the assumptions of [37], with the notable difference that we do not require the expected reward function to be three-times differentiable. Cope considers a multidimensional unimodal expected reward function and shows that the Kiefer-Wolfowitz stochastic approximation algorithm achieves a regret of the order of $O(T^{1/2})$. It is however well-known that Kiefer-Wolfowitz type algorithms require suitable differentiability assumptions and no convergence rate result is available for these algorithms [3]. Correspondingly, the regret guarantees of [37] are asymptotic. In contrast, we show finite-time bounds with explicit constants using a particularly efficient algorithm based on a simple one-dimensional line search method combined with an appropriate sampling method. Our algorithm iteratively eliminates subsets of arms. This approach is reminiscent of algorithms such as the successive elimination algorithm for the case of finite bandit problems [46].

Bandit problems with abrupt changes at unknown time instants of the reward distributions is a generalization of two important models. The stochastic bandit corresponds to case without changes, whereas the adversarial bandit corresponds to the case of changes at each time instant. This generalization

is similar to the non-stationary bandit problem of Chapter 6. However, our model differs in having a continuum of arms and the unimodality of expected rewards. Our solution technique is completely different, relying on a simple and efficient sampling scheme. We do not require additional assumptions such as side observations (as in Chapter 6), or knowledge of the frequency of changes [58].

7.3 Assumption of Unimodality

Because of the infinite number of arms considered in our model, we need a dependence assumption between the arms' expected rewards. In this section, we state and motivate our assumption of unimodal rewards. The notion of change-points in the sequence of reward distribution functions is the same as in [130]. For simplicity, we only introduce change-points in Section 7.5.

We consider a bandit problem where the set of arms is the interval $[0, 1]$ of the real line. As an example, consider a sequential pricing problem with the goal of maximizing the total revenue from the sale of a sequence of identical items. We may think of the arms as the possible prices for an item. At each time instant t, the agent chooses a price x_t from an interval $[0, 1]$. The reward of the arm $x \in [0, 1]$ at time t is a random variable

$$r_t(x) = x\ w_{t,x}.$$

For a fixed x, the sequence $w_{1,x}, w_{2,x}, \ldots$ is a sequence of i.i.d. Bernoulli random variables. Each Bernoulli random variable $w_{t,x}$ corresponds to whether an item is sold at time t. Hence, for a fixed x, the sequence $r_1(x), r_2(x), \ldots$ is also an i.i.d. random sequence. The expected reward of a fixed arm x at every time t is

$$\bar{r}(x) \triangleq \mathbb{E}[x\ w_{t,x}] = x \Pr(w_{t,x} = 1), \quad x \in [0, 1]$$

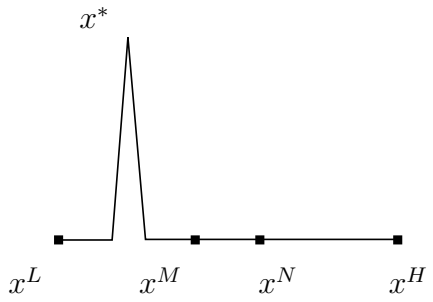


Figure 7–1: Cases to avoid: sharp peak and flat plateau.

which is independent of t. The function $\bar{r}$ is the *expected reward function* over the set of arm $[0, 1]$.

We do not know the probability $\Pr(w_{t,x} = 1)$ of sale of the item (*i.e.*, success) for each price x. However, we assume that the success probabilities are monotonously decreasing in the price. Hence, these probabilities are ordered: if we have two arms x and y such that $x \leq y$, then

$$\Pr(w_{*,y} = 1) \leq \Pr(w_{*,x} = 1).$$

This leads to the following unimodality property on the expected reward function $\bar{r}$, which is the main assumption of this chapter.

Assumption 7.1 (Unimodal)**.** The expected reward function $\bar{r}$ is unimodal, *i.e.*, there exists an *optimal arm* $x^* \in [0, 1]$ such that $\bar{r}$ is monotonically increasing in the interval $[0, x^*]$, and monotonically decreasing in the interval $[x^*, 1]$.

Observe that Assumption 7.1 also ensures the uniqueness of the optimal arm x^*.

We also need the following assumption to avoid the situations depicted in Figure 7–1. First, we need an upper bound on the rate of increase of $\bar{r}$ to avoid sharp peaks that induce high regret even when we choose arms very close to the best arm. Second, we need a lower bound on the rate of increase

of $\bar{r}$ to avoid intervals where there is too little separation between expected reward values.

Assumption 7.2 (Strong max)**.** Suppose that $\bar{r}$ is unimodal with the maximum at x^*. The function $\bar{r}$ has a *strong maximum* at x^*, *i.e.*, there exist positive Lipschitz constants $C_L > 0$ and $C_H > 0$ such that

$$C_L \left| x - y \right| \leq \left| \bar{r}(x) - \bar{r}(y) \right| \leq C_H \left| x - y \right|,$$

for all pairs $x, y \in [0, x^*]$ or $x, y \in [x^*, 1]$.

Remark 33. Assumption 7.2 is a strong assumption that excludes functions that are smooth around the maximum.

Suppose that an algorithm generates the sequence of actions $x_1, x_2, \ldots$. The corresponding total expected reward is $\sum_{t=1}^{T} \bar{r}(x_t)$. The *unimodal regret* of this algorithm is

$$L_T^{\mathrm{U}} \triangleq \sum_{t=1}^{T} (\bar{r}(x^*) - \mathbb{E}\bar{r}(x_t)).$$

This notion of average expected regret is similar to that of [85, 9]. We use this notion of regret in Section 7.4.

When the sequence of reward distributions contains change-points, we do not have a fixed expected reward function $\bar{r}$ for all time instants. In that case, we use the following notion of regret:

$$L_T^{\mathrm{UPS}} \triangleq \left(\sum_{t=1}^{T} \max_{x \in [0,1]} \bar{r}_t(x) \right) - \left(\sum_{t=1}^{T} \mathbb{E}\bar{r}_t(x_t) \right),$$

where $\bar{r}_t$ denotes the expected reward function at time t. The baseline of comparison for the above regret is the average of *optimal* expected rewards, which differs entirely from the baseline used in the adversarial bandit problem (cf. [10]). We call this the *unimodal piecewise-stationary regret* and use it in Section 7.5.

7.4 Unimodal Bandit Without Changes

In this section, we consider the bandit problem when there are no change-points. First, we present a simple algorithm for the stochastic multi-armed bandit problem with a probably approximately correct (PAC) guarantee. The sampling algorithm (Algorithm 7–1) works on a finite set of arms $\{1, \ldots, m\}$. It takes two parameters ϵ and δ as input and samples the arms sequentially in the order

$$\text{arm } 1,\ \text{arm } 2, \ldots,\ \text{arm } m,\ \text{arm } 1,\ \text{arm } 2, \ldots,$$

and stops after $\frac{4m}{\epsilon^2}\log(\frac{2m}{\delta})$ time steps.

(Initialization.) Fix $\epsilon > 0$ and $\delta > 0$.

1. Sample all arms $1, \ldots, m$ sequentially, until each arm has been sampled $\frac{4}{\epsilon^2}\log(\frac{2m}{\delta})$ times.
2. Let the sample-average reward of arm i be denoted by $\widehat{r}(i)$. Output the arm $\arg\max_{i=1,\ldots,m} \widehat{r}(i)$.

Algorithm 7–1: Sampling algorithm

Theorem 7.1 (Theorem 1 of [46]). *With probability $1-\delta$, the sampling algorithm outputs an arm j that is ϵ-optimal,* i.e.*, which has average reward*

$$\bar{r}(j) \geq \max_{i=1,\ldots,m} \bar{r}(i) - \epsilon.$$

We also say that the output arm of the sampling algorithm is (ϵ_u, δ_u)-PAC.

Next, we present our algorithm for the unimodal bandit problem, along with its performance guarantees.

7.4.1 The LSE algorithm

Consider the LSE algorithm (Algorithm 7–2). At every iteration of the main loop, it narrows down the *sampling interval* $[x^L, x^H]$ in which the true optimal arm x^* lies with high probability. During each iteration, the LSE algorithm runs the sampling algorithm over four arms x^L, x^M, x^N, x^H. These

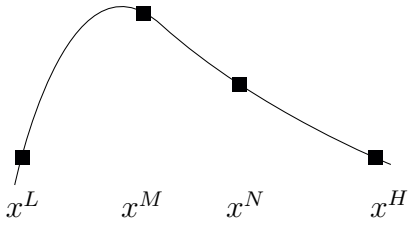


Figure 7–2: The four sampled arms and their expected rewards during one iteration. The interval $[x^N, x^H]$ is eliminated with high probability at the end of this iteration. If $|x^H - x^L| = 1$, then the distances between the points are: $|x^M - x^L| = \varphi^{-2}$, $|x^N - x^M| = \varphi^{-3} = (2/\varphi - 1)$, and $|x^H - x^N| = \varphi^{-2}$.

four arms are chosen as in Kiefer's golden section search algorithm [74]. Three of these arms are kept from one iteration to the next; the fourth arm is new. At the end of each iteration, the algorithm narrows down the sampling interval by a constant factor $1/\varphi$, where φ is the golden ratio. Figure 7–2 illustrates the four arms and their true expected rewards during one iteration. Over time, the goal is to eliminate intervals that do not contain the optimal arm with high probability, such as $[x^N, x^H]$ in Figure 7–2, at the end of each iteration, and hence narrow down to smaller and smaller sampling intervals what contain the optimal arm x^* with high probability.

Remark 34 (Notation). Although the arms x^L, x^M, x^N, x^H change from one iteration to another, for simplicity of notation, we shall sometimes omit the subscript n. Hence, we write x^L instead of x_n^L when the n-th iteration is implicitly understood.

Remark 35. Observe that, from one iteration of the LSE algorithm to the next, three of the four arms do not change. These arms have samples from previous iterations, and hence, we can save on sampling by reusing these samples. Unfortunately, however, we can not detect change-points at the same time as recycling samples. This will become clear in Section 7.5.

(Initialization.) Set $[x^L, x^H] = [0, 1]$. Set x^M such that $\frac{x^H - x^M}{x^M - x^L} = \varphi$ (cf. Figure 7–2), where φ is the golden ratio. Set x^N in $[x^M, x^H]$ such that $\frac{x^N - x^L}{x^H - x^N} = \varphi$.
For iterations $n = 1, 2, \ldots$:

1. (Find (ϵ_n, δ_n)-PAC arm.) Use the sampling algorithm (Algorithm 7–1) on the arms $\{x^L, x^M, x^N, x^H\}$ with parameters ϵ_n and δ_n, returning the (ϵ_n, δ_n)-PAC arm x_n^*.
2. (Interval elimination.)
   - If $x_n^* = x^N$ or $x_n^* = x^H$, then eliminate the interval $[x^L, x^M]$. Update the points $x^L := x^M$ and $x^M := x^N$, and $x^N = (x^L + \varphi x^H)/(1 + \varphi)$.
   - Else, eliminate $[x^N, x^H]$. Update the points $x^H := x^N$, $x^N := x^M$, and $x^M = (\varphi x^L + x^H)/(1 + \varphi)$.

Algorithm 7–2: Line search elimination (LSE) algorithm

7.4.2 Regret Bound

Next, we show that the regret of the LSE algorithm is of the order of $O(\sqrt{T}\log(T))$.

Theorem 7.2 (Expected regret of LSE)**.** *Suppose that Assumptions 7.1 and 7.2 hold. Let T be fixed and known. Let $\delta_n = 8/T$ and $\epsilon_n = (2/\varphi - 1)C_L/\varphi^n$ for every interval n. Then the expected regret of the LSE algorithm is*

$$L_T^{\mathrm{U}} \leq \frac{32\varphi}{(2/\varphi - 1)^2(\varphi - 1)} \frac{C_H}{C_L^2} \sqrt{1 + C_L^2 T}\log(T) + 2\log_\varphi^2(1 + C_L^2 T),$$

where $\varphi = (1 + \sqrt{5})/2$ is the golden ratio.

Remark 36. Other algorithms exist that have similar or better performance guarantees. For example, in a more general setting that does not assume unimodality, the HOO algorithm of [31] achieves a regret of the order of $O(\sqrt{T\log T})$ without prior knowledge of the time horizon T.

Remark 37. This upper bound is independent of the number of arms and the difference between the best and second-best distributions, but depends on the

Lipschitz constants. Moreover, it matches the lower bounds of [11, 37] up to a logarithmic factor.

Remark 38. For this and the subsequent results, we assume that the time horizon T is known. This assumption can be easily removed by employing the doubling trick [34]. This trick consists of starting with an arbitrary horizon, then, at the end of that horizon, we reset and restart our algorithm with a new horizon twice as long.

First, we prove the following lemma.

Lemma 7.3. *Suppose that the assumptions of Theorem 7.2 hold. Then, at the n-th interval of the LSE algorithm, we have:*

$$\Pr(x^* \notin [x_n^L, x_n^H]) \leq \sum_{i=1}^{n} \delta_i.$$

Proof. We can write the above probability recursively:

$$\begin{aligned}
&\Pr(x^* \notin [x_n^L, x_n^H]) \\
&= \Pr(\{x^* \notin [x_{n-1}^L, x_{n-1}^H]\} \wedge \{x^* \notin [x_n^L, x_n^H]\}) \\
&+ \Pr(\{x^* \in [x_{n-1}^L, x_{n-1}^H]\} \wedge \{x^* \notin [x_n^L, x_n^H]\}) \\
&= \Pr(x^* \notin [x_{n-1}^L, x_{n-1}^H]) \\
&+ \Pr(\{x^* \in [x_{n-1}^L, x_{n-1}^H]\} \wedge \{x^* \notin [x_n^L, x_n^H]\}). \qquad (7.4.1)
\end{aligned}$$

Let τ_n denote the length of the n-th iteration. During the n-th iteration, we employ the sampling algorithm to find an ϵ_n-optimal arm with probability $1 - \delta_n$, denoted x_n^*. Since $\tau_n \geq \frac{16}{\epsilon_n^2} \log(8/\delta_n)$, by Theorem 7.1, the sampling algorithm outputs an ϵ_n-optimal arm with probability $1 - \delta_n$.

Observe that, by definition of the golden section search algorithm of [74],

$$\min\{x_n^M - x_n^L, \quad x_n^N - x_n^M, \quad x_n^H - x_n^N\} = (2/\varphi - 1)/\varphi^n.$$

Suppose that the arm x_n^* is x^L or x^M, then by Assumption 7.2 and the fact that $\epsilon_n = (2/\varphi - 1)C_L/\varphi^n$, the true optimal arm x^* lies in the interval $[x^N, x^H]$ with probability at most δ_n. Similarly, if the arm x_n^* is x^N or x^H, then the optimal arm x^* lies in the interval $[x^L, x^M]$ with probability at most δ_n. Hence, the LSE algorithm eliminates the optimal with probability at most δ_n, *i.e.*,

$$\Pr(\{x^* \in [x_{n-1}^L, x_{n-1}^H]\} \wedge \{x^* \notin [x_n^L, x_n^H]\}) \leq \delta_n.$$

Substituting this in (7.4.1) and carrying-out the recursion, we obtain the claim.

■

7.4.3 PAC Guarantee

Observe that the LSE algorithm continues exploration (narrowing down the sampling interval $[x^L, x^H]$) until the end of the time horizon T. A natural alternative is the following: once the sampling interval $[x_n^L, x_n^H]$ is small enough, we may cease the sampling and elimination process and exploit arms from that interval. Hence, we have a modified version of the LSE algorithm, which follows the LSE algorithm until the beginning of the n-th iteration, and thereafter, picks one of the arms in the interval $[x_n^L, x_n^H]$. We call this modified algorithm the *stopped* LSE algorithm. Next, we present a PAC guarantee for the performance of the stopped LSE algorithm, in the spirit of [46].

Theorem 7.4 (PAC arm)**.** *Suppose that we run the LSE algorithm with parameters $\epsilon_n \leq (2/\varphi - 1)C_L/\varphi^n$ and δ_n. for $n = 1, 2, \ldots$. Let $\tau_n = \frac{16}{\epsilon_n^2}\log(\frac{8}{\delta_n})$ denote the length of the n-th iteration. Then, after $\sum_{i=1}^n \tau_i$ time steps, every arm in the interval $[x_n^L, x_n^H]$ has $\frac{C_H}{\varphi^n}$-optimal expected reward with probability $1 - \sum_{i=1}^n \delta_i$.*

Proof. Since $\epsilon_n \leq (2/\varphi - 1)C_L/\varphi^n$, by Theorem 7.1, we have enough samples at the n-th iteration to find the best arm from the four arms $\{x_n^L, x_n^M, x_n^N, x_n^H\}$ with probability δ_n.

By Lemma 7.3, we have

$$\Pr(x^* \notin [x_n^L, x_n^H]) \leq \sum_{i=1}^{n} \delta_i.$$

Hence, after $\sum_{i=1}^{n} |\tau_i|$ time steps, with probability $1 - \sum_{i=1}^{n} \delta_i$, we have

$$x^* \in [x_n^L, x_n^H]$$

in which case, by Assumption 7.2, every arm in the interval $[x_n^L, x_n^H]$ is an $\frac{C_H}{\varphi^n}$-optimal arm. ■

Corollary 7.5. *Suppose that the assumptions of Theorem 7.4 hold. Let T be a fixed and known integer. After*

$$\frac{\varphi^{2n+2} - 1}{\varphi^2 - 1} \frac{16}{(2/\varphi - 1)^2 C_L^2} \log(T)$$

time steps, every arm in the interval $[x_n^L, x_n^H]$ has $\frac{C_H}{\varphi^n}$-optimal expected reward with probability $1 - n/T$.

Proof. Observe that if we let $\epsilon_n = (2/\varphi - 1)C_L/\varphi^n$ and $\delta_n = 8/T$, then we have

$$\begin{aligned} \sum_{i=1}^{n} \tau_i &= \sum_{i=1}^{n} \frac{16}{(2/\varphi - 1)^2 C_L^2} \varphi^{2n} \log(T) \\ &= \frac{\varphi^{2n+2} - 1}{\varphi^2 - 1} \frac{16}{(2/\varphi - 1)^2 C_L^2} \log(T). \end{aligned}$$

■

7.4.4 Unimodal Bandit with Finitely Many Arms

We consider in this section a special case of the unimodal bandit with a finite ordered set of arms $(v_1, \ldots, v_m)$. Accordingly, we modify the unimodality assumption as follows.

Assumption 7.3 (Finite unimodality)**.** The expected reward function $\bar{r}$ over $(v_1, \ldots, v_m)$ is unimodal with a maximum at v_k, *i.e.*,

$$\begin{aligned} \bar{r}(v_i) \leq \bar{r}(v_k) \quad & \text{for all } i < k \\ \text{and} \quad \bar{r}(v_j) \geq \bar{r}(v_k) \quad & \text{for all } j > k. \end{aligned}$$

With finitely many arms, we also need a version of the strong maximum assumption that does not depend on distances. The following assumption bounds on the separation in expect rewards of adjacent arms.

Assumption 7.4 (Finite strong max)**.** There exist positive constants D_L and D_H such that

$$D_L \leq |\bar{r}(v_i) - \bar{r}(v_{i+1})| \leq D_H,$$

for every pair of adjacent arms v_i and v_{i+1}.

The LSE algorithm can be modified for finitely many arms as follow. To each arm v_i of $(v_1, \ldots, v_m)$, we assign the interval $[\frac{i-1}{m}, \frac{i}{m}) \subset [0, 1]$, for $i = 1, \ldots, m$. Then, we replace every reference in the LSE algorithm to an arm x in the interval $[\frac{i-1}{m}, \frac{i}{m})$ by the arm v_i. With this modification, we can show the following $O(\log T)$ bound on the regret.

Theorem 7.6 (Finite unimodal bandit)**.** *Suppose that Assumptions 7.3 and 7.4 hold. Let T be fixed and known. Suppose that we run the LSE algorithm (modified for m arms) with parameters $\epsilon_n = D_L$ and $\delta_n = 8/T$ for all n. Then, the expected regret is at most*

$$L_T^{\mathrm{F}} \leq \frac{16 C_H}{D_L^2} \left(\frac{2}{1 - 1/\varphi} + \frac{8 \log_\varphi^2(m)}{T} \right) \log(T) + 8 C_H \log_\varphi(m).$$

Proof. As in the proof of Theorem 7.2, the regret of the n-th iteration is bounded by

$$L_{I_n} \leq 2\frac{C_H}{\varphi^n}\tau_n + C_H\tau_n \sum_{i=1}^{n} \delta_i.$$

Since the sampling interval $[x_n^L, x_n^H]$ is $1/\varphi^n$ at the n-th iteration, the sampling interval is at most $1/m$ wide after $\log_\varphi(m)$ and hence contains a single arm. Therefore, the LSE algorithm continues picking the same arm after $\log_\varphi(m)$ iterations.

Observe that, by Assumption 7.4 and Lemma 7.3, this final arm is the optimal arm with probability at least $1 - \sum_{i=1}^{\log_\varphi(m)} \delta_i$. Observe also that the length of the n-th iteration of the LSE algorithm is $\tau_n = \frac{16}{D_L^2}\log(\frac{8}{\delta_n})$ for all n. Hence, the total regret is

$$\begin{aligned}
L_T^{\mathrm{F}} &\leq \sum_{n=1}^{\log_\varphi(m)} \left(2\frac{C_H}{\varphi^n}\tau_n + C_H\tau_n \sum_{i=1}^{n} \delta_i\right) + TC_H \sum_{i=1}^{\log_\varphi(m)} \delta_i \\
&\leq \sum_{n=1}^{\log_\varphi(m)} C_H \left(\frac{16}{D_L^2}\log(T)\right)\left(\frac{2}{\varphi^n} + 8n/T\right) + 8C_H \log_\varphi(m) \\
&\leq C_H \frac{16}{D_L^2}\log(T)\left(\frac{2}{1-1/\varphi} + \frac{8\log_\varphi^2(m)}{T}\right) + 8C_H \log_\varphi(m),
\end{aligned}$$

where the second inequality follows by setting $\delta_n = 8/T$. ∎

7.4.5 Bandit on Unimodal Graphs

In addition to having an unimodal structure, the expected reward function may have other interesting structures. A particularly interesting one arises in the context of networks, *e.g.*, social networks, communication networks. Here, the arms of the bandit correspond to vertices in a graph and their relation is captured by the edges of the graph. It is possible to take advantage of this structure, especially when the number of edges is relatively small (as in sparse graphs), in order to find the best arm.

Bandit problems on graphs has only recently begun. In the case of stochastic rewards on tree-structured arms, a variant of the UCB algorithm has been proposed in [78]. For graphs where the rewards are nonstochastic (adversarial), but the agent fully observes the rewards, the online learning problem has recently been studied in [32, 33]. In this section, we consider graphs where vertices have stochastic rewards and with limited observation (the bandit setting). First, we make the following unimodality assumption.

Assumption 7.5 (Unimodality on graphs)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, and $\bar{r} : \mathcal{V} \to [0, 1]$ be an expected reward function that assigns a value to each vertex. We assume that $\bar{r}$ is unimodal along every path of $\mathcal{G}$, as defined in Assumption 7.3.

A heuristic solution is to combine the modification of the LSE algorithm for finitely many arms with a greedy search method, we obtain the following algorithm (Algorithm 7–3) for bandits with unimodal rewards on a graph. Although the performance of the component LSE algorithm has clear guarantees, the performance of the GLSE algorithm as a whole, however, depends critically on the characteristics of the graph that affect the number of iterations of the main loop.

7.5 Unimodal Bandit with Changes

First, we define the notion of abrupt changes in the reward distributions. We assume that there is a sequence of change-points $\nu_1, \nu_2, \ldots$ such that between consecutive change-points ν_i and ν_{i+1}, the rewards of the arms are i.i.d. random variables. Similarly to adversarial learning problems (cf. [34]), both the change-points $\nu_1, \nu_2, \ldots$ and the reward distributions are unknown. We are interested in the case where the change-point occur with small frequency.

In this section, we show that with the strong maximum assumption, there exists an efficient way to detect changes when the expected reward function

(Initialization.) $\mathcal{G}_1 = \mathcal{G}$. Fix the pivot v_1^+. The vertex set of $\mathcal{G}$ is the set of arms.
Construct a minimal spanning tree for the graph.
For $i = 1, 2, \ldots$:

1. Pick an arbitrary maximal path $\mathcal{P}$ on the spanning tree through v_i^+.
2. Run the LSE algorithm on this path until only one vertex is left: this vertex is v_i^+. Let $\mathcal{P} \setminus v_i^+$ denote the eliminated vertices.
3. Eliminate $\mathcal{P} \setminus v_i^+$ from the $\mathcal{G}_i$:

$$\mathcal{G}_{i+1} = \mathcal{G}_i \setminus \{\mathcal{P} \setminus v_i^+\}.$$

4. If $\mathcal{G}_{i+1}$ is a single vertex, then stop and output $\mathcal{G}_{i+1}$.

Algorithm 7–3: Graphical line search elimination (GLSE) algorithm

after each change-point is also unimodal. One approach is to use the windowed mean-shift detection scheme of Chapter 6. Another method is to detect when the current best arm is no longer the best: *i.e.*, by sampling the current best arm, an arm to its left and an arm to its right. We take the second approach in this chapter.

Our objective is to detect changes, but we exclude infinitesimal changes in the following assumption.

Assumption 7.6. Suppose that Assumptions 7.1 and 7.2 hold. Let ν denote a change-point. Let $\bar{r}_{\nu-1}$ and $\bar{r}_{\nu+1}$ denote the expected reward functions before and after the change-point ν. Let $x^*(\nu - 1)$ and $x^*(\nu + 1)$ denote the optimal arms for $\bar{r}_{\nu-1}$ and $\bar{r}_{\nu+1}$. We assume that there exists a constant $\beta > 0$ such that

$$|x^*(\nu - 1) - x^*(\nu + 1)| > \beta$$

for every change-point ν.

Remark 39. By Assumption 7.2, if we do not detect a change-point ν where $|x^*(\nu-1) - x^*(\nu+1)| \leq \beta$, then we incur a regret of at most $C_H\beta$ per time step, where β can be chosen appropriately small.

The following lemma gives the main idea of our method.

Lemma 7.7. *Suppose that Assumptions 7.1 and 7.2 hold. Let the three arms $z_1 < z_2 < z_3$ be given, such that $z_2 = (z_3 - z_1)/2$ and $|z_3 - z_1| = 3\beta/2$. Suppose that the true optimal arm x^* belongs to the interval $[z_2 - \beta/4, z_2 + \beta/4]$ before a change-point ν satisfying Assumption 7.6. Suppose that, after the change-point ν, we run the sampling algorithm on $\{z_1, z_2, z_3\}$ with parameters $\epsilon = C_L\beta/4$ and $\delta = \gamma$, and obtain the output z^*. Then, we have $P(z^* \neq z_2) \geq 1 - \gamma$,* i.e., *we detect the change-point ν with probability $1 - \gamma$ after $\frac{4}{\epsilon^2}\log(\frac{2n}{\gamma})$ time steps.*

Proof. First, observe that since $x^* \in [z_2 - \beta/4, z_2 + \beta/4]$ before the change-point ν, we have

$$\bar{r}_{\nu-1}(z_2) \geq \bar{r}_{\nu-1}(z_1) + C_L\beta/2$$
$$\text{and} \quad \bar{r}_{\nu-1}(z_2) \geq \bar{r}_{\nu-1}(z_3) + C_L\beta/2$$

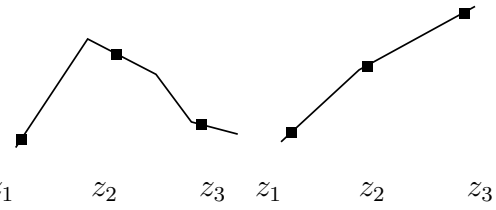
Observe also that, by Assumption 7.2, after a change-point where $|x^*(\nu-1) - x^*(\nu+1)| > \beta$, we have either

$$\bar{r}_{\nu+1}(z_1) \geq \bar{r}_{\nu+1}(z_2) + C_L\beta/2$$

or

$$\bar{r}_{\nu+1}(z_3) \geq \bar{r}_{\nu+1}(z_2) + C_L\beta/2.$$

Therefore, since $\epsilon = C_L\beta/4$, the (ϵ, γ)-PAC arm z^* is not z_2 with probability $1 - \gamma$ by Theorem 7.1. ■



(a) Expected reward function $\bar{r}_{\nu-1}$ before ν.

(b) Expected reward function $\bar{r}_{\nu+1}$ after ν.

Figure 7–3: Typical expected reward of arms z_1, z_2, z_3 before and after change-point ν.

Lemma 7.7 gives a new method of change-detection when the average reward functions before and after the change-point satisfy the assumptions: we repeatedly use the sampling algorithm on the three arms z_1, z_2, z_3 to find the appropriate PAC arm, and raise and alarm when this arm is no longer z_2. Figure 7–3 shows an example where the best arm changes from z_2 to z_3, indicating that a change-point has occurred.

7.5.1 The LCD Algorithm

Piecewise-stationary bandit problems are hard. Even with a linear number of additional queries for side observations, the delay for detecting changes is logarithmic Chapter 6. With the assumptions of this chapter, we show that piecewise-stationary bandits can be solve efficiently without side observations.

Our approach combines regret minimization and change-detection, as detailed in Algorithm 7–4. The LCD algorithm employs the stopped LSE algorithm to narrow down the sampling interval and the sampling algorithm to detect changes.

7.5.2 Regret Bound

By combining Theorem 7.4 and Lemma 7.7, we obtain the following bound on the expected regret of the LCD algorithm. Our notion of expected regret L_T^{UPS} is similar to Chapter 6.

(Initialization.) Fix u. Set $x^L = 0$ and $x^H = 1$.
(Phase 1. Stopped LSE.) Run the LSE algorithm for u iterations. After u iterations, pick three points z_1, z_2, z_3 as follows[a] : $z_1 = x_u^L - 1/\varphi^u$, $z_2 = (x_u^H - x_u^L)/2$, and $z_3 = x_u^H + 1/\varphi^u$.
(Phase 2. Change-detection.) Run the sampling algorithm on z_1, z_2, z_3 with parameters $\epsilon = \frac{C_L}{2\varphi^u}$ and $\delta = \gamma$. If the output z^* is not z_2, then a change is detected: reset $x^L = 0$ and $x^H = 1$, and go to Phase 1. Otherwise, repeat Phase 2.

[a] We allow the arms z_1 and z_3 to take values outside $[0, 1]$, but such arms are artificial and always generate zero reward.

Algorithm 7–4: The LSE with change-detection (LCD) algorithm

Theorem 7.8 (Regret of LCD)**.** *Suppose that Assumptions 7.1 and 7.2 hold. Suppose also that every change-point ν satisfies Assumption 7.6. Suppose that we run the LCD algorithm with the parameters $\epsilon_n = (2/\varphi - 1)C_L/\varphi^n$, $\delta_n = 6/T$, $\gamma = 6/T$, and the parameter u is set to $u = \frac{1}{3}\log_\varphi\left(\frac{C_L^2 T}{48\log T}\right)$ and such that $u > \log_\varphi(2/\beta)$. Then, the regret of the LCD algorithm when there are k changes up to time T is at most*

$$L_T^{\mathrm{UPS}} \le (2k+1)C_H T^{2/3}\left(\frac{48}{C_L^2}\log T\right)^{1/3} + (k+7)L_T^{\mathrm{U}} + 8C_H\frac{1}{3}\log_\varphi\left(\frac{C_L^2 T}{48\log T}\right),$$

for $T \ge 12 + 16u$.

Remark 40. This upper bound is of the order of $O(T^{2/3}\log T)$, as that of [75]. However, our notion of regret L_T^{UPS} is different. It includes, as basis for comparison, policies that may change arms at the change-points.

Remark 41. The unimodal bandit problem with change-points presents a trade-off in the width of the sampling intervals. On the one hand, a small sampling interval $[x_u^L, x_u^H]$ ensures smaller regret. On the other hand, the sampling interval $[z_1, z_3]$ must be large enough so that changes are detected quickly.

CHAPTER 8
Conclusion

In this thesis, we considered the problem of sequential decision making in non-stationary environments. We introduced models that capture various degrees of non-stationarity that better reflect real-world problems. In particular, we increase the degree of non-stationarity with the addition of non-stationary state-transition dynamics and non-stationary constraints. We limit the degree of non-stationarity by limiting the magnitude and frequency of non-stationary changes. These models allow us to provide more competitive solutions than generic regret-minimizing algorithms for worst-case non-stationary environments. In each of these models, we provide efficient learning algorithms and prove corresponding performance guarantees that depend critically on the degree of non-stationarity. We conclude this thesis by repeating the main ideas and mentioning some open problems.

For the case of Markov decision processes with non-stationary rewards and stationary transitions (Chapter 2), we considered no-regret policies within the extended model of MDPs with arbitrarily varying rewards. We showed that a simple reinforcement learning algorithm achieves diminishing average regret against any oblivious opponent. In contrast to most of the online learning literature, the obliviousness of the opponent plays a key role in characterizing the performance that the agent can achieve. Various techniques were introduced to deal with possible challenges. The Lazy FPL algorithm deals with the Markovian dynamics and an unknown time horizon T. The Q-FPL algorithm circumvents the need to calculate the exact value functions. The Exploratory FPL algorithm overcomes partially observable reward functions.

The Tracking FPL algorithm surmounts a more ambitious comparison baseline of regret composed of dynamic policies with infrequent changes. The salient features of all these algorithms can be combined to deal with combinations of the mentioned challenges.

Observe that an oblivious environment (cf. Assumption 2.2) and a completely non-oblivious one are two opposite extremes. An interesting alternative notion for the degree of non-stationarity is to model the opponent's level of obliviousness and study the effect on the achievable regret. For example, one can consider opponents that have limited memory, or that have delayed information or imperfect observations of the history (*e.g.*, opponents that only observe visits by the agent to particular states). Optimizing the convergence rate of the regret remains another open problem.

In Chapter 3, we considered the more general case where both rewards and state-transitions are subject to non-stationary changes. Although a vanishing regret is not possible in this case, we present algorithms whose regret is bounded in terms of the magnitude of the non-stationary changes. One approach is robust dynamic programming, another more efficient approach uses a simulation-based method. An open problem is quantifying the degradation of the regret of the ORDP algorithm when the uncertainty set $\mathcal{D}$ is mismatched, *i.e.*, when the ORDP algorithm is robustified with respect to an uncertainty set that is slightly different from the true uncertainty set.

In Chapter 4, we showed that in sequential convex optimization with non-stationary constraints, it is necessary to relax the notion of regret. Our formulation of sequential optimization is only a first step in considering multi-objective problems in online learning. Other formulations, such as considering the number of times that the constraints are violated, are of interest. The first open question that remains is the issue of convergence rate. We noted

that there exists an algorithm based on approachability that converges at the rate of $t^{-1/3}$, and that the usual lower bound of $t^{-1/2}$ holds. The algorithm that is based on calibration suffers from potentially even worse convergence rate, as we are not aware of of any approximate calibration algorithm with comparable convergence rates. Second, the complexity of the online learning algorithms we presented leaves much to be desired. The complexity of a policy based on approachability theory is left undetermined because we do not have a specific procedure for computing the agent's action at each stage. The per stage complexity is unknown for calibrated forecasts, but is exponential in the approximation level for approximately calibrated schemes [34]. Moreover, it is not clear whether online learning with constraints is as hard computationally as finding a calibrated forecast. Third, we only established the tightness of the lower convex hull of the Bayes envelope for the case of a one-dimensional penalty function. While this is a remarkable result because it establishes the tightness of an envelope other than the Bayes envelope, and we are not aware of any such results for similar settings. However, it is not clear whether such a result also holds for two-dimensional penalties. In particular, the proof technique of the tightness result does not seem to extend to higher dimensions.

We considered another sequential constrained optimization problem in Chapter 5, with temporal constraints that have not been studied before. We proposed a efficient expert-based algorithm and provided guarantees in the form of bounds on the regret and the frequency of constraint violation. Experiments show that this solution brings significant improvements over existing methods in real-world power management problem. An interesting generalization of this solution is to minimize the regret with respect to policies that switch between experts over time. An open question is whether constraint

violation can be limited by other techniques than arbitration, such as penalty methods.

In Chapter 6, we present efficient method to minimize the regret in bandit problem where the rewards undergo abrupt changes at unknown time instants, but remain otherwise stationary. This method queries for side observations to detect changes in the empirical mean. Its logarithmic regret is optimal up to a constant factor due to the unavoidable delay in change-detection. It would be interesting to consider different models of querying for side information: for instance, the case when queries may succeed or fail according to an i.i.d. random sequence, or the case where the agent queries two arms and then receives the reward of the better of the two.

In Chapter 7, we also consider bandits with a limited number of non-stationary changes, but with an unimodality assumption and a continuum of arms. We provide algorithm that minimizes the regret and detects change-points at the same time. One important advantage of the unimodal setting is that no side observations are necessary to detect non-stationary changes. This algorithm is also of independent interest in the case of unimodal bandits without change-points, in which case, its regret bound matches the known lower bound (up to a logarithmic factor). One interesting open problem is to generalize our results to higher dimensions. Another open problem is to modify our algorithms into so-called *anytime* bandit algorithms—which do no require prior knowledge of the time horizon T—without the doubling trick. This algorithm also has a promising application in large-scale optimizations with uncertain parameters—e.g., where the objective function is prohibitively difficult to evaluate or observed through noisy measurements.

References

[1] N. Abe and A. Nakamura. Learning to optimally schedule internet banner advertisements. In *Proceedings of ICML*, 1999.

[2] J. Abounadi, D. Bertsekas, and V. S. Borkar. Learning algorithms for Markov decision processes with average cost. *SIAM J. Control Optim.*, 40(3):681–698, 2001.

[3] R. Agrawal. The continuum-armed bandit problem. *SIAM J. Control and Optimization*, 33:1926–1951, 1995.

[4] N. Akakpo. Detecting change-points in a discrete distribution via model selection. Preprint. `http://arxiv.org/abs/0801.0970`, 2008.

[5] E. Altman. *Constrained Markov Decision Processes*. Chapman and Hall, 1999.

[6] E. Altman. Applications of dynamic games in queues. *Advances in Dynamic Games*, 7:309–342, 2005.

[7] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 1997.

[8] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3:397–422, 2002.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[10] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The non-stochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002.

[11] P. Auer, R. Ortner, and C. Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. In *Proceedings of COLT*, 2007.

[12] R. J. Aumann. Markets with a continuum of traders. *Econometrica*, 32:39–50, 1964.

[13] B. Awerbuch and R. Kleinberg. Near-optimal adaptive routing: Shortest paths and geometric generalizations. In *Proceedings of STOC*, 2004.

[14] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Online oblivious routing. In *Proceedings of SPAA*, pages 44–49, 2003.

[15] R. E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[16] D. Berry and L. Pearson. Optimal designs for two-stage clinical trials with dichotomous responses. *Statistics in Medicine*, 4:487–508, 1985.

[17] D. A. Berry, R. W. Chen, A. Zames, D. C. Heath, and L. A. Shepp. Bandit problems with infinitely many arms. *Ann. Statist.*, 1997.

[18] D. A. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Springer, 1985.

[19] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 2nd edition, 2001.

[20] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[21] Dimitri Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[22] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[23] D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.*, 1956.

[24] D. Blackwell. An analog of the minimax theorem for vector payoffs. *Pacific J. Math.*, 6(1):1–8, 1956.

[25] A. Blum, C. Burch, and A. Kalai. Finely-competitive paging. In *Proceedings of FOCS*, 1999.

[26] A. Blum, S. Chawla, and A. Kalai. Static optimality and dynamic search-optimality in lists and trees. *Algorithmica*, 36(3):249–260, 2003.

[27] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. In *Symp. on Discrete Alg.*, pages 202–204, 2003.

[28] S. G. Bobkov and P. Tetali. Modified logarithmic Sobolev inequalities in discrete settings. *Journal of Theoretical Probability*, 19(2):289–336, 2006.

[29] V. S. Borkar and S. P. Meyn. The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control Optim.*, 38(2):447–469, 2000.

[30] R. I. Brafman and M. Tennenholtz. R-max—a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2003.

[31] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in X-armed bandits. In *Advances in Neural Information Processing Systems*, 2008.

[32] N. Cesa-Bianchi, C. Gentile, and F. Vitale. Fast and optimal prediction on a labeled tree. In *Proceedings of COLT*, 2009.

[33] N. Cesa-Bianchi, C. Gentile, and F. Vitale. Learning unknown graphs. In *Proceedings of Algorithmic Learning Theory*, 2009.

[34] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[35] N. Cesa-Bianchi, G. Lugosi, and G. Stoltz. Regret minimization under partial monitoring. *Mathematics of Operations Research*, 31(3):562–580, 2006.

[36] M. Chen, I.-K. Cho, and S. P. Meyn. Reliability by design in distributed power transmission networks. *Automatica*, 42(8):1267–1281, 2006.

[37] E. W. Cope. Regret and convergence bounds for a class of continuum-armed bandit problems. *IEEE Transactions on Automatic Control*, 54(6):1243–1253, 2009.

[38] T. M. Cover and E. Ordentlich. Universal portfolios with side information. *IEEE Transactions on Information Theory*, 42(2), 1996.

[39] R. H. Crites and A. G. Barto. An actor/critic algorithm that is equivalent to Q-learning. In *Advances in Neural Information Processing Systems*, pages 401–408, 1995.

[40] M. Csörgö and L. Horváth. *Limit Theorems in Change-Point Analysis*. Wiley, 1998.

[41] D. P. de Farias and N. Megiddo. Combining expert advice in reactive environments. *Journal of the ACM*, 53(5):762–799, 2006.

[42] Gaurav Dhiman and Tajana Simunic. Dynamic power management using machine learning. In *Proceedings of the International Conference on Computer-Aided Design*, 2006.

[43] D. L. Donoho and P. B. Stark. Uncertainty principles and signal recovery. *SIAM Journal on Applied Mathematics*, 49(3):906–931, 1989.

[44] E. Even-Dar, S. Kakade, and Y. Mansour. Experts in a Markov decision process. In *Advances in Neural Information Processing Systems*, pages 401–408, 2004.

[45] E. Even-Dar, S. M. Kakade, and Y. Mansour. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

[46] E. Even-Dar, S. Mannor, and Y. Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *Proceedings of COLT*, pages 255–270, 2002.

[47] E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *J. Mach. Learn. Res.*, 7:1079–1105, 2006.

[48] V. F. Farias, C. C. Moallemi, B. Van Roy, and T. Weissman. Universal reinforcement learning. Preprint, 2007.

[49] M. Feder. Gambling using a finite state machine. *IEEE Transactions on Information Theory*, 37(5):1459–1465, 1991.

[50] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, 1996.

[51] D. P. Foster and R. V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21:40–55, 1997.

[52] Y. Freund and R. Schapire. Adaptive game playing using multiplicative weights. *GEB*, 29:79–103, 1999.

[53] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[54] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(12):79–103, 1999.

[55] D. Fudenberg and D. M. Kreps. Learning mixed equilibria. *Games and Economic Behavior*, 5(3):320–367, 1993.

[56] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[57] C. D. Fuh. Asymptotic operating characteristics of an optimal change point detection in hidden Markov models. *Ann. Statist.*, pages 2305–2339, 2004.

[58] A. Garivier and E. Moulines. On upper-confidence bound policies for non-stationary bandit problems. In *European Workshop on Reinforcement Learning*, 2008.

[59] J. C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

[60] L. Gordon and M. Pollak. A robust surveillance scheme for stochastically ordered alternatives. *The Annals of Statistics*, 23(4):1350–1375, 1995.

[61] R. Gramacy, M. Warmuth, S. Brandt, and I. Ari. Adaptive caching by refetching. In *Advances in Neural Information Processing Systems*, 2003.

[62] Robert Gramacy, Manfred Warmuth, Scott Brandt, and Ismail Ari. Adaptive caching by refetching. In *Advances in Neural Information Processing Systems*, pages 1465–1472, 2003.

[63] X. Guo and P. Shi. Limiting average criteria for nonstationary markov decision processes. *SIAM Journal on Optimization*, 11(4):1037–1053, 2000.

[64] J. Hannan. Approximation to Bayes risk in repeated play. In *Contributions to the Theory of Games*, volume 3, pages 97–139. Princeton University Press, 1957.

[65] S. Hart and A. Mas-Colell. A general class of adaptive strategies. *Journal of Economic Theory*, 98(1):26–54, 2001.

[66] C. Hartland, S. Gelly, N. Baskiotis, O. Teytaud, and M. Sebag. Multi-armed bandit, dynamic environments and meta-bandits. In *Online Trading of Exploration and Exploitation Workshop*, *NIPS*, 2006.

[67] E. Hazan and N. Megiddo. Online learning with prior information. In *Proceedings of COLT*, 2007.

[68] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. On-line portfolio selection using multiplicative updates. In *Proceedings of ICML*, 1996.

[69] David Helmbold, Darrell Long, Tracey Sconyers, and Bruce Sherrod. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications*, 5(4):285–297, 2000.

[70] M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

[71] Mark Herbster and Manfred Warmuth. Tracking the best expert. In *Proceedings of the 12th International Conference on Machine Learning*, pages 286–294, 1995.

[72] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[73] Anna Karlin, Mark Manasse, Lyle McGeoch, and Susan Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

[74] J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.

[75] R. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, pages 697–704, 2004.

[76] R. Kleinberg and T. Leighton. The value of knowing a demand curve: Bounds on regret for on-line posted-price auctions. In *Proceedings of FOCS*, 2003.

[77] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of STOC*, pages 681–690, 2008.

[78] L. Kocsis and C. Szepesvári. Discounted-UCB. 2nd PASCAL Challenges Workshop, 2006.

[79] H. J. Kushner and G. G. Yin. *Stochastic Approximation Algorithms and Applications*. Springer-Verlag, 1997.

[80] B. Kveton, J. Y. Yu, G. Theocharous, and S. Mannor. A lazy approach to online learning with constraints. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics*, pages 1–7, 2008.

[81] B. Kveton, J. Y. Yu, G. Theocharous, and S. Mannor. Online learning with expert advice and finite-horizon constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 331–336, 2008.

[82] Branislav Kveton, Prashant Gandhi, Georgios Theocharous, Shie Mannor, Barbara Rosario, and Nilesh Shah. Adaptive timeout policies for fast fine-grained power management. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, pages 1795–1800, 2007.

[83] T. L. Lai. Sequential changepoint detection in quality control and dynamical systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(4):613–658, 1995.

[84] T. L. Lai. Sequential analysis: some classical problems and new challenges. *Statistica Sinica*, 11:303–408, 2001.

[85] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6:4–22, 1985.

[86] A. Lapidoth and P. Narayan. Reliable communication under channel uncertainty. *IEEE Transactions on Information Theory*, 44(6):2148–2177, 1998.

[87] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[88] G. Lorden. Procedures for reacting to a change in distribution. *Ann. Math. Statist.*, 42:1897–1908, 1971.

[89] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: An application of large-scale online learning. In *Proceedings of the International Conference on Machine Learning*, pages 681–688, 2009.

[90] S. Mannor, J. S. Shamma, and G. Arslan. Online calibrated forecasts: Memory efficiency versus universality for learning in games. *Machine Learning*, 67(1–2):77–115, 2007.

[91] S. Mannor and N. Shimkin. The empirical Bayes envelope and regret minimization in competitive Markov decision processes. *Mathematics of Operations Research*, 28(2):327–345, 2003.

[92] S. Mannor and N. Shimkin. A geometric approach to multi-criterion reinforcement learning. *Journal of Machine Learning Research*, 5:325–360, 2004.

[93] S. Mannor and N. Shimkin. Regret minimization in repeated matrix games with variable stage duration. *Games and Economic Behavior*, 2007. In press.

[94] S. Mannor, J. Tsitsiklis, and J. Y. Yu. Online learning with sample path constraints. *Journal of Machine Learning Research*, 10(Mar):569–590, 2009.

[95] Y. J. Mei. Sequential change-point detection when unknown parameters are present in the pre-change distribution. *Ann. Statist.*, 34(1):92–122, 2006.

[96] N. Merhav and M. Feder. Universal prediction. *IEEE Transactions on Information Theory*, 44(6):2124–2147, 1998.

[97] N. Merhav, E. Ordentlich, G. Seroussi, and M. J. Weinberger. On sequential strategies for loss functions with memory. *IEEE Transactions on Information Theory*, 48(7):1947–1958, 2002.

[98] A. J. Mersereau, P. Rusmevichientong, and J. N. Tsitsiklis. A structured multiarmed bandit problem and the greedy policy. In *Proceedings IEEE Conference on Decision and Control*, 2008.

[99] J. F. Mertens, S. Sorin, and S. Zamir. Repeated games. CORE Reprint Dps 9420, 9421 and 9422, Center for Operation Research and Econometrics, Universite Catholique De Louvain, Belgium, 1994.

[100] C. Monteleoni and T. Jaakkola. Online learning of non-stationary sequences. In *Advances in Neural Information Processing Systems*, 2004.

[101] A. Nilim and L. El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[102] S. Özekici. Markov modulated Bernoulli process. *Mathematical Methods of Operations Research*, 45(3):311–324, 1997.

[103] E. S. Page. Continuous inspection scheme. *Biometrika*, 41:100–115, 1954.

[104] S. Pandey, D. Chakrabarti, and D. Agarwal. Multi-armed bandit problems with dependent arms. In *Proceedings of ICML*, 2007.

[105] S. Pandey and C. Olston. Handling advertisements of unknown quality in search advertising. In *Advances in Neural Information Processing Systems*, 2006.

[106] M. Pollak. Optimal detection of a change in distribution. *Ann. Statist.*, 13:206–227, 1985.

[107] M. Pollak. Average run lengths of an optimal method of detecting a change in distribution. *Ann. Statist.*, 15(2):749–779, 1987.

[108] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[109] T. E. S. Raghavan and J. A. Filar. Algorithms for stochastic games—a survey. *Journal Mathematical Methods of Operations Research*, 35(6):437–472, 1991.

[110] J. Renegar. Some perturbation theory for linear programming. *Math. Programming*, 65(1):73–91, 1994.

[111] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, 1984.

[112] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952.

[113] S. M. Robinson. Bounds for error in the solution set of a perturbed linear program. *Linear Algebra and its Applications*, 6:69–81, 1973.

[114] M. Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9:185–202, 1974.

[115] P. J. Schweitzer. Perturbation theory and finite Markov chains. *Journal of Applied Probability*, 5:401–413, 1968.

[116] L. Shapley. Stochastic games. *PNAS*, 39(10):1095–1100, 1953.

[117] N. Shimkin. Stochastic games with average cost constraints. In T. Basar and A. Haurie, editors, *Advances in Dynamic Games and Applications*, pages 219–230. Birkhauser, 1994.

[118] N. Shimkin and A. Shwartz. Guaranteed performance regions in Markovian systems with competing decision makers. *IEEE Transactions on Automatic Control*, 38(1):84–95, 1993.

[119] A. N. Shiryayev. On optimum methods in quickest detection problems. *Theory Probab. Appl.*, 8:22–46, 1963.

[120] X. Spinat. A necessary and sufficient condition for approachability. *Mathematics of Operations Research*, 27(1):31–44, 2002.

[121] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[122] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 1996.

[123] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[124] V. Vovk. Aggregating strategies. In *Proceedings of COLT*, pages 372–383, 1990.

[125] Y. Wang and S. Gelly. Modification of UCT with patterns in Monte-Carlo Go. In *Proceedings of ADPRL*, 2007.

[126] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[127] H. Xu and S. Mannor. The robustness-performance tradeoff in Markov decision processes. In *Advances in Neural Information Processing Systems*, pages 1537–1544, 2006.

[128] J. Y. Yu and S. Mannor. Arbitrarily modulated Markov decision processes. In *Proceedings IEEE Conference on Decision and Control*, 2009.

[129] J. Y. Yu and S. Mannor. Online learning in Markov decision processes with arbitrarily changing rewards and transitions. In *Proceedings of the International Conference on Game Theory for Networks (GameNets)*, 2009.

[130] J. Y. Yu and S. Mannor. Piecewise-stationary bandit problems with side observations. In *Proceedings of ICML*, 2009.

[131] J. Y. Yu, S. Mannor, and N. Shimkin. Markov decision processes with arbitrary reward processes. *Mathematics of Operations Research*, 34(3):737–757, 2009.

[132] F. Zhdanov, V. Vovk, B. Burford, D. Devetyarov, I. Nouretdinov, and A. Gammerman. Online prediction of ovarian cancer. In *Proceedings of the Conference on Artificial Intelligence in Medicine*, pages 375–379, 2009.

[133] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of ICML*, 2003.

[134] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.