EFFICIENT REINFORCEMENT LEARNING WITH VALUE FUNCTION GENERALIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL
ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Zheng Wen
March 2014

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Benjamin Van Roy, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Stephen Boyd

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Ramesh Johari

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Daniel O'Neill

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

Abstract

Reinforcement learning (RL) is concerned with how an agent should learn to make decisions over time while interacting with an environment. A growing body of work has produced RL algorithms with sample and computational efficiency guarantees. However, most of this work focuses on *tabula rasa* learning; i.e. algorithms aim to learn with little or no prior knowledge about the environment. Such algorithms exhibit sample complexities that grow at least linearly in the number of states, and they are of limited practical import since state spaces in most relevant contexts are enormous. There is a need for algorithms that generalize in order to learn how to make effective decisions at states beyond the scope of past experience. This dissertation focuses on the open issue of developing efficient RL algorithms that leverage value function generalization (VFG).

This dissertation consists of two parts. In the first part, we present sample complexity results for two classes of RL problems – deterministic systems with general forms of VFG and Markov decision processes (MDPs) with a finite hypothesis class. The results provide upper bounds that are independent of state and action space cardinalities and polynomial in other problem parameters. In the second part, building on insights from our sample complexity analyses, we propose randomized least-square value iteration (RLSVI), a RL algorithm for MDPs with VFG via linear hypothesis classes. The algorithm is based on a new notion of randomized value function exploration. We compare through computational studies the performance of RLSVI against least-square value iterations (LSVI) with Boltzmann exploration or epsilon-greedy exploration, which are widely used in RL with VFG. Results demonstrate that RLSVI is orders of magnitude more efficient.

To my parents

Acknowledgments

First and foremost, I would like to thank my adviser, Prof. Benjamin Van Roy. I met with Ben on my first day at Stanford, and it has been an honor to be one of his Ph.D. students. Over the past few years, he has not only taught me a lot of technical knowledge and skills in the broad areas of operations research, machine learning, and dynamic systems, but more importantly, he has also taught me how to do research in these fields. Under his guidance, I have learned how to identify and formulate research problems, how to develop rigorous solutions, and how to present the results in a comprehensible manner. This thesis work would not be possible without his most helpful guidance, and I very much appreciate every minute he has spent on my research projects. Furthermore, Ben is always willing to share his experiences in both academia and various industries, which has greatly broadened my horizons.

I am also indebted to my oral committee members Prof. Stephen Boyd, Prof. Ramesh Johari, and Prof. Daniel O'Neill. Their most insightful comments have helped me develop different perspectives on this research, from which I benefit extensively. I also would like to thank Prof. Boyd for teaching me convex optimization, and thank Prof. Johari for teaching me game theory. I very much enjoyed both classes. I also would like to thank Prof. O'Neill for guiding me on an interesting research project of building an automatic energy management system based on reinforcement learning, which is complementary to my Ph.D. dissertation.

In addition, I am deeply grateful to Prof. Louis Durlofsky and Prof. Khalid Aziz, who supported me during my first year at Stanford University and supervised me on a research project of optimizing oil production based on approximate dynamic programming. I would also like to thank Prof. Durlofsky for serving as the chair of my oral committee.

I also wish to express my gratitude to my colleagues during my internship at Technicolor research center, especially my mentor Dr. Branislav Kveton. Working with them has inspired me to formulate and tackle research problems from a more practical point of view.

I also want to thank all my course instructors at Stanford University. In the past few years, I have taken many high-quality courses at Stanford and have benefited a lot from them. I also would like to express my thanks to the staff in the Department of Electrical Engineering, the Smart Fields Consortium, and the Graduate School of Business at Stanford University, who are always very helpful.

I feel lucky and privileged to be surrounded by my fellow group members Robbie Yan, Michael Padilla, Beomsoo Park, Waraporn Tongprasit, Edward Kao, Hamid Maei, Dan Russo, and Ian Osband. I cannot overemphasize how much I enjoy and have benefited from our conversations and interactions.

Finally, I would like to thank all my friends and my family for their companionship, encouragement, and support.

The author was supported by a Stanford Graduate Fellowship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Reinforcement Learning

Reinforcement learning (RL) is concerned with how an agent (decision-maker) should learn to make decisions over time while interacting with an environment (see [9], [10], [43], [44]). In each time period, the agent needs to choose an *action* (i.e. make a decision) based on its prior knowledge about the environment and all its past observations and actions, with the objective of optimizing some form of *cumulative reward* received from the environment. In most practical RL problems, the agent's prior knowledge about the environment is informationally insufficient to derive an optimal (or near-optimal) sequence of actions, even if the agent had infinite computational power. Consequently, the agent must discover such sequence of actions by *exploring* different actions as it interacts with the environment. Furthermore, in the most interesting and challenging cases, the agent's actions may affect not only the immediate reward, but also the next *state* of the environment, and, through that, all subsequent rewards. In other words, the agent's actions can have *delayed consequences.*

Exploration and delayed consequences are the two key features of RL, which distinguish it from other types of learning, such as *supervised learning* and learning with *stochastic multi-armed bandits (MAB).* Supervised learning is an important class of learning problems, and most current literature in the field of machine learning focuses on it. Specifically, it addresses the problem of inferring a function from examples (a labeled training dataset) provided by a knowledgeable external supervisor. Since the examples are already available, supervised learning algorithms do not need to explore. However, supervised learning is usually inadequate for interactive learning problems in which it is impractical to obtain examples of desired behavior that are both correct and representative. On the other hand, stochastic MAB is a class of RL problems with a single state. In a stochastic MAB problem, the agent still needs to explore, however, since there is a single state, its actions only affect the immediate reward. That is, the agent's actions do not have delayed consequences. Since RL addresses the need for exploration and learning from delayed consequences, in many cases, RL can address elements of system behavior that are not adequately handled by other learning approaches, such as supervised learning and learning with stochastic MAB.

In many RL problems, the environment is formulated as a *Markov decision process (MDP)*, and the agent's objective is to maximize the *expected cumulative reward*. RL problems in this context are highly related to *stochastic control* and *dynamic programming (DP)* (see [8], [9]). The main differences between the classical stochastic control problem and a RL problem is that the latter does not need complete knowledge about the MDP (environment), and learns to make good decisions based on past observations. From the perspective of stochastic control and dynamic programming, RL can be viewed as data-driven (or sample-based) dynamic programming. It is worth pointing out that RL algorithms are often used to approximate solutions to large scale dynamic programs. In such contexts, there is no need for statistical learning as challenges are purely computational. In particular, dynamic programming and *approximate dynamic programming (ADP)* algorithms (see [9], [10], [36]) are able to access the underlying system model directly without making inferences through realized states and rewards. Nevertheless, RL algorithms make up popular solution techniques.

A typical RL problem is illustrated in Figure 1.1, where the environment is formulated as an MDP. In each time period, the agent chooses an action based on its prior knowledge, past observations, and current state, and observes the immediate reward and the state transition.

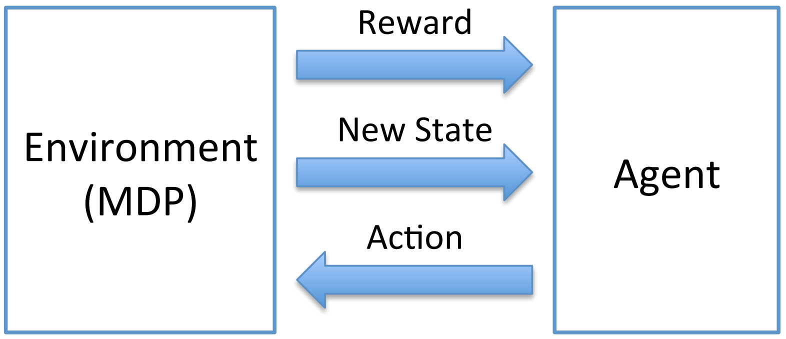

Figure 1.1: Illustration of typical RL problems

Recently, fueled by modern information technology (IT), RL is becoming increasingly important in a wide variety of applications, such as recommendation systems, smart grid, operations management (see examples below), robotics (see [15]), and financial engineering (see [33], [32]). In the remainder of this section, we discuss several potential applications of RL.

1. Recommendation Systems

There is an extensive class of Web applications that try to recommend products (e.g. movies, music, news, books), services (e.g. restaurants, financial services), or persons (e.g. friends on social networks, online dating) to its potential customers. A lot of literature is dedicated to this field (see [23], [49], [52] and references therein) and many different problem formulations have been proposed. Specifically,

- Under a supervised learning formulation, a recommendation system seeks to predict the "rating" or "preference" that user would give to an item based on a *labeled training dataset* from a knowledgeable expert.

- Under an MAB formulation, a recommendation system *explores* to learn to predict the "rating" or "preference" that user would give to an item. Notice that under this formulation, the recommendation system does not require a labeled training dataset from a knowledgeable expert.

- Under an RL formulation, a recommendation system *explores with delayed consequences* to learn to predict the "rating" or "preference" that user would give to an item. Similarly as an MAB formulation, under an RL formulation, the recommendation system also does not require a labeled training dataset from a knowledgeable expert. Furthermore, under an RL formulation, a recommendation system has the potential to learn *how to influence* customer behavior, since the recommendation system's influences on customer behavior can be modeled as state transitions in the MDP modeling the customer.

Thus, in a recommendation system context, an RL formulation has the potential to learn how to influence customer behavior, which is not addressed under a supervised learning formulation or an MAB formulation. Consequently, we might be able to build better recommendation systems based on RL.

2. Device Management in Smart Home

From the perspective of smart grid, demand response (DR) systems dynamically adjust electrical demand in response to changing electrical energy prices or other grid signals (see [11], [13], [6]). Furthermore, DR for residential and small commercial buildings is estimated to account for as much as 65% of the total energy savings potential of DR. However, due to the "decision fatigues" of the residential electricity users, an automated energy management system (EMS) is a necessary prerequisite to DR in the residential sector (see [34]). As is illustrated in Figure 1.2, an automated EMS receives requests and evaluations from the user, observes the energy prices, and reschedules (e.g. delays) user requests in response to dynamic energy prices. Rescheduling user requests can reduce the energy cost, but might also incur user dissatisfaction.

Consequently, the main tradeoff in this problem is that an automated EMS needs to balance the energy cost and the user's dissatisfaction from rescheduling. Moreover, in most cases, modeling idiosyncratic users' behavior is not cost-effective. Thus, in practice, an automated EMS needs to learn to make good rescheduling decisions while interacting with the user, and this problem is naturally formulated as an RL problem

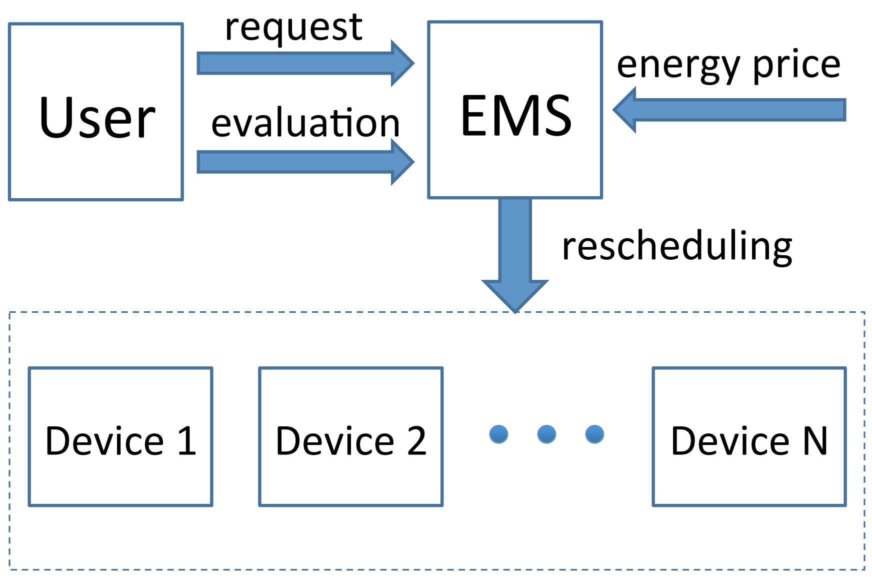

Figure 1.2: Illustration of the automated energy management system

(see [34], [53]).

3. Petroleum Reservoir Production Optimization

Recently, there has been increasing interest in applying advanced optimization techniques to optimize petroleum reservoir production (see [40], [19], [50], [51]). Unlike the previous two examples, the petroleum reservoir production optimization problem usually does not involve statistical learning; instead, in most cases, the dynamic model of the petroleum reservoir is known, and this problem can be formulated as an optimal control problem. Thus, in principle, an optimal solution of this problem can be computed via dynamic programming.

However, most petroleum reservoir production problems of practical interest are large-scale, nonlinear dynamic optimization problems. Thus, except for a few special cases, if we directly apply DP to such problems, the computational requirements become prohibitive due to the enormous or even infinite number of states (known as *curse of dimensionality* in DP and RL literature). One way to overcome the curse of dimensionality in DP is to apply ADP, and some recent literature has proposed an

ADP algorithm for this problem based on linear programming approach in DP (see [16], [17], [50], [51]).

On the other hand, petroleum reservoir simulation is a well-studied field, and reservoir simulators are available for most petroleum reservoirs (see [5]). Notice that by treating a reservoir simulator as the "environment", one can use RL algorithms to learn near-optimal controls in a petroleum reservoir production optimization problem. In some cases, some suitable RL algorithms might achieve better solutions than existing optimization techniques.

1.2 Efficient Reinforcement Learning and Generalization

Efficient reinforcement learning (RL) is concerned with how to make the best use of data and computational resources in the course of RL. Specifically, an RL algorithm is *statistically efficient* (or *sample efficient*) if its *sample complexity* – the maximum expected number of time steps with poor performance – is low; it is *computationally efficient* if its per-step computational complexity is low; and it is *efficient* if it is both statistically efficient and computationally efficient.

A growing body of work on efficient RL provides algorithms with guarantees on sample and computational efficiency [27, 12, 3, 42, 7, 22]. This literature highlights the point that efficient RL is nontrivial, and an agent needs to *plan to learn* to achieve efficient RL. Specifically, an effective exploration scheme is critical to the design of any efficient RL algorithm. In particular, popular exploration schemes such as ϵ-greedy, Boltzmann, and knowledge gradient can require learning times that grow exponentially in the number of states and/or the planning horizon. We will provide such examples in this dissertation.

The idea of "plan to learn" is illustrated in Figure 1.3. Specifically, assume that the green nodes correspond to states whose local reward and state transition model is "well-known" to the agent, while the red nodes correspond to states whose local reward and state transition model is not "well-known". Furthermore, suppose that

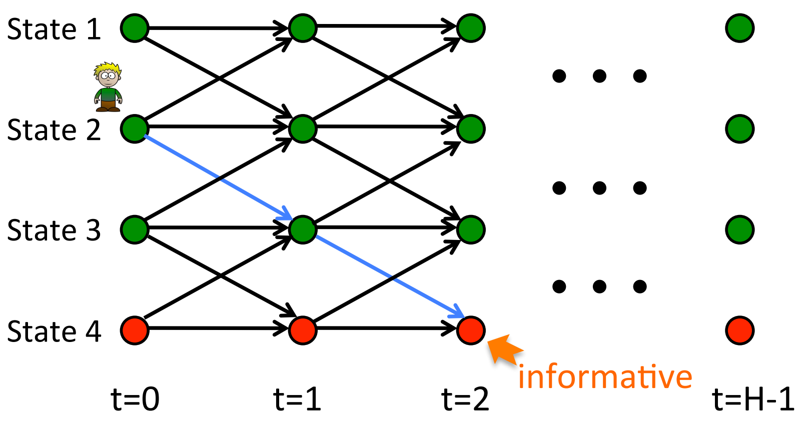

Figure 1.3: Illustration of "plan to learn"

the agent is at state 2 in time period 0. Although the reward and state transition model at the current state is well-known, however, the available information indicates that to achieve a significantly improved policy, the agent needs to visit state 4, which is an "informative state" in this case. Thus, to achieve efficient RL, the agent needs to plan to reach state 4 as quickly as possible.

The aforementioned literature focuses on *tabula rasa* learning; that is, algorithms aim to learn with little or no prior knowledge about transition probabilities and rewards. Such algorithms require learning times that grow at least linearly with the number of states. Despite the valuable insights that have been generated through their design and analysis, these algorithms are of limited practical import because state spaces in most contexts of practical interest are enormous (the curse of dimensionality). There is a need for algorithms that generalize from past experience in order to learn how to make effective decisions in reasonable time.

There has been much work on reinforcement learning algorithms that generalize (see, e.g., [10, 43, 44, 37] and references therein). Most of these algorithms do not come with statistical or computational efficiency guarantees, though there are a few noteworthy exceptions, which we now discuss. A number of results treat policy-based algorithms (see [24, 4] and references therein), in which the goal is to select high-performers among a pre-specified collection of policies as learning progresses. Though interesting results have been produced in this line of work, each entails quite restrictive assumptions or does not make strong guarantees. Another body of work focuses on model-based algorithms. An algorithm is proposed in [26] that fits a factored model to observed data and makes decisions based on the fitted model. The authors establish a sample complexity bound that is polynomial in the number of model parameters rather than the number of states, but the algorithm is computationally intractable because of the difficulty of solving factored MDPs. A recent paper [29] proposes a novel algorithm for the case where the true environment is known to belong to a finite or compact class of models, and shows that its sample complexity is polynomial in the cardinality of the model class if the model class is finite, or the ϵ-covering-number if the model class is compact. Though this result is theoretically interesting, for most model classes of interest, the ϵ-covering-number is enormous since it typically grows exponentially in the number of free parameters. Another recent paper [35] establishes a regret bound for an algorithm that applies to problems with continuous state spaces and Hölder-continuous rewards and transition kernels. Though the results represent an interesting contribution to the literature, a couple features of the regret bound weaken its practical implications. First, regret grows linearly with the Hölder constant of the transition kernel, which for most contexts of practical relevance grows exponentially in the number of state variables. Second, the dependence on time becomes arbitrarily close to linear as the dimension of the state space grows. Reinforcement learning in linear systems with quadratic cost is treated in [1]. The method proposed is shown to realize regret that grows with the square root of time. The result is interesting and the property is desirable, but to the best of our knowledge, expressions derived for regret in the analysis exhibit an exponential dependence on the number of state variables, and further, we are not aware of a computationally efficient way of implementing the proposed method. This work was extended by [21] to address linear systems with sparse structure. Here, there are efficiency guarantees that scale gracefully with the number of state variables, but only under sparsity and other technical assumptions.

The most popular approach to generalization in the applied reinforcement learning literature involves fitting parameterized value functions. Such approaches relate closely to supervised learning in that they learn functions from state to value, though a difference is that value is influenced by action and observed only through delayed feedback. One advantage over model learning approaches is that, given a fitted value function, decisions can be made without solving a potentially intractable control problem. We see this as a promising direction, though there currently is a lack of theoretical results that provide attractive bounds on learning time with value function generalization. A relevant paper along this research line is [30], which studies the efficient reinforcement learning with value function generalization in the KWIK framework (see [31]), and reduces the efficient reinforcement learning problem to the efficient KWIK online regression problem. However, the authors do not show how to solve the general KWIK online regression problem efficiently, and it is not even clear whether this is possible. Thus, though the result of [30] is interesting, it does not provide a provably efficient algorithm.

An important challenge that remains is to develop efficient RL algorithms with value function generalization (VFG) by coupling exploration and value function generalization in an effective way. In this dissertation, we aim to make progress in this direction. Specifically, in this dissertation, we distinguish between the *provably efficient* RL algorithms and the *practically efficient* RL algorithms. We say an RL algorithm is provably efficient if sample and computational efficiency guarantees that scale gracefully with the planning horizon and model complexity have been established for that algorithm; and we say an RL algorithm is practically efficient if it is sample and computationally efficient in most practical RL problems. To see the differences between provably efficient RL algorithms and practically efficient RL algorithms, notice that

- Some provably efficient RL algorithms are designed for specific RL problems, and hence might be hard to extend to more general RL problems. Specifically, for some practical RL problems, they are either not applicable, or will suffer poor performance.

- Some provably efficient RL algorithms are designed to facilitate analysis, and hence are over-complicated and/or too conservative in exploring. In many practical RL problems, the performances of such algorithms are not as good as some simple RL algorithms, though the latter class of algorithms usually does not have provable sample efficiency guarantees. Furthermore, some RL algorithms are provably sample efficient, but computationally intractable. Such RL algorithms should be viewed as constructive proofs of sample complexity guarantees, rather than practical RL algorithms.

- Some RL algorithms are sample and computationally efficient in most practical RL problems. However, for such algorithms, it is hard to establish provable sample efficiency results that scale gracefully with the planning horizon and model complexity.

1.3 Overview of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we consider the problem of RL over episodes of a finite-horizon deterministic system, and propose a provably sample efficient RL algorithm called *optimistic constraint propagation (OCP)*. OCP is designed to synthesize efficient exploration and value function generalization. Specifically, we establish that when the true value function Q^* lies within the hypothesis class $\mathcal{Q}$, OCP selects optimal actions over all but at most $\dim_{\mathrm{E}}[\mathcal{Q}]$ episodes, where $\dim_{\mathrm{E}}$ denotes the *eluder dimension*. We establish further efficiency and asymptotic performance guarantees that apply even if Q^* does not lie in $\mathcal{Q}$, for the special case where $\mathcal{Q}$ is the span of pre-specified indicator functions over disjoint sets.

In Chapter 3, we consider an RL problem in which an agent repeatedly interacts with a finite-horizon MDP, and has prior knowledge that the true value function Q^* is "close" to a known finite hypothesis class. We reformulate this problem as an MAB problem, and propose a provably efficient RL algorithm based on an existing MAB algorithm. The sample complexity of our proposed algorithm is independent of state and action space cardinalities, and polynomial in other problem parameters.

In Chapter 4, we consider RL problems in which an agent has prior knowledge that the true value function Q^* is "close" to a linear subspace (linear value function generalization). We propose an approach to exploration based on *randomized value functions* and an algorithm – *randomized least-squares value iteration (RLSVI)* – that embodies this approach. We explain why versions of least-squares value iteration that use Boltzmann or ϵ-greedy exploration can be highly inefficient and present computational results that demonstrate dramatic efficiency gains enjoyed by RLSVI. Our experiments focus on learning over episodes of a finite-horizon MDP and use a version of RLSVI designed for that task, but we also propose a version of RLSVI that addresses continual learning in an infinite-horizon discounted MDP.

Finally, we conclude and discuss future work in Chapter 5. Some parts of this dissertation have been published (see [54] and [48]).

Chapter 2

Efficient Reinforcement Learning in Episodic Deterministic Systems

2.1 Overview

In this chapter, we restrict our attention to deterministic systems that evolve over finite time horizons, and we consider episodic learning, in which an agent repeatedly interacts with the same system. As a solution to the problem, we propose *optimistic constraint propagation (OCP)*, a computationally efficient reinforcement learning algorithm designed to synthesize efficient exploration and value function generalization. We establish that when the true value function Q^* lies within the hypothesis class $\mathcal{Q}$, OCP selects optimal actions over all but at most $\dim_{\mathrm{E}}[\mathcal{Q}]$ episodes. Here, $\dim_{\mathrm{E}}$ denotes the *eluder dimension*, which quantifies complexity of the hypothesis class. A corollary of this result is that regret is bounded by a function that is constant over time and linear in the problem horizon and eluder dimension.

To put our aforementioned result in perspective, it is useful to relate it to other lines of work. Consider first the broad area of reinforcement learning algorithms that fit value functions, such as SARSA [38]. Even with the most commonly used sort of hypothesis class $\mathcal{Q}$, which is made up of linear combinations of fixed basis functions, and even when the hypothesis class contains the true value function Q^*, there are no guarantees that these algorithms will efficiently learn to make near-optimal decisions.

On the other hand, our result implies that OCP attains near-optimal performance in time that scales linearly with the number of basis functions. Now consider the more specialized context of a deterministic linear system with quadratic cost and a finite time horizon. The analysis of [1] can be leveraged to produce regret bounds that scale exponentially in the number of state variables. On the other hand, using a hypothesis space $\mathcal{Q}$ consisting of quadratic functions of state-action pairs, the results of this chapter show that OCP behaves near optimally within time that scales quadratically in the number of state and action variables.

We also establish efficiency and asymptotic performance guarantees that apply to agnostic reinforcement learning, where Q^* does not necessarily lie in $\mathcal{Q}$. In particular, we consider the case where $\mathcal{Q}$ is the span of pre-specified indicator functions over disjoint sets. Our results here add to the literature on agnostic reinforcement learning with such a hypothesis class [41, 46, 20, 47]. Prior work in this area has produced interesting algorithms and insights, as well as bounds on performance loss associated with potential limits of convergence, but no convergence or efficiency guarantees.

As we have discussed in Chapter 1, our algorithm and results also serve as contributions to approximate dynamic programming (ADP). For prior ADP algorithms that fit a linear combination of basis functions to the value function, even when the optimal value function is within the span, there are no guarantees that a near-optimal policy can be computed efficiently. In this chapter, we establish such a guarantee for OCP.

This chapter proceeds as follows. We briefly review RL in episodic deterministic systems in Section 2.2 and discuss inefficient exploration schemes in Section 2.3. In Section 2.4, we propose OCP and discuss several contexts of practical relevance and/or theoretical interest in which OCP can be applied. The sample efficiency results are established in Section 2.5 and we briefly discuss the computational complexity of OCP in Section 2.6.

The notations in this chapter are summarized in Table 2.1.

Notation	Definition
$\mathcal{M}$	A discrete-time finite-horizon deterministic system
$\mathcal{S}$	The state space of the deterministic system
$\mathcal{A}$	The action space of the deterministic system
H	The time horizon of the deterministic system
R	Reward function
$\overline{R}$	Bound on rewards
F	System function
S	The sequence of initial states of all the episodes
x	State
a	Action
j	Index of episode
t	index of period in an episode
μ	Policy
V^{μ}	State value function under policy μ
V^{*}	Optimal state value function
Q^{*}	Action-contingent optimal value function
$Q, \tilde{Q}$	Real functions of the same domain as Q^{*}
$\mathrm{Regret}(T)$	Regret over episodes experienced prior to time T
$\mathcal{Q}$	Hypothesis class
$\mathcal{C}$	A sequence of constraints on the action-contingent value function
$\varnothing$	Empty set
$\mathcal{Z}$	Set of all state-action-period triples
$\dim_{\mathrm{E}}[\mathcal{Q}]$	Eluder dimension of the hypothesis class $\mathcal{Q}$
$\mathbb{M}$	A class of finite-horizon deterministic systems
$\mathcal{Z}_t$	State-action space at period t
ρ	Distance between Q^{*} and hypothesis class $\mathcal{Q}$
$Q^{\smiley}$	optimistic action-contingent value function
$Q^{\frownie}$	pessimistic action-contingent value function

Table 2.1: Notation for Chapter 2

2.2 Reinforcement Learning in Episodic Deterministic Systems

In this chapter, we consider an episodic reinforcement learning (RL) problem in which an agent repeatedly interacts with a discrete-time finite-horizon deterministic system, and refer to each interaction as an *episode*. The system is identified by a sextuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, H is the horizon, F is a system function, R is a reward function and S is a sequence of states. If action $a \in \mathcal{A}$ is selected while the system is in state $x \in \mathcal{S}$ at period $t = 0, 1, \cdots, H-1$, a reward of $R_t(x, a)$ is realized; furthermore, if $t < H-1$, the state transitions to $F_t(x, a)$. Each episode terminates at period $H-1$, and then a new episode begins. The initial state of episode j is the jth element of S.

To represent the history of actions and observations over multiple episodes, we will often index variables by both episode and period. For example, $x_{j,t}$ and $a_{j,t}$ denote the state and action at period t of episode j, where $j = 0, 1, \cdots$ and $t = 0, 1, \cdots, H-1$. To count the total number of steps since the agent started learning, we say period t in episode j is time $jH + t$.

A (deterministic) policy $\mu = (\mu_0, \dots, \mu_{H-1})$ is a sequence of functions, each mapping $\mathcal{S}$ to $\mathcal{A}$. For each policy μ, define a value function $V_t^\mu(x) = \sum_{\tau=t}^{H-1} R_\tau(x_\tau, a_\tau)$, where $x_t = x$, $x_{\tau+1} = F_\tau(x_\tau, a_\tau)$, and $a_\tau = \mu_\tau(x_\tau)$. The optimal value function is defined by $V_t^*(x) = \sup_\mu V_t^\mu(x)$. A policy μ^* is said to be optimal if $V^{\mu^*} = V^*$. Throughout this chapter, we will restrict attention to systems $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$ that admit optimal policies. Note that this restriction incurs no loss of generality when the action space is finite.

It is also useful to define an action-contingent optimal value function:

$$Q_t^*(x, a) = \begin{cases} R_t(x, a) + V_{t+1}^*(F_t(x, a)) & \text{if } t < H-1 \\ R_{H-1}(x, a) & \text{if } t = H-1 \end{cases}$$

Then, a policy μ^* is optimal if

$$\mu_t^*(x) \in \arg\max_{a \in \mathcal{A}} Q_t^*(x,a) \quad \forall (x,t).$$

A reinforcement learning algorithm generates each action $a_{j,t}$ based on observations made up to the tth period of the jth episode, including all states, actions, and rewards observed in previous episodes and earlier in the current episode, as well as the state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, and possible prior information. In each episode, the algorithm realizes reward $R^{(j)} = \sum_{t=0}^{H-1} R_t\left(x_{j,t}, a_{j,t}\right)$. Note that $R^{(j)} \leq V_0^*(x_{j,0})$ for each jth episode. One way to quantify performance of a reinforcement learning algorithm is in terms of the number of episodes J_L for which $R^{(j)} < V_0^*(x_{j,0}) - \epsilon$, where $\epsilon \geq 0$ is a pre-specified performance loss threshold. If the reward function R is bounded, with $|R_t(x,a)| \leq \overline{R}$ for all (x,a,t), then this also implies a bound on regret over episodes experienced prior to time T, defined by

$$\text{Regret}(T) = \sum_{j=0}^{\lfloor T/H \rfloor - 1} \left(V_0^*(x_{j,0}) - R^{(j)}\right).$$

In particular, $\text{Regret}(T) \leq 2\overline{R} H J_L + \epsilon \lfloor T/H \rfloor$.

2.3 Inefficient Exploration Schemes

Before proceeding, it is worth pointing out that for the reinforcement learning problem proposed above, even in the *tabula rasa* case, a number of popular exploration schemes give rise to unsatisfactory sample complexities. Boltzmann and ϵ-greedy exploration schemes (see, e.g., [36]), for example, lead to worst-case regret exponential in H and/or $|\mathcal{S}|$. Also, the knowledge gradient exploration scheme [37] can converge to suboptimal policies, and even when the ultimate policy is optimal, the time required can grow exponentially in H and/or $|\mathcal{S}|$. Thus, even for the tabula rasa case, efficient exploration schemes are necessary for an algorithm to achieve a regret polynomial in H and $|\mathcal{S}|$.

In the remainder of this section, we provide an example that takes tabular Q-learning with Boltzmann exploration exponentially many episodes to learn an optimal policy. One can construct such examples for ϵ-greedy exploration similarly. More examples will be provided in Chapter 4.

Example 1 *Consider the deterministic system described in Figure 2.1. Specifically,*

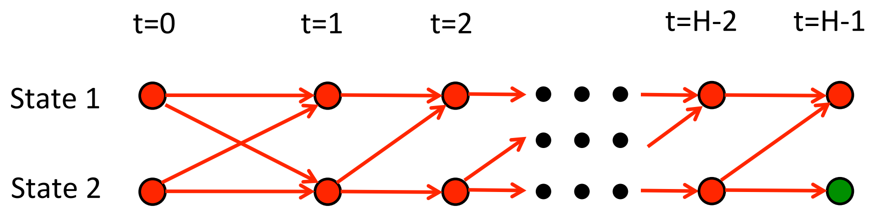

Figure 2.1: Deterministic system for which Boltzmann exploration is inefficient

in Figure 2.1, each node represents a state-time pair, and each arrow corresponds to a possible deterministic state transition. We further assume that the rewards only depend on the state-time pair, with $R_t(x) = 0$ if the node is red and $R_t(x) = 1$ if the node is green. Obviously, for this example, the optimal policy is to follow the unique path to the green node.

Now assume that we apply the Q-learning algorithm with Boltzmann exploration to this example, with initial Q-values $Q_t(x, a) = 0$, $\forall(x, a, t)$. Thus, from the Q-learning algorithm, $Q_t(x, a) = 0$, $\forall(x, a, t)$, will hold until the first visit to the green node. We also note that if $Q_t(x, a) = 0$, $\forall(x, a, t)$, then with Boltzmann exploration, the Q-learning algorithm will choose actions uniformly randomly at every state-time pair. Thus, in this case, the probability that the algorithm will visit the green node in one episode is $\frac{1}{2^{H-1}}$. Consequently, in expectation, it takes the algorithm 2^{H-1} episodes to first visit the green node.

2.4 Optimistic Constraint Propagation

Our reinforcement learning algorithm – optimistic constraint propagation (OCP) – takes as input the state space $\mathcal{S}$, the action space $\mathcal{A}$, the horizon H, and a hypothesis class $\mathcal{Q}$ of candidates for Q^*. The algorithm maintains a sequence of subsets of $\mathcal{Q}$ and a sequence of scalar "upper bounds", which summarize constraints that past experience suggests for ruling out hypotheses. Each constraint in this sequence is specified by a state $x \in \mathcal{S}$, an action $a \in \mathcal{A}$, a period $t = 0, \dots, H-1$, and an interval $[L, U] \subseteq \Re$, and takes the form $\{Q \in \mathcal{Q} : L \leq Q_t(x,a) \leq U\}$. The upper bound of the constraint is U. Given a sequence $\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_{|\mathcal{C}|})$ of such constraints and upper bounds $\mathcal{U} = (\mathcal{U}_1, \dots, \mathcal{U}_{|\mathcal{C}|})$, a set $\mathcal{Q}_\mathcal{C}$ is defined constructively by Algorithm 1. Note that if the constraints do not conflict then $\mathcal{Q}_\mathcal{C} = \mathcal{Q} \cap \mathcal{C}_1 \cap \dots \cap \mathcal{C}_{|\mathcal{C}|}$. When constraints do conflict, priority is assigned first based on upper bound, with smaller upper bound preferred, and then, in the event of ties in upper bound, based on position in the sequence, with more recent experience preferred.

Algorithm 1 Constraint Selection

```
Require: $\mathcal{Q}$, $\mathcal{C}$
  $\mathcal{Q}_\mathcal{C} \leftarrow \mathcal{Q}$, $u \leftarrow \min \mathcal{U}$
  while $u \leq \infty$ do
    for $\tau = |\mathcal{C}|$ to 1 do
      if $\mathcal{U}_\tau = u$ and $\mathcal{Q}_\mathcal{C} \cap \mathcal{C}_\tau \neq \varnothing$ then
        $\mathcal{Q}_\mathcal{C} \leftarrow \mathcal{Q}_\mathcal{C} \cap \mathcal{C}_\tau$
      end if
    end for
    if $\{u' \in \mathcal{U} : u' > u\} = \varnothing$ then
      return $\mathcal{Q}_\mathcal{C}$
    end if
    $u \leftarrow \min\{u' \in \mathcal{U} : u' > u\}$
  end while
```

OCP, presented below as Algorithm 2, at each time t computes for the current state $x_{j,t}$ and each action a the greatest state-action value $Q_t(x_{j,t}, a)$ among functions in $\mathcal{Q}_\mathcal{C}$ and selects an action that attains the maximum. In other words, an action is chosen based on the most optimistic feasible outcome subject to constraints. The subsequent reward and state transition give rise to a new constraint that is used to update $\mathcal{C}$. Note that the update of $\mathcal{C}$ is postponed until one episode is completed.

Note that if $Q^* \in \mathcal{Q}$ then each constraint appended to $\mathcal{C}$ does not rule out Q^*, and therefore, the sequence of sets $\mathcal{Q}_\mathcal{C}$ generated as the algorithm progresses is decreasing

Algorithm 2 Optimistic Constraint Propagation

```
Require: $\mathcal{S}$, $\mathcal{A}$, $H$, $\mathcal{Q}$
  Initialize $\mathcal{C} \leftarrow \varnothing$
  for episode $j = 0, 1, \cdots$ do
    Set $\mathcal{C}' \leftarrow \mathcal{C}$
    for period $t = 0, 1, \cdots, H-1$ do
      Apply $a_{j,t} \in \arg\max_{a\in\mathcal{A}} \sup_{Q\in\mathcal{Q}_{\mathcal{C}}} Q_t(x_{j,t}, a)$
      if $t < H-1$ then
        $U_{j,t} \leftarrow \sup_{Q\in\mathcal{Q}_{\mathcal{C}}} \left(R_t(x_{j,t}, a_{j,t}) + \sup_{a\in\mathcal{A}} Q_{t+1}\left(x_{j,t+1}, a\right)\right)$
        $L_{j,t} \leftarrow \inf_{Q\in\mathcal{Q}_{\mathcal{C}}} \left(R_t(x_{j,t}, a_{j,t}) + \sup_{a\in\mathcal{A}} Q_{t+1}\left(x_{j,t+1}, a\right)\right)$
      else
        $U_{j,t} \leftarrow R_t(x_{j,t}, a_{j,t})$, $L_{j,t} \leftarrow R_t(x_{j,t}, a_{j,t})$
      end if
      $\mathcal{C}' \leftarrow \mathcal{C}' \frown \{Q \in \mathcal{Q} : L_{j,t} \leq Q_t(x_{j,t}, a_{j,t}) \leq U_{j,t}\}$
    end for
    Update $\mathcal{C} \leftarrow \mathcal{C}'$
  end for
```

and contains Q^* in its intersection. In the agnostic case, where Q^* may not lie in $\mathcal{Q}$, new constraints can be inconsistent with previous constraints, in which case selected previous constraints are relaxed as determined by Algorithm 1.

Let us briefly discuss several contexts of practical relevance and/or theoretical interest in which OCP can be applied.

- **Finite state/action tabula rasa case.** With finite state and action spaces, Q^* can be represented as a vector, and without special prior knowledge, it is natural to let $\mathcal{Q} = \Re^{|\mathcal{S}|\cdot|\mathcal{A}|\cdot H}$.

- **Polytopic prior constraints.** Consider the aforementioned example, but suppose that we have prior knowledge that Q^* lies in a particular polytope. Then we can let $\mathcal{Q}$ be that polytope and again apply OCP.

- **Linear systems with quadratic cost (LQ).** In this classical control model, if $\mathcal{S} = \Re^n$, $\mathcal{A} = \Re^m$, and R is a positive semidefinite quadratic, then for each t, Q_t^* is known to be a positive semidefinite quadratic, and it is natural to let $\mathcal{Q} = \mathcal{Q}_0^H$ with $\mathcal{Q}_0$ denoting the set of positive semidefinite quadratics.

- **Finite hypothesis class.** Consider a context when we have prior knowledge that Q^* can be well approximated by some element in a finite hypothesis class. Then we can let $\mathcal{Q}$ be that finite hypothesis class and apply OCP. This scenario is of particular interest from the perspective of learning theory. Note that this context entails agnostic learning, which is accommodated by OCP.

- **Linear combination of features.** It is often effective to hand-select a set of features $\phi_1, \ldots, \phi_K$, each mapping $\mathcal{S} \times \mathcal{A}$ to $\Re$, and, then for each t, aiming to compute weights $\theta^{(t)} \in \Re^K$ so that $\sum_k \theta_k^{(t)} \phi_k$ approximates Q_t^* without knowing for sure that Q_t^* lies in the span of the features. To apply OCP here, we would let $\mathcal{Q} = \mathcal{Q}_0^H$ with $\mathcal{Q}_0 = \mathrm{span}(\phi_1, \ldots, \phi_K)$. Note that this context also entails agnostic learning.

- **Sigmoid.** If it is known that rewards are only received upon transitioning to the terminal state and take values between 0 and 1, it might be appropriate to use a variation of the aforementioned feature based model that applies a sigmoidal function to the linear combination. In particular, we could have $\mathcal{Q} = \mathcal{Q}_0^H$ with $\mathcal{Q}_0 = \left\{ \psi\left(\sum_k \theta_k \phi_k(\cdot)\right) : \theta \in \Re^K \right\}$, where $\psi(z) = e^z/(1+e^z)$.

- **Sparse linear combination of features.** Another case of potential interest is where Q^* can be encoded by a sparse linear combination of a large number of features $\phi_0, \cdots, \phi_K$. In particular, suppose that $\Phi = [\phi_0, \cdots, \phi_K] \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, and $\mathcal{Q} = \mathcal{Q}_0^H$ with

$$\mathcal{Q}_0 = \left\{ \Phi\theta : \ \theta \in \Re^K, \|\theta\|_0 \leq K_0 \right\},$$

  where $\|\theta\|_0$ is the L_0-“norm” of θ and $K_0 \ll K$.

It is worth mentioning that OCP, as we have defined it, assumes that an action a maximizing $\sup_{Q \in \mathcal{Q}_C} Q_t(x_{j,t}, a)$ exists in each iteration. Note that this assumption always holds if the action space $\mathcal{A}$ is finite, and it is not difficult to modify the algorithm so that it addresses cases where this is not true. But we have not presented the more general form of OCP in order to avoid complicating this chapter.

2.5 Sample Efficiency of Optimistic Constraint Propagation

We now establish results concerning the sample efficiency of OCP. Our results bound the time it takes OCP to learn, and this must depend on the complexity of the hypothesis class. As such, we begin by defining the eluder dimension, as introduced in [39], which is the notion of complexity we will use.

2.5.1 Eluder Dimension

Let $\mathcal{Z} = \{(x, a, t) : x \in \mathcal{S}, a \in \mathcal{A}, t = 0, \dots, H-1\}$ be the set of all state-action-period triples, and let $\mathcal{Q}$ denote a nonempty set of functions mapping $\mathcal{Z}$ to $\Re$. For all $(x, a, t) \in \mathcal{Z}$ and $\tilde{\mathcal{Z}} \subseteq \mathcal{Z}$, (x, a, t) is said to be *dependent* on $\tilde{\mathcal{Z}}$ with respect to $\mathcal{Q}$ if any pair of functions $Q, \tilde{Q} \in \mathcal{Q}$ that are equal on $\tilde{\mathcal{Z}}$ are equal at (x, a, t). Further, (x, a, t) is said to be *independent* of $\tilde{\mathcal{Z}}$ with respect to $\mathcal{Q}$ if (x, a, t) is not dependent on $\tilde{\mathcal{Z}}$ with respect to $\mathcal{Q}$.

The *eluder dimension* $\dim_{\mathrm{E}}[\mathcal{Q}]$ of $\mathcal{Q}$ is the length of the longest sequence of elements in $\mathcal{Z}$ such that every element is independent of its predecessors. Note that $\dim_{\mathrm{E}}[\mathcal{Q}]$ can be zero or infinity, and it is straightforward to show that if $\mathcal{Q}_1 \subseteq \mathcal{Q}_2$ then $\dim_{\mathrm{E}}[\mathcal{Q}_1] \leq \dim_{\mathrm{E}}[\mathcal{Q}_2]$. Based on results of [39], we can characterize the eluder dimensions of various hypothesis classes presented in the previous section.

- **Finite state/action tabula rasa case.** If $\mathcal{Q} = \Re^{|\mathcal{S}| \cdot |\mathcal{A}| \cdot H}$, then

$$\dim_{\mathrm{E}}[\mathcal{Q}] = |\mathcal{S}| \cdot |\mathcal{A}| \cdot H.$$

- **Polytopic prior constraints.** If $\mathcal{Q}$ is a polytope of dimension d in $\Re^{|\mathcal{S}| \cdot |\mathcal{A}| \cdot H}$, then $\dim_{\mathrm{E}}[\mathcal{Q}] = d$.

- **Linear systems with quadratic cost (LQ).** If $\mathcal{Q}_0$ is the set of positive semidefinite quadratics with domain $\Re^{m+n}$ and $\mathcal{Q} = \mathcal{Q}_0^H$, then

$$\dim_{\mathrm{E}}[\mathcal{Q}] = (m + n + 1)(m + n)H/2.$$

- **Finite hypothesis space.** If $|\mathcal{Q}| < \infty$, then $\dim_{\mathrm{E}}[\mathcal{Q}] \leq |\mathcal{Q}| - 1$.
- **Linear combination of features.** If $\mathcal{Q} = \mathcal{Q}_0^H$ with $\mathcal{Q}_0 = \mathrm{span}(\phi_1, \ldots, \phi_K)$, then $\dim_{\mathrm{E}}[\mathcal{Q}] \leq KH$.
- **Sigmoid.** If $\mathcal{Q} = \mathcal{Q}_0^H$ with $\mathcal{Q}_0 = \left\{\psi\left(\sum_k \theta_k \phi_k(\cdot)\right) : \theta \in \Re^K\right\}$, then
$$\dim_{\mathrm{E}}[\mathcal{Q}] \leq KH.$$
- **Sparse linear combination of features.** If $\mathcal{Q} = \mathcal{Q}_0^H$ with
$$\mathcal{Q}_0 = \left\{\Phi\theta : \theta \in \Re^K, \|\theta\|_0 \leq K_0\right\}$$
and $2K_0 \leq \min\{|\mathcal{S}||\mathcal{A}|, K\}$, and any $2K_0 \times 2K_0$ submatrix of Φ has full rank, then $\dim_{\mathrm{E}}[\mathcal{Q}] \leq 2K_0 H$. We will establish this eluder dimension bound in the appendix.

2.5.2 Coherent Learning

We now present results that apply when OCP is presented with a coherent hypothesis class; that is, where $Q^* \in \mathcal{Q}$. Our first result establishes that OCP can deliver less than optimal performance in no more than $\dim_{\mathrm{E}}[\mathcal{Q}]$ episodes.

Theorem 1 *For any system $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$, if OCP is applied with $Q^* \in \mathcal{Q}$, then*

$$|\{j : R^{(j)} < V_0^*(x_{j,0})\}| \leq \dim_{\mathrm{E}}[\mathcal{Q}]. \tag{2.1}$$

This theorem follows from an "exploration-exploitation lemma" (Lemma 2), which asserts that in each episode, OCP either delivers optimal reward (exploits) or introduces a constraint that reduces the eluder dimension of the hypothesis class by one (explores). Consequently, OCP will experience sub-optimal performance in at most

$\dim_{\mathrm{E}}[\mathcal{Q}]$ episodes. We outline the proof of Theorem 1 at the end of this subsection and the detailed analysis is provided in the appendix. An immediate corollary bounds regret.

Corollary 1 *For any $\overline{R}$, any system $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$ with*

$$\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R},$$

and any T, if OCP is applied with $Q^* \in \mathcal{Q}$, then $\mathrm{Regret}(T) \leq 2\overline{R}H\dim_{\mathrm{E}}[\mathcal{Q}]$.

Note the regret bound in Corollary 1 does not depend on time T, thus, it is an $O(1)$ bound. Furthermore, this regret bound is linear in $\overline{R}$, H and $\dim_{\mathrm{E}}[\mathcal{Q}]$, and does not directly depend on $|\mathcal{S}|$ or $|\mathcal{A}|$. The following result demonstrates that the bounds of the above theorem and corollary are sharp.

Theorem 2 *For any $\overline{R} \geq 0$, any $K, H' = 1, 2, \cdots$ and any reinforcement learning algorithm $\tilde{\mu}$ that takes as input a state space, an action space, a horizon and a coherent hypothesis class, there exist a system $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$ and a hypothesis class $\mathcal{Q}$ satisfying (1) $\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R}$, (2) $H = H'$, (3) $\dim_{\mathrm{E}}[\mathcal{Q}] = K$ and (4) $Q^* \in \mathcal{Q}$ such that if we apply $\tilde{\mu}$ to $\mathcal{M}$ with input $(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$, then $|\{j : R^{(j)} < V_0^*(x_{j,0})\}| \geq$ $\dim_{\mathrm{E}}[\mathcal{Q}]$ and $\sup_T \mathrm{Regret}(T) \geq 2\overline{R}H\dim_{\mathrm{E}}[\mathcal{Q}]$.*

A constructive proof of these lower bounds is provided at the end of this subsection. Following our discussion in previous sections, we discuss several interesting contexts in which the agent knows a coherent hypothesis class $\mathcal{Q}$ with finite eluder dimension.

- **Finite state/action tabula rasa case.** If we apply OCP in this case, then it will deliver sub-optimal performance in at most $|\mathcal{S}| \cdot |\mathcal{A}| \cdot H$ episodes. Furthermore, if $\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R}$, then for any T, $\mathrm{Regret}(T) \leq 2\overline{R}|\mathcal{S}||\mathcal{A}|H^2$.

- **Polytopic prior constraints.** If we apply OCP in this case, then it will deliver sub-optimal performance in at most d episodes. Furthermore, if

$$\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R},$$

then for any T, $\text{Regret}(T) \leq 2\overline{R}Hd$.

- **Linear systems with quadratic cost (LQ).** If we apply OCP in this case, then it will deliver sub-optimal performance in at most $(m+n+1)(m+n)H/2$ episodes.

- **Finite hypothesis class case.** Assume that the agent has prior knowledge that $Q^* \in \mathcal{Q}$, where $\mathcal{Q}$ is a finite hypothesis class. If we apply OCP in this case, then it will deliver sub-optimal performance in at most $|\mathcal{Q}| - 1$ episodes. Furthermore, if $\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R}$, then for any T, $\text{Regret}(T) \leq 2\overline{R}H\left[|\mathcal{Q}| - 1\right]$.

- **Sparse linear combination case.** Assume that the agent has prior knowledge that $Q^* \in \mathcal{Q}$, where $\mathcal{Q} = \left\{\Phi\theta :\, \theta \in \Re^K,\, \|\theta\|_0 \leq K_0\right\}^H$ and $2K_0 \leq \min\{|\mathcal{S}||\mathcal{A}|, K\}$, and any $2K_0 \times 2K_0$ submatrix of Φ has full rank. If we apply OCP in this case, then it will deliver sub-optimal performance in at most $2K_0H$ episodes. Furthermore, if $\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R}$, then for any T, $\text{Regret}(T) \leq 4\overline{R}K_0H^2$.

Sketch of Proof for Theorem 1

We start by defining some useful notations. Specifically, we use $\mathcal{C}_j$ to denote the $\mathcal{C}$ in episode j to distinguish $\mathcal{C}$'s in different episodes, and use z as a shorthand notation for a state-action-time triple (x,a,t). As we have discussed above, since $Q^* \in \mathcal{Q}$, thus each constraint appended to $\mathcal{C}$ does not rule out Q^*, and thus we have $Q^* \in \mathcal{Q}_{\mathcal{C}_j}$ for any $j = 0, 1, \cdots$.

For any episode $j = 0, 1, \cdots$, we define $\mathcal{Z}_j$ and t_j^* by Algorithm 3. Note that by definition, $\forall j = 0, 1, \cdots$,

- $\mathcal{Z}_j$ is a sequence (ordered set) of elements in $\mathcal{Z}$. Furthermore, each element in $\mathcal{Z}_j$ is independent of its predecessors.

- If $t_j^* \neq \text{NULL}$, then it is the last period in episode j s.t. $(x_{j,t}, a_{j,t}, t)$ is independent of $\mathcal{Z}_j$ with respect to $\mathcal{Q}$.

Based on the notions of $\mathcal{Z}_j$ and t_j^*, we have the following technical lemma:

Algorithm 3 Definition of $\mathcal{Z}_j$ and t_j^*

```
Initialize $\mathcal{Z}_0 \leftarrow \varnothing$
for $j = 0, 1, \cdots$ do
  Set $t_j^* \leftarrow$ NULL
  if $\exists t = 0, 1, \cdots, H-1$ s.t. $(x_{j,t}, a_{j,t}, t)$ is independent of $\mathcal{Z}_j$ with respect to $\mathcal{Q}$ then
    Set
      $t_j^* \leftarrow$ last period $t$ in episode $j$ s.t. $(x_{j,t}, a_{j,t}, t)$ is independent of $\mathcal{Z}_j$ with respect to $\mathcal{Q}$
    and $\mathcal{Z}_{j+1} \leftarrow \left[\mathcal{Z}_j, (x_{j,t_j^*}, a_{j,t_j^*}, t_j^*)\right]$
  else
    Set $\mathcal{Z}_{j+1} \leftarrow \mathcal{Z}_j$
  end if
end for
```

Lemma 1 $\forall j = 0, 1, \cdots$ *and* $\forall t = 0, 1, \cdots, H-1$, *we have*

(a) $\forall z \in \mathcal{Z}_j$ *and* $\forall Q \in \mathcal{Q}_{\mathcal{C}_j}$, *we have* $Q(z) = Q^*(z)$.

(b) *If* $(x_{j,t}, a_{j,t}, t)$ *is dependent on* $\mathcal{Z}_j$ *with respect to* $\mathcal{Q}$, *then (1)* $a_{j,t}$ *is optimal and (2)* $Q_t(x_{j,t}, a_{j,t}) = Q_t^*(x_{j,t}, a_{j,t}) = V_t^*(x_{j,t})$, $\forall Q \in \mathcal{Q}_{\mathcal{C}_j}$.

Please refer to the appendix for the proof of Lemma 1. Based on Lemma 1, we have the following exploration/exploitation lemma, which states that in each episode j, OCP algorithm either achieves the optimal reward (exploits), or updates $\mathcal{Q}_{\mathcal{C}_{j+1}}$ based on the Q-value at an independent state-action-time triple (explores).

Lemma 2 *For any* $j = 0, 1, \cdots$, *if* $t_j^* \neq$ NULL, *then* $(x_{j,t_j^*}, a_{j,t_j^*}, t_j^*)$ *is independent of* $\mathcal{Z}_j$, $|\mathcal{Z}_{j+1}| = |\mathcal{Z}_j| + 1$ *and* $Q_{t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) = Q_{t_j^*}^*(x_{j,t_j^*}, a_{j,t_j^*})$ $\forall Q \in \mathcal{Q}_{\mathcal{C}_{j+1}}$ (Exploration). Otherwise, if $t_j^* =$ NULL, then $R^{(j)} = V_0^*(x_{j,0})$ (Exploitation).

Theorem 1 follows from Lemma 2. Please refer to the appendix for the detailed proofs for Lemma 2 and Theorem 1.

Constructive Proof for Theorem 2

We start by defining some useful terminologies and notations. First, for any state space $\mathcal{S}$, any time horizon $H = 1, 2, \cdots$, any action space $\mathcal{A}$, and any hypothesis class $\mathcal{Q}$, we use $\mathbb{M}(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$ to denote the set of all finite-horizon deterministic system $\mathcal{M}$'s with state space $\mathcal{S}$, action space $\mathcal{A}$, horizon H and $Q^* \in \mathcal{Q}$. Notice that for any reinforcement learning algorithm that takes $\mathcal{S}$, $\mathcal{A}$, H, $\mathcal{Q}$ as input, and knows that $\mathcal{Q}$ is a coherent hypothesis class, $\mathbb{M}(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$ is the set of all finite-horizon deterministic systems that are consistent with the algorithm's prior information.

We prove a result that is stronger than Theorem 2 by considering a scenario in which an adversary adaptively chooses a deterministic system $\mathcal{M} \in \mathbb{M}(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$. Specifically, we assume that

- At the beginning of each episode j, the adversary adaptively chooses the initial state $x_{j,0}$.
- At period t in episode j, the agent first chooses an action $a_{j,t} \in \mathcal{A}$ based on some RL algorithm[1], and then the adversary adaptively chooses a set of state-action-time triples $\mathcal{Z}_{j,t} \subseteq \mathcal{Z}$ and specifies the rewards and state transitions on $\mathcal{Z}_{j,t}$, subject to the constraints that (1) $(x_{j,t}, a_{j,t}, t) \in \mathcal{Z}_{j,t}$ and (2) these adaptively specified rewards and state transitions must be consistent with the agent's prior knowledge and past observations.

We assume that the adversary's objective is to maximize the number of episodes in which the agent achieves sub-optimal rewards. Then we have the following lemma:

Lemma 3 $\forall H, K = 1, 2, \cdots$ *and* $\forall \overline{R} \geq 0$*, there exist a state space* $\mathcal{S}$*, an action space* $\mathcal{A}$ *and a hypothesis class* $\mathcal{Q}$ *with* $\dim_{\mathrm{E}}[\mathcal{Q}] = K$ *such that no matter how the agent adaptively chooses actions, the adversary can adaptively choose an* $\mathcal{M} \in \mathbb{M}(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$ *with* $\sup_{(x,a,t)} |R_t(x, a)| \leq \overline{R}$ *such that the agent will achieve sub-optimal rewards in at least* K *episodes, and* $\sup_T \mathrm{Regret}(T) \geq 2\overline{R}HK$*.*

[1]In general, the RL algorithm can choose actions randomly. If so, all the results in this section hold on the realized sample path.

Since the fact that an adversary can adaptively choose a "bad" deterministic system simply implies that such system exists, thus, Theorem 2 follows directly from Lemma 3.

Proof for Lemma 3: We provide a constructive proof for Lemma 3. Specifically, $\forall H, K = 1, 2, \cdots$ and $\forall \overline{R} \geq 0$, we construct the state space as $\mathcal{S} = \{1, 2, \cdots, 2K\}$, and the action space as $\mathcal{A} = \{1, 2\}$. Recall that

$$\mathcal{Z} = \{(x, a, t) : x \in \mathcal{S}, t = 0, 1, \cdots, H-1, \text{ and } a \in \mathcal{A}\},$$

thus, for $\mathcal{S}$ and $\mathcal{A}$ constructed above, we have $|\mathcal{Z}| = 4KH$. Hence, Q^*, the optimal Q-function, can be represented as a vector in $\Re^{4KH}$.

Before specifying the hypothesis class $\mathcal{Q}$, we first define a matrix $\Phi \in \Re^{4KH \times K}$ as follows. $\forall (x, a, t) \in \mathcal{Z}$, let $\Phi(x, a, t) \in \Re^K$ denote the row of Φ corresponding to the state-action-time triple (x, a, t), we construct $\Phi(x, a, t)$ as:

$$\Phi(x,a,t) = \begin{cases} (H-t)\mathbf{e}_k & \text{if } x = 2k-1 \text{ for some } k = 1, \cdots, K,\ a = 1, 2 \text{ and } t > 0 \\ -(H-t)\mathbf{e}_k & \text{if } x = 2k \text{ for some } k = 1, \cdots, K,\ a = 1, 2 \text{ and } t > 0 \\ H\mathbf{e}_k & \text{if } x = 2k-1 \text{ or } 2k \text{ for some } k = 1, \cdots, K,\ a = 1 \text{ and } t = 0 \\ -H\mathbf{e}_k & \text{if } x = 2k-1 \text{ or } 2k \text{ for some } k = 1, \cdots, K,\ a = 2 \text{ and } t = 0 \end{cases} \tag{2.2}$$

where $\mathbf{e}_k \in \Re^K$ is a (row) indicator vector with a one at index k and zeros everywhere else. Obviously, $\text{rank}(\Phi) = K$. We choose $\mathcal{Q} = \text{span}[\Phi]$, thus $\dim_{\text{E}}[\mathcal{Q}] = \dim(\text{span}[\Phi]) = \text{rank}(\Phi) = K$.

Now we describe how the adversary adaptively chooses a finite-horizon deterministic system $\mathcal{M} \in \mathbb{M}(\mathcal{S}, \mathcal{A}, H, \mathcal{Q})$:

- For any $j = 0, 1, \cdots$, at the beginning of episode j, the adversary chooses the initial state in that episode as $x_{j,0} = (j \bmod K) \times 2 + 1$. That is, $x_{0,0} = x_{K,0} = x_{2K,0} = \cdots = 1$, $x_{1,0} = x_{K+1,0} = x_{2K+1,0} = \cdots = 3$, etc.

- Before interacting with the agent, the adversary chooses the following system

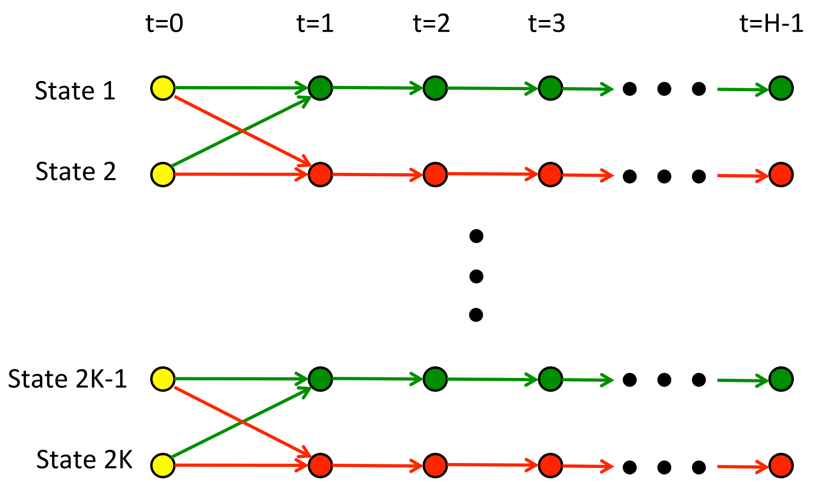

Figure 2.2: Deterministic system constructed by the adversary

function F^2:

$$F_t(x,a) = \begin{cases} 2k-1 & \text{if } t=0,\ x=2k-1 \text{ or } 2k \text{ for some } k=1,\cdots,K, \text{ and } a=1 \\ 2k & \text{if } t=0,\ x=2k-1 \text{ or } 2k \text{ for some } k=1,\cdots,K, \text{ and } a=2 \\ x & \text{if } t=1,\cdots,H-2 \text{ and } a=1,2 \end{cases}$$

The state transition is illustrated in Figure 2.2.

- In episode $j = 0, 1, \cdots, K-1$, the adversary adaptively chooses the reward function R as follows. If the agent takes action 1 in period 0 in episode j at initial state $x_{j,0} = 2j+1$, then the adversary set

$$R_0(2j+1,1) = R_0(2j+2,1) = R_t(2j+1,1) = R_t(2j+1,2) = -\overline{R}$$

and

$$R_0(2j+1,2) = R_0(2j+2,2) = R_t(2j+2,1) = R_t(2j+2,2) = \overline{R},$$

$\forall t = 1, 2, \cdots, H-1$. Otherwise (i.e. if the agent takes action 2 in period 0 in

[2]More precisely, in this constructive proof, the adversary does not need to adaptively choose the system function F. He can choose F beforehand.

episode j), then the adversary set

$$R_0(2j+1,1) = R_0(2j+2,1) = R_t(2j+1,1) = R_t(2j+1,2) = \overline{R}$$

and

$$R_0(2j+1,2) = R_0(2j+2,2) = R_t(2j+2,1) = R_t(2j+2,2) = -\overline{R}.$$

Notice that the adversary completes the construction of the deterministic system $\mathcal{M}$ at the end of episode $K-1$.

Note that for the constructed deterministic system $\mathcal{M}$, we have $Q^* \in \mathcal{Q}$. Specifically, it is straight forward to see that $Q^* = \Phi\theta^*$, where $\theta^* \in \Re^K$, and θ^*_k, the kth element of θ, is defined as $\theta^*_k = -\overline{R}$ if $a_{k-1,0} = 1$ and $\theta^*_k = \overline{R}$ if $a_{k-1,0} = 2$, for any $k = 1, 2, \cdots, K$. Thus, the constructed deterministic system $\mathcal{M} \in \mathbb{M}\left(\mathcal{S}, \mathcal{A}, H, \mathcal{Q}\right)$.

Finally, we show that the constructed deterministic system $\mathcal{M}$ satisfies Lemma 3. Obviously, we have $|R_t(x,a)| \leq \overline{R}$, $\forall (x,a,t) \in \mathcal{Z}$. Furthermore, note that the agent achieves sub-optimal rewards in the first K episodes, thus, he will achieve sub-optimal rewards in at least K episodes. In addition, the cumulative regret in the first K episodes is $2KH\overline{R}$, thus, $\sup_T \mathrm{Regret}(T) \geq 2KH\overline{R}$. **q.e.d.**

2.5.3 Agnostic Learning

As we have discussed in Section 2.4, OCP can also be applied in agnostic learning cases, where Q^* may not lie in $\mathcal{Q}$. For such cases, the performance of OCP should depend on not only the complexity of $\mathcal{Q}$, but also the distance between $\mathcal{Q}$ and Q^*. We now present results when OCP is applied in a special agnostic learning case, where $\mathcal{Q}$ is the span of pre-specified indicator functions over disjoint subsets. We henceforth refer to this case as the state aggregation case.

Specifically, we assume that for any $t = 0, 1, \cdots, H-1$, the state-action space at period t, $\mathcal{Z}_t = \{(x,a,t) : x \in \mathcal{S}, a \in \mathcal{A}\}$, can be partitioned into K_t disjoint subsets $\mathcal{Z}_{t,1}, \mathcal{Z}_{t,2}, \cdots, \mathcal{Z}_{t,K_t}$, and use $\phi_{t,k}$ to denote the indicator function for partition $\mathcal{Z}_{t,k}$

(i.e. $\phi_{t,k}(x,a,t) = 1$ if $(x,a,t) \in \mathcal{Z}_{t,k}$, and $\phi_{t,k}(x,a,t) = 0$ otherwise). We define $K = \sum_{t=0}^{H-1} K_t$, and $\mathcal{Q}$ as

$$\mathcal{Q} = \operatorname{span}\left\{\phi_{0,1}, \phi_{0,2}, \cdots, \phi_{0,K_0}, \phi_{1,1}, \cdots, \phi_{H-1,K_{H-1}}\right\}. \tag{2.3}$$

Note that $\dim_{\mathrm{E}}[\mathcal{Q}] = K$. We define the distance between Q^* and the hypothesis class $\mathcal{Q}$ as

$$\rho = \min_{Q \in \mathcal{Q}} \|Q - Q^*\|_\infty = \min_{Q \in \mathcal{Q}} \sup_{(x,a,t)} |Q_t(x,a) - Q_t^*(x,a)|. \tag{2.4}$$

The following result establishes that with $\mathcal{Q}$ and ρ defined above, the performance loss of OCP is larger than $2\rho H(H+1)$ in at most K episodes.

Theorem 3 *For any system $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$, if OCP is applied with $\mathcal{Q}$ defined in Eqn(2.3), then*

$$|\{j : R^{(j)} < V_0^*(x_{j,0}) - 2\rho H(H+1)\}| \leq K,$$

where K is the number of partitions and ρ is defined in Eqn(2.4).

Similar to Theorem 1, this theorem also follows from an "exploration-exploitation lemma" (Lemma 6), which asserts that in each episode, OCP either delivers near-optimal reward (exploits), or approximately determines $Q_t^*(x,a)$'s for all the (x,a,t)'s in a disjoint subset (explores). We outline the proof for Theorem 3 at the end of this subsection, and the detailed analysis is provided in the appendix. An immediate corollary bounds regret.

Corollary 2 *For any $\overline{R} \geq 0$, any system $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, F, R, S)$ with*

$$\sup_{(x,a,t)} |R_t(x,a)| \leq \overline{R},$$

and any time T, if OCP is applied with $\mathcal{Q}$ defined in Eqn(2.3), then

$$\mathrm{Regret}(T) \leq 2\overline{R}KH + 2\rho(H+1)T,$$

where K is the number of partitions and ρ is defined in Eqn(2.4).

Note that the regret bound in Corollary 2 is $O(T)$, and the coefficient of the linear term is $2\rho(H+1)$. Consequently, if Q^* is close to $\mathcal{Q}$, then the regret will increase slowly with T. Furthermore, the regret bound in Corollary 2 does not directly depend on $|\mathcal{S}|$ or $|\mathcal{A}|$.

We further notice that the threshold performance loss in Theorem 3 is $O(\rho H^2)$. The following proposition provides a condition under which the performance loss in one episode is $O(\rho H)$.

Proposition 1 *For any episode j, if $\mathcal{Q}_{\mathcal{C}} \subseteq \{Q \in \mathcal{Q} : L_{j,t} \leq Q_t(x_{j,t}, a_{j,t}) \leq U_{j,t}\}$, $\forall t = 0, \cdots, H-1$, then we have $V_0^*(x_{j,0}) - R^{(j)} \leq 6\rho H = O(\rho H)$.*

That is, if all the new constraints in an episode are redundant, then the performance loss in that episode is $O(\rho H)$. Note that if the condition for Proposition 1 holds in an episode, then $\mathcal{Q}_{\mathcal{C}}$ will not be modified at the end of that episode. Furthermore, if the system has a fixed initial state and the condition for Proposition 1 holds in one episode, then it will hold in all the subsequent episodes, and consequently, the performance losses in all the subsequent episodes are $O(\rho H)$.

Sketch of Proof for Theorem 3 and Proposition 1

We start by briefly describing how the constraint selection algorithm (Algorithm 1) updates $\mathcal{Q}_{\mathcal{C}}$'s for the function class $\mathcal{Q}$ specified in Eqn(2.3). Specifically, let $\theta_{t,k}$ denote the coefficient of the indicator function $\phi_{t,k}$, for any (t,k). Assume that (x,a,t) belongs to partition $\mathcal{Z}_{t,k}$, then, with $\mathcal{Q}$ specified in Eqn(2.3), $L \leq Q_t(x,a) \leq U$ is a constraint on and only on $\theta_{t,k}$, and is equivalent to $L \leq \theta_{t,k} \leq U$. By induction, it is straightforward to see in episode j, $\mathcal{Q}_{\mathcal{C}_j}$ can be represented as

$$\left\{\theta \in \Re^K : \underline{\theta}_{t,k}^{(j)} \leq \theta_{t,k} \leq \overline{\theta}_{t,k}^{(j)},\ \forall (t,k)\right\},$$

for some $\underline{\theta}_{t,k}^{(j)}$'s and $\overline{\theta}_{t,k}^{(j)}$'s. Note that $\underline{\theta}_{t,k}^{(j)}$ can be $-\infty$ and $\overline{\theta}_{t,k}^{(j)}$ can be ∞, and when $j=0$, $\overline{\theta}_{t,k}^{(0)} = \infty$ and $\underline{\theta}_{t,k}^{(0)} = -\infty$. Furthermore, from Algorithm 1, $\overline{\theta}_{t,k}^{(j)}$ is monotonically non-increasing in j, for all (t,k). Specifically, if OCP adds a new constraint $L \leq \theta_{t,k} \leq U$

on $\theta_{t,k}$ in episode j, we have $\overline{\theta}_{t,k}^{(j+1)} = \min\{\overline{\theta}_{t,k}^{(j)}, U\}$; otherwise, $\overline{\theta}_{t,k}^{(j+1)} = \overline{\theta}_{t,k}^{(j)}$. Thus, if $\overline{\theta}_{t,k}^{(j)} < \infty$, then $\overline{\theta}_{t,k}^{(j')} < \infty$, $\forall j' \geq j$.

For any $(x, a, t) \in \mathcal{Z}$, and any j, we define $Q^{\smiley}_{j,t}(x, a)$, the optimistic Q-function in episode j, as

$$Q^{\smiley}_{j,t}(x, a) = \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x, a), \tag{2.5}$$

and $Q^{\frownie}_{j,t}(x, a)$, the pessimistic Q-function in episode j, as

$$Q^{\frownie}_{j,t}(x, a) = \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x, a). \tag{2.6}$$

Clearly, if $(x, a, t) \in \mathcal{Z}_{t,k}$, then we have $Q^{\smiley}_{j,t}(x, a) = \overline{\theta}_{t,k}^{(j)}$, and $Q^{\frownie}_{j,t}(x, a) = \underline{\theta}_{t,k}^{(j)}$. Moreover, (x, a, t)'s in the same partition have the same optimistic and pessimistic Q-values.

It is also worth pointing out that by definition of ρ, if (x, a, t) and (x', a', t) are in the same partition, then we have

$$|Q_t^*(x, a) - Q_t^*(x', a')| \leq 2\rho. \tag{2.7}$$

To see it, let $\tilde{Q} \in \arg\min_{Q \in \mathcal{Q}} \|Q - Q^*\|_\infty$, then we have $|\tilde{Q}_t(x, a) - Q_t^*(x, a)| \leq \rho$ and $|\tilde{Q}_t(x', a') - Q_t^*(x', a')| \leq \rho$. Since $\tilde{Q} \in \mathcal{Q}$ and (x, a, t) and (x', a', t) are in the same partition, we have $\tilde{Q}_t(x, a) = \tilde{Q}_t(x', a')$. Then from triangular inequality, we have $|Q_t^*(x, a) - Q_t^*(x', a')| \leq 2\rho$.

The following lemma states that if $Q^{\smiley}_{j,t}(x, a) < \infty$, then it is "close" to $Q_t^*(x, a)$.

Lemma 4 $\forall (x, a, t)$ *and* $\forall j = 0, 1, \cdots$, *if* $Q^{\smiley}_{j,t}(x, a) < \infty$, *then*

$$|Q^{\smiley}_{j,t}(x, a) - Q_t^*(x, a)| \leq 2\rho(H - t). \tag{2.8}$$

Please refer to the appendix for the detailed proof of Lemma 4. Based on this lemma, we have the following result:

Lemma 5 *$\forall j = 0, 1, \cdots$, if $Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) < \infty$ for any $t = 0, 1, \cdots, H-1$, then we have*

$$V_0^*(x_{j,0}) - R^{(j)} \leq 2\rho H(H+1) = O\left(\rho H^2\right). \tag{2.9}$$

Furthermore, if the conditions of Proposition 1 hold, then we have $V_0^*(x_{j,0}) - R^{(j)} \leq 6\rho H = O(\rho H)$.

Please refer to the appendix for the detailed proof of Lemma 5. Obviously, Proposition 1 directly follows from Lemma 5.

For any $j = 0, 1, \cdots$, we define t_j^* as the last period t in episode j s.t. $Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) = \infty$. If $Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) < \infty$ for all $t = 0, 1, \cdots, H-1$, we define $t_j^* = \text{NULL}$. We then have the following lemma:

Lemma 6 *$\forall j = 0, 1, \cdots$, if $t_j^* \neq \text{NULL}$, then $\forall j' \leq j$, $Q^{\smiley}_{j',t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) = \infty$, and $\forall j' > j$, $Q^{\smiley}_{j',t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) < \infty$ (Exploration). Otherwise, if $t_j^* = \text{NULL}$, then $V_0^*(x_{j,0}) - R^{(j)} \leq 2\rho H(H+1)$ (Exploitation). Furthermore, $\sum_{j=0}^{\infty} \mathbf{1}[t_j^* \neq \text{NULL}] \leq K$, where K is the number of partitions.*

Again, please refer to the appendix for the proof of Lemma 6. Note that Theorem 3 directly follows from Lemma 6.

2.6 Computational Efficiency of Optimistic Constraint Propagation

We now briefly discuss the computational complexity of OCP. As typical in the complexity analysis of optimization algorithms, we assume that basic operations include the arithmetic operations, comparisons, and assignment, and measure computational complexity in terms of the number of basic operations (henceforth referred to as operations) per period.

First, it is worth pointing out that for a general hypothesis class $\mathcal{Q}$ and general action space $\mathcal{A}$, the per period computations of OCP can be intractable. This is because:

- Computing $\sup_{Q \in \mathcal{Q}_{\mathcal{C}}} Q_t(x_{j,t}, a)$, $U_{j,t}$ and $L_{j,t}$ requires solving a possibly intractable optimization problems.
- Selecting an action that maximizes $\sup_{Q \in \mathcal{Q}_{\mathcal{C}}} Q_t(x_{j,t}, a)$ can be intractable.

Further, the number of constraints in $\mathcal{C}$, and with it the number of operations per period, can grow over time.

However, if $|\mathcal{A}|$ is tractably small and $\mathcal{Q}$ has some special structures (e.g. $\mathcal{Q}$ is a finite set or a linear subspace or, more generally a polytope), then by discarding some "redundant" constraints in $\mathcal{C}$, OCP with a variant of Algorithm 1 will be computationally efficient, and the sample efficiency results developed in Section 2.5 will still hold. In this section, we discuss the scenario where $\mathcal{Q}$ is a polytope of dimension d. Note that the finite state/action *tabula rasa* case, the linear-quadratic case, and the case with linear combinations of disjoint indicator functions are all special cases of this scenario.

Specifically, if $\mathcal{Q}$ is a polytope of dimension d (i.e., within a d-dimensional subspace), then any $Q \in \mathcal{Q}$ can be represented by a weight vector $\theta \in \Re^d$, and $\mathcal{Q}$ can be characterized by a set of linear inequalities of θ. Furthermore, the new constraints of the form $L_{j,t} \le Q_t(x_{j,t}, a_{j,t}) \le U_{j,t}$ are also linear inequalities of θ. Hence, in each episode, $\mathcal{Q}_{\mathcal{C}}$ is characterized by a polyhedron in $\Re^d$, and $\sup_{Q \in \mathcal{Q}_{\mathcal{C}}} Q_t(x_{j,t}, a)$, $U_{j,t}$ and $L_{j,t}$ can be computed by solving linear programming (LP) problems. If we assume that each observed numerical value can be encoded by B bits, and LPs are solved by Karmarkar's algorithm [25], then the following proposition bounds the computational complexity.

Proposition 2 *If $\mathcal{Q}$ is a polytope of dimension d, each numerical value in the problem data or observed in the course of learning can be represented with B bits, and OCP uses Karmarkar's algorithm to solve linear programs, then the computational complexity of OCP is $O\left(\left[|\mathcal{A}| + |\mathcal{C}|\right] |\mathcal{C}| d^{4.5} B\right)$ operations per period.*

Proof: Note that OCP needs to perform the following computation in one period:

1. Construct $\mathcal{Q}_{\mathcal{C}}$ by constraint selection algorithm. This requires sorting $|\mathcal{C}|$ constraints by comparing their upper bounds and positions in the sequence (with $O\left(|\mathcal{C}| \log |\mathcal{C}|\right)$ operations), and checking whether $\mathcal{Q}_{\mathcal{C}} \cap \mathcal{C}_\tau \neq \varnothing$ for $|\mathcal{C}|$ times. Note that checking whether $\mathcal{Q}_{\mathcal{C}} \cap \mathcal{C}_\tau \neq \varnothing$ requires solving an LP feasibility problem with d variables and $O\left(|\mathcal{C}|\right)$ constraints.

2. Choose action $a_{j,t}$. Note that $\sup_{Q \in \mathcal{Q}_{\mathcal{C}}} Q_t(x_{j,t}, a)$ can be computed by solving an LP with d variables and $O\left(|\mathcal{C}|\right)$ constraints, thus $a_{j,t}$ can be derived by solving $|\mathcal{A}|$ such LPs.

3. Compute the new constraint $L_{j,t} \leq Q_t(x_{j,t}, a_{j,t}) \leq U_{j,t}$. Note $U_{j,t}$ can be computed by solving $|\mathcal{A}|$ LPs with d variables and $O\left(|\mathcal{C}|\right)$ constraints, and $L_{j,t}$ can be computed by solving one LP with d variables and $O\left(|\mathcal{C}| + |\mathcal{A}|\right)$ constraints.

If we assume that each observed numerical value can be encoded by B bits, and use Karmarkar's algorithm to solve LPs, then for an LP with d variables and m constraints, the number of bits input to Karmarkar's algorithm is $O\left(mdB\right)$, and hence it requires $O\left(mBd^{4.5}\right)$ operations to solve the LP. Thus, the computational complexities for the first, second, third steps are $O\left(|\mathcal{C}|^2 d^{4.5} B\right)$, $O\left(|\mathcal{A}||\mathcal{C}|d^{4.5}B\right)$ and $O\left(|\mathcal{A}||\mathcal{C}|d^{4.5}B\right)$, respectively. Hence, the computational complexity of OCP is $O\left(\left[|\mathcal{A}| + |\mathcal{C}|\right] |\mathcal{C}| d^{4.5} B\right)$ operations per period. **q.e.d.**

Notice that the computational complexity is polynomial in d, B, $|\mathcal{C}|$ and $|\mathcal{A}|$, and thus, OCP will be computationally efficient if all these parameters are tractably small. Note that the bound in Proposition 2 is a worst-case bound, and the $O(d^{4.5})$ term is incurred by the need to solve LPs. For some special cases, the computational complexity is much less. For instance, in the state aggregation case, the computational complexity is $O\left(|\mathcal{C}| + |\mathcal{A}| + d\right)$ operations per period.

As we have discussed above, one can ensure that $|\mathcal{C}|$ remains bounded by using variants of Algorithm 1 that discard the redundant constraints and/or update $\mathcal{Q}_{\mathcal{C}}$ more efficiently. Specifically, it is straightforward to design such constraint selection algorithms if $\mathcal{Q}$ is a coherent hypothesis class, or if $\mathcal{Q}$ is the span of pre-specified indicator functions over disjoint sets. Furthermore, if the notion of redundant constraints is properly defined, the sample efficiency results derived in Section 2.5 will still hold.

Chapter 3

Efficient Reinforcement Learning with Finite Hypothesis Class

3.1 Overview

In this chapter, we consider an RL problem in which an agent repeatedly interacts with a finite-horizon MDP, and has prior knowledge that the true value function Q^* is "close" to a known finite hypothesis class $\mathcal{Q}$. We reformulate this RL problem as an MAB problem, and propose a provably efficient RL algorithm based on an existing MAB algorithm. The sample complexity of our proposed algorithm is independent of state and action space cardinalities, and polynomial in other problem parameters.

This chapter is closely related to [29], which considers the case where the true environment is known to belong to a finite or compact class of models. The major difference between this chapter and [29] is that we consider value function generalization rather than model generalization. Thus, as we have discussed in Chapter 1, in our proposed algorithm, decisions can be made without solving a potentially intractable DP. Moreover, in this chapter, we do not require the true value function Q^* lies in the provided hypothesis class $\mathcal{Q}$; instead, our proposed algorithm addresses the agnostic learning case when $Q^* \notin \mathcal{Q}$, and $\mathcal{Q}$ can be an arbitrary finite hypothesis class. As we will detail in Section 3.3, the choice of $\mathcal{Q}$ affects both the sample complexity and computational complexity of our proposed algorithm.

The remainder of this chapter proceeds as follows. In Section 3.2, we briefly review the RL in episodic MDPs, and in Section 3.3, we present the MAB formulation and our proposed efficient RL algorithm. The notations in this chapter are summarized in Table 3.1.

Notation	Definition
$\mathcal{M}$	Finite-horizon MDP
$\mathcal{S}$	State space of the finite-horizon MDP
$\mathcal{A}$	Action space of the finite-horizon MDP
H	Time horizon of the finite-horizon MDP
P	Transition kernel of the finite-horizon MDP
R	Reward distributions of the finite-horizon MDP
π	Distribution of the initial state
h	Index of period in an episode
i, j	Index of episode
μ	Policy
V^{μ}	State value function under policy μ
V^{*}	Optimal state value function
μ^{*}	Optimal policy
Q^{*}	Action-contingent optimal value function
$R^{(i)}$	Realized reward in episode i
Regret(j)	Expected cumulative regret over the first j episodes
$\mathcal{Q}$	Finite hypothesis class
ρ	Distance between Q^{*} and $\mathcal{Q}$, in infinity norm
$\tilde{Q}$	Projection of Q^{*} on $\mathcal{Q}$, in infinity norm
μ_{Q}	Policy greedy to Q

Table 3.1: Notation for Chapter 3

3.2 Reinforcement Learning in Episodic MDP

In this chapter, we consider a reinforcement learning (RL) problem in which an agent interacts with a discrete-time finite-horizon Markov decision process (MDP) over a sequence of episodes. The MDP is identified by a sextuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, R, \pi)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, H is the horizon length, P encodes transition probabilities, R encodes reward distributions, and π is a probability distribution over $\mathcal{S}$. In each episode, the initial state x_0 is independently sampled from π. Furthermore, in period $h = 0, 1, \cdots, H-1$, if the state is x_h and an action a_h is selected then a subsequent state x_{h+1} is sampled from $P(\cdot|x_h, a_h)$ and a reward r_h is sampled from $R\left(\cdot|x_h, a_h, x_{h+1}\right)$. The episode terminates at the end of period $H-1$, after which a new episode begins.

Similar to the deterministic case described in Chapter 2, to represent the history of actions and observations over multiple episodes, we will often index variables by both episode and period. For example, x_{ih}, a_{ih} and r_{ih} respectively denote the state, action, and reward observed during period h in episode i[1]. As we have mentioned before, the initial state of episode i, x_{i0}, is independently sampled from π. The duration over which the agent has operated up to the beginning of period h of episode i is $iH + h$.

A policy $\mu = (\mu_0, \mu_1, \cdots, \mu_{H-1})$ is a sequence of functions, each mapping $\mathcal{S}$ to $\mathcal{A}$. For each policy μ, we define a value function $V_h^\mu(x) = \mathbb{E}\left[\sum_{\tau=h}^{H-1} r_\tau \middle| x_h = x, \mu\right]$, where $x_{\tau+1} \sim P(\cdot|x_\tau, a_\tau)$, $r_\tau \sim R\left(\cdot|x_\tau, a_\tau, x_{\tau+1}\right)$, $x_h = x$, and $a_\tau = \mu_\tau(x_\tau)$. The optimal value function is defined by $V_h^*(x) = \sup_\mu V_h^\mu(x)$ for all $h = 0, 1, \cdots, H-1$. A policy μ^* is said to be optimal if $V^{\mu^*} = V^*$. Throughout this chapter, we will restrict attention to MDPs with finite state and action spaces. Such MDPs always admit optimal policies.

It is also useful to define an action-contingent optimal value function:

$$Q_h^*(x, a) = \begin{cases} \mathbb{E}\left[r_h + V_{h+1}^*(x_{h+1}) \middle| x_h = x, a_h = a\right] & \text{if } h < H-1 \\ \mathbb{E}\left[r_h | x_h = x, a_h = a\right] & \text{if } h = H-1 \end{cases} \tag{3.1}$$

where $x_{h+1} \sim P(\cdot|x_h, a_h)$ and $r_h \sim R\left(\cdot|x_h, a_h, x_{h+1}\right)$. Then, a policy μ^* is optimal if and only if

$$\mu_h^*(x) \in \arg\max_{\alpha \in \mathcal{A}} Q_h^*(x, \alpha) \quad \forall x, h.$$

A reinforcement learning algorithm generates each action a_{ih} based on observations made up to period h of the episode i, including all states, actions, and rewards observed in previous episodes and earlier in the current episode, as well as the state

[1]If necessary, we use a comma to separate the episode-subscript and the period-subscript.

space $\mathcal{S}$, action space $\mathcal{A}$, horizon H, and possible prior information. In each episode, the algorithm realizes reward $R^{(i)} = \sum_{h=0}^{H-1} r_{ih}$. Note that $\mathbb{E}\left[R^{(i)} \middle| x_{i0}\right] \leq V_0^*(x_{i0})$ for all i. One way to quantify the performance of a reinforcement learning algorithm is in terms of the *expected cumulative regret* over j episodes, defined by

$$\text{Regret}(j) = \sum_{i=0}^{j-1} \mathbb{E}_{x_{i0}\sim\pi}\left[V_0^*(x_{i0}) - R^{(i)}\right]. \tag{3.2}$$

To simplify the exposition, in this chapter, we assume that the rewards are deterministic and do not depend on x_{h+1}. That is, in each period $h = 0, 1, \cdots, H-1$, if the state is x_h and action a_h is selected, then a deterministic reward $r_h = R(x_h, a_h) \in \Re$ is received. Since in this chapter we also restrict attention to MDPs with finite state and action spaces, this further implies that the rewards are bounded. We use $\overline{R}$ to denote an upper bound on the maximum magnitude of the rewards, i.e.

$$\max_{(x,a)\in\mathcal{S}\times\mathcal{A}} |R(x,a)| \leq \overline{R}. \tag{3.3}$$

It is worth pointing out that this assumption can be easily relaxed, and all the analysis in this chapter can be extended to the cases when there exist sub-Gaussian reward noises.

3.3 MAB Formulation and Efficient Reinforcement Learning Algorithm

In this chapter, we consider the scenario in which the agent has prior knowledge that the action-contingent optimal value function Q^* is "close" to a finite hypothesis class

$$\mathcal{Q} = \left\{Q^{(1)}, Q^{(2)}, \cdots, Q^{(K)}\right\}. \tag{3.4}$$

We use ρ to denote the distance between hypothesis class $\mathcal{Q}$ and Q^*, in infinity norm,

$$\rho = \min_{Q\in\mathcal{Q}} \|Q^* - Q\|_\infty = \min_{Q\in\mathcal{Q}} \max_{(x,a,t)} |Q_t(x,a) - Q_t^*(x,a)|. \tag{3.5}$$

We also use $\tilde{Q}$ to denote an arbitrary function $Q \in \mathcal{Q}$ that achieves this minimum, i.e.

$$\tilde{Q} \in \arg\min_{Q \in \mathcal{Q}} \|Q - Q^*\|_\infty. \tag{3.6}$$

Recall that in this chapter, we assume that x_{i0} is independently sampled from a fixed probability distribution π over $\mathcal{S}$. As we will discuss below, under this assumption, we can reformulate the RL problem as a multi-armed bandit (MAB) problem, and derive provably efficient RL algorithms based on existing provably efficient MAB algorithms (see, e.g., [28], [2], [39], [14] and references therein).

Specifically, for any function $Q : \mathcal{S} \times \mathcal{A} \times \{0, 1, \cdots, H-1\} \rightarrow \Re$, we use μ_Q to denote a policy greedy to Q, that is

$$\mu_{Q,h}(x) \in \arg\max_{a \in \mathcal{A}} Q_h(x, a), \quad \forall (x, h) \tag{3.7}$$

If there are multiple policies greedy to Q, then μ_Q can be chosen as an arbitrary policy satisfying Eqn(3.7). Consequently, the finite hypothesis class defined in Eqn(3.4) corresponds to K policies $\mu_{Q^{(1)}}, \cdots, \mu_{Q^{(K)}}$. To simplify the exposition, we use μ_k to denote $\mu_{Q^{(k)}}$ in the remainder of this section. Since $x_{i0} \sim \pi$, $\forall i = 0, 1, \cdots$, each policy μ_k corresponds to a fixed value (expected total reward) $\mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right]$. Furthermore, we define

$$k^* \in \arg\max_{k=1,\cdots,K} \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right],$$

that is, $\mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right]$ is the highest value achieved by the finite hypothesis class $\mathcal{Q}$.

The following lemma formalizes the result that if Q^* is "close" to the hypothesis class $\mathcal{Q}$, then the performance loss of $\mathcal{Q}$, which is defined as

$$\mathbb{E}_{x_0 \sim \pi}\left[V_0^*(x_0)\right] - \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] = \mathbb{E}_{x_0 \sim \pi}\left[V_0^*(x_0)\right] - \max_k \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right],$$

is also small.

Lemma 7 *With ρ defined in Eqn(3.5) and $\tilde{Q}$ defined in Eqn(3.6), we have*

$$\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] \geq \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{\tilde{Q}}}(x_0)\right] \geq \mathbb{E}_{x_0\sim\pi}\left[V_0^{*}(x_0)\right] - 2\rho H.$$

Please refer to the appendix for the proof of Lemma 7.

From Lemma 7, we see that if ρ is small, then a near-optimal policy can be derived by finding the best policy among $\mu_1, \cdots, \mu_K$. Since the initial state x_0 of the episodic MDP $\mathcal{M}$ is sampled from π, then if a policy μ is applied in episode i, then $R^{(i)}$, the realized total reward in episode i, is an unbiased estimate of $\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu}(x_0)\right]$. Furthermore, $R^{(i)}$ is bounded and $-\overline{R}H \leq R^{(i)} \leq \overline{R}H$.

Consequently, we can reformulate the RL problem in this case as an MAB problem with K "arms", where the kth arm of the bandit is policy μ_k. Notice that the reward of the kth arm is bounded by $\overline{R}H$ in magnitude, with mean $\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_k}(x_0)\right]$.

Many statistically efficient algorithms have been proposed for such MAB problems (see [28], [2], [39], [14] and references therein), and each efficient MAB algorithm will lead to an efficient RL algorithm in this case. In this section, we present one of such algorithms, which is based on the UCB1 algorithm proposed in [2].

The RL algorithm is presented in Algorithm 4, where $\tilde{V}_k$ is the average reward obtained from policy μ_k (arm k), and l_k is the number of times policy μ_k has been applied so far. This algorithm consists of two phases. In the first phase (from episode 0 to episode $K-1$), each policy μ_k is applied once and we use the observed realized rewards $R^{(i)}$'s to initialize $\tilde{V}_k$'s. In the second phase (from episode K on), the applied policy $\mu_{\tilde{k}}$ is selected based on the *optimism in the face of uncertainty (OFU)* principle, and is greedy to the upper confidence bounds

$$\tilde{V}_k + 2\overline{R}H\sqrt{\frac{2\log(i)}{l_k}} \quad \forall k = 1, \cdots, K.$$

Notice that the upper confidence radius $2\overline{R}H\sqrt{\frac{2\log(i)}{l_k}}$ follows from [2], and is based on Hoeffding's inequality. It is worth pointing out that there is no need to explicitly compute and store μ_k's, instead, when applying μ_k, the algorithm only needs to choose actions greedy to $Q^{(k)}$, with a fixed tie-breaking rule. Hence, if both K and $|\mathcal{A}|$ are tractably small, then Algorithm 4 is computationally efficient.

Algorithm 4 UCB Based RL Algorithm

```
Require: $\mathcal{S}$, $\mathcal{A}$, $H$, $\overline{R}$ and finite hypothesis class $\mathcal{Q}$ with $|\mathcal{Q}| = K$
  for episode $i = 0, 1, \cdots, K-1$ do
    Apply policy $\mu_{i+1}$ in episode $i$
    Initialize $\tilde{V}_{i+1} \leftarrow R^{(i)}$, and $l_{i+1} \leftarrow 1$
  end for
  for episode $i = K, K+1, \cdots$ do
    Choose
        $\tilde{k} \in \arg\max_k \left[ \tilde{V}_k + 2\overline{R}H\sqrt{\frac{2\log(i)}{l_k}} \right]$
    Apply policy $\mu_{\tilde{k}}$ in episode $i$, and observe $R^{(i)}$
    Update
        $\tilde{V}_{\tilde{k}} \leftarrow \frac{l_{\tilde{k}}}{l_{\tilde{k}}+1}\tilde{V}_{\tilde{k}} + \frac{1}{l_{\tilde{k}}+1}R^{(i)}$
        $l_{\tilde{k}} \leftarrow l_{\tilde{k}} + 1$
  end for
```

Based on Lemma 7, and following the regret analysis in [2], we can derive the following regret bound:

Theorem 4 *For any episodic MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, R, \pi)$ with finite $\mathcal{S}$ and $\mathcal{A}$ and deterministic rewards, if Algorithm 4 is applied with a finite hypothesis class $\mathcal{Q}$, then for any $j > K$, we have*

$$\text{Regret}(j) \leq 2\rho H j + 8\overline{R}H\sqrt{Kj\log(j)}, \tag{3.8}$$

where ρ is defined in Eqn(3.5), $\overline{R}$ is defined in Eqn(3.3), and $K = |\mathcal{Q}|$.

Roughly speaking, Theorem 4 is proved as follows. First, we can decompose Regret(j)

into two terms

$$\text{Regret}(j) = \left(\mathbb{E}_{x_0\sim\pi}\left[V_0^*(x_0)\right] - \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right]\right) j + \sum_{i=0}^{j-1}\left\{\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - \mathbb{E}\left[R^{(i)}\right]\right\}.$$

From Lemma 7, the first term is bounded by $2\rho H j$; and following the analysis in [2], the second term is bounded by $8\overline{R}H\sqrt{Kj\log(j)}$ under Algorithm 4. Please refer to the appendix for the proof of Theorem 4. Since we are primarily interested in the dependence of this regret bound on j, the number of episodes, we refer to the first term $2\rho H j$ as the "linear term", and the second term $8\overline{R}H\sqrt{Kj\log(j)}$ as the "sublinear term".

We now briefly discuss this regret bound. First, notice that this regret bound is independent of $|\mathcal{S}|$ and $|\mathcal{A}|$, and linear in other parameters (i.e. j, ρ, H, $\overline{R}$, $\sqrt{K}$). Second, note that the coefficient of the linear term is

$$2\rho H = 2H \min_{Q\in\mathcal{Q}} \|Q^* - Q\|_\infty.$$

Hence, the linear term will be small if $\mathcal{Q}$ is close to Q^*, and will be 0 if $Q^* \in \mathcal{Q}$ (the coherent learning case). Finally, notice that the *expected average regret* in the first j episodes is

$$\frac{1}{j}\text{Regret}(j) \le 2\rho H + 8\overline{R}H\sqrt{\frac{K\log(j)}{j}}.$$

Thus we have[2] $\lim_{j\to\infty}\frac{1}{j}\text{Regret}(j) \le 2\rho H$, hence $2\rho H$ is a bound on the *asymptotic expected average regret*. On the other hand, $8\overline{R}H\sqrt{K}$ can be viewed as a measure on *sample complexity*, since it determines how large j needs to be to ensure $8\overline{R}H\sqrt{\frac{K\log(j)}{j}}$ is acceptably small.

In this section, we make the simplifying assumptions that the episodic MDP $\mathcal{M}$ has finite state and action spaces and deterministic rewards. It is worth pointing out that both assumptions can be relaxed. Specifically, based on existing results in the field of MAB (see, e.g. [14]), we can derive a provably efficient RL algorithm as long

[2]Notice that the limit exists, since with probability 1, Algorithm 4 eventually learns to apply μ_{k^*}.

as the following conditions hold:

1. The expected rewards at all the state-action-state triple (x, a, y)'s are bounded. That is, there exists $\overline{R} \geq 0$ s.t.

$$|\mathbb{E}\left[R(\cdot|x, a, y)\right]| \leq \overline{R} \quad \forall (x, a, y) \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}.$$

2. The reward noises

$$R(\cdot|x, a, y) - \mathbb{E}\left[R(\cdot|x, a, y)\right]$$

   at all the state-action-state triple (x, a, y)'s are sub-Gaussian.

Notice that these new conditions no longer require the state and action spaces are finite.

Chapter 4

Randomized Least-Squares Value Iteration

4.1 Overview

As we have discussed in Chapter 1, an important challenge that remains is to design statistically and computationally efficient reinforcement learning (RL) algorithms that simultaneously generalize, explore, and learn from delayed consequences. In this chapter, we aim to make progress on this front. Similarly with the substantial value-based RL literature (see literature review in Chapter 1), we focus on an approach that generalizes through modeling the state-action value function as a linear combination of pre-selected basis functions. What distinguishes our approach from the pre-existing value function generalization literature is in how the agent explores. In particular, we consider incentivizing exploring through randomly perturbing value functions. As a specific algorithm of this kind, we propose *randomized least-squares value iteration (RLSVI)*. This algorithm fits value functions using randomly perturbed version of least-squares value iteration and selects actions that are greedy with respect to these value functions.

To put RLSVI, our proposed exploration scheme, in perspective, it is also useful to review existing work on exploration schemes that randomly perturb value functions. It is worth pointing out that for multi-armed bandit (MAB) problems, which can be considered as very special cases of RL problems, which are restricted to have a single state, Thompson sampling constitutes such an exploration scheme (see [45],[55]). Recent work [39] has established efficiency guarantees in this context and may serve as a stepping stone toward the analysis of RLSVI. Another related line of work [18] proposes Q-value sampling, a Bayesian exploration scheme with randomized value functions. The idea of Q-value sampling is to first sample state-action values (Q-values) based on the agent's posterior beliefs, and then select actions greedy with respect to the sampled state-action values. This is equivalent to sampling actions according to the posterior probability that they are optimal. This work does not, however, offer an approach to generalization. Furthermore, as the authors point out, certain assumptions (e.g. Assumption 4) in that paper are likely to be violated in practice. To the best of our knowledge, RLSVI is the first exploration scheme that randomly perturbs value functions for general RL problems with value function generalization.

On the other hand, the use of least-squares value iteration, however, is quite common to the RL literature. For example, it is commonly applied in conjunction with Boltzmann or ϵ-greedy exploration. We will explain in this chapter why these forms of exploration can lead to highly inefficient learning. We will also present computational results that demonstrate dramatic efficiency gains enjoyed by RLSVI relative to these alternatives.

As we have mentioned before, RL algorithms are often used to approximate solutions to large-scale dynamic programs. Thus, our algorithm and results also serve as contributions to dynamic programming (DP) and approximate dynamic programming (ADP).

The remainder of this chapter is organized as follows. We explain in Section 4.2 why Boltzmann and ϵ-greedy exploration can be highly inefficient. Next, in Section 4.3, we motivate and describe RLSVI. Experimental results are presented in Section 4.4. The version of RLSVI presented in Section 4.3, which serves as the subject of our experiments, is designed for episodic learning in a finite-horizon MDP. In Section 4.5, we present a variation of RLSVI that addresses continual learning in an infinite-horizon discounted MDP.

The notations in this chapter are summarized in Table 4.1.

Notation	Definition
$\mathcal{M}$	MDP
$\mathcal{S}$	State space of the MDP
$\mathcal{A}$	Action space of the MDP
H	Time horizon of the finite-horizon MDP
γ	Discount factor of the infinite-horizon MDP
P	Transition kernel of the MDP
R	Reward distributions of the MDP
π	Distribution of the initial state
h	Index of period in an episode
i, j	Index of episode
μ	Policy
V^μ	State value function under policy μ
V^*	Optimal state value function
μ^*	Optimal policy
Q^*	Action-contingent optimal value function
$R^{(i)}$	Realized reward in episode i
Regret(j)	Expected cumulative regret over the first j episodes
$\mathit{\Phi}_h$, $\mathit{\Phi}$	Generalization matrix
K	Number of basis functions
θ, $\hat{\theta}$	Coefficient vectors
λ	Regularization parameter in LSVI and RLSVI
ϵ	Exploration parameter in ϵ-greedy exploration
η	Exploration parameter in Boltzmann exploration
σ	Exploration parameter in randomized value function exploration

Table 4.1: Notation for Chapter 4

4.2 Randomized Actions

We first consider a reinforcement learning (RL) problem in which an agent interacts with a discrete-time finite-horizon Markov decision process (MDP) over a sequence of episodes. Please refer to Section 3.2 for the formulation of such RL problems. Notice that in this chapter, we do not assume that the rewards are deterministic. That is, in period $h = 0, 1, \cdots, H-1$, if the state is x_h and an action a_h is selected, then a subsequent state x_{h+1} is sampled from $P(\cdot|x_h, a_h)$ and a reward r_h is sampled from $R(\cdot|x_h, a_h, x_{h+1})$.

Reinforcement learning algorithms we consider in this chapter will be based on least-squares value iteration (LSVI). LSVI – as presented below as Algorithm 5[1] – can be applied by the agent at the beginning of each episode to estimate the optimal value function Q^* from data gathered over previous episodes as well as prior knowledge encoded in terms of generalization matrices and a regularization parameter λ. The algorithm iterates backwards over time periods in the planning horizon, in each iteration fitting a value function to the sum of immediate rewards and value estimates of the next period. Each value function is fitted via least-squares: note that vectors θ_{jh} satisfy

$$\theta_{jh} \in \underset{\zeta \in \Re^K}{\arg\min} \left(\|A\zeta - b\|^2 + \lambda \|\zeta\|^2 \right).$$

Algorithm 5 Least-Squares Value Iteration

```
Input: $\Phi_0, \dots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\{(x_{ih}, a_{ih}, r_{ih}) : i < j, h = 0, \dots, H-1\}$
Output: $\theta_{j0}, \dots, \theta_{j,H-1}$
1: $\theta_{jH} \leftarrow 0$, $\Phi_H \leftarrow \mathbf{0}$
2: for $h = H-1, \dots, 1, 0$ do
3:     Generate regression matrix and vector
           $A \leftarrow \begin{bmatrix} \Phi_h(x_{0h}, a_{0h}) \\ \vdots \\ \Phi_h(x_{j-1,h}, a_{j-1,h}) \end{bmatrix} \quad b \leftarrow \begin{bmatrix} r_{0,h} + \max_{\alpha \in \mathcal{A}} (\Phi_{h+1}\theta_{j,h+1})(x_{0,h+1}, \alpha) \\ \vdots \\ r_{j-1,h} + \max_{\alpha \in \mathcal{A}} (\Phi_{h+1}\theta_{j,h+1})(x_{j-1,h+1}, \alpha) \end{bmatrix}$
4:     Estimate value function
           $\theta_{jh} \leftarrow (A^\top A + \lambda I)^{-1} A^\top b$
5: end for
```

To fully specify a reinforcement learning algorithm, we must describe how the

[1]Notice that in Algorithm 5, when $j = 0$, matrix A and vector b are empty. In this case, we simply set $\theta_{j0} = \theta_{j1} = \cdots = \theta_{j,H-1} = 0$.

agent selects actions. Let us consider in this section algorithms that, during each episode, select actions that are nearly greedy with respect to value estimates, differing from greedy behavior only due to random perturbations. The form of these random perturbations is governed by an exploration scheme. We consider two popular exploration schemes: Boltzmann exploration and ϵ-greedy exploration (see, e.g., [36]). Reinforcement learning algorithms produced by synthesizing each of these schemes with LSVI are presented as Algorithms 6 and 7. Note that the "temperature" parameters η in Boltzmann exploration and ϵ in ϵ-greedy exploration control the degree to which random perturbations distort greedy actions.

Algorithm 6 LSVI with Boltzmann exploration

```
Input: $\Phi_0, \dots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\eta \in \Re_+$
1: for $j = 0, 1, \cdots$ do
2:     Compute $\theta_{j0}, \dots, \theta_{j,H-1}$ based on Algorithm 5
3:     Observe $x_{j0}$
4:     for $h = 0, 1, \dots, H-1$ do
5:         Sample $a_{jh} \sim \exp\left[(\Phi_h \theta_{jh})(x_{jh}, a)/\eta\right]$
6:         Observe $r_{jh}$ and $x_{j,h+1}$
7:     end for
8: end for
```

Algorithm 7 LSVI with ϵ-greedy exploration

```
Input: $\Phi_0, \dots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\lambda \in \Re_{++}$, $\epsilon \in [0, 1]$
1: for $j = 0, 1, \dots$ do
2:     Compute $\theta_{j0}, \dots, \theta_{j,H-1}$ using Algorithm 5
3:     Observe $x_{j0}$
4:     for $h = 0, 1, \cdots, H-1$ do
5:         Sample $\xi \sim \text{Bernoulli}(\epsilon)$
6:         if $\xi = 1$ then
7:             Sample $a_{jh} \sim \text{unif}(\mathcal{A})$
8:         else
9:             Sample $a_{jh} \sim \text{unif}\left(\arg\max_{\alpha \in \mathcal{A}} (\Phi_t \theta_{jh})(x_{jh}, \alpha)\right)$
10:        end if
11:        Observe $r_{jh}$ and $x_{j,h+1}$
12:    end for
13: end for
```

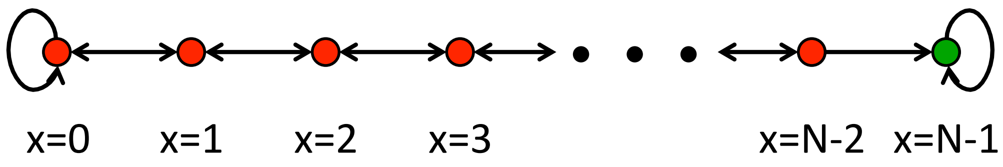

Figure 4.1: MDP for which Boltzmann/ϵ-greedy exploration is inefficient.

Unfortunately, both reinforcement learning algorithms we have described exhibit worst-case regret that grows exponentially in H and/or $|\mathcal{S}|$. More intelligent exploration schemes are necessary to achieve polynomial regret, even in the case of *tabula rasa* learning, in which each Φ_h is an identity matrix and, therefore, the agent does not generalize across state-action pairs. In the remainder of this section, we provide an example for which LSVI with Boltzmann exploration or ϵ-greedy exploration require exponentially many episodes to learn an optimal policy, even in a coherent learning context with a small number of basis functions.

Example 2 *Consider the MDP illustrated in Figure 4.1. Each node represents a state, and each arrow corresponds to a possible state transition. The state space is $\mathcal{S} = \{0, 1, \cdots, N-1\}$ and the action space is $\mathcal{A} = \{a^{(1)}, a^{(2)}\}$. If the agent takes action $a^{(1)}$ at state $x = 0, 1, \cdots, N-2$ (the red nodes), the the state transitions to $y = [x-1]^+$. On the other hand, if the agent takes action $a^{(2)}$ at state $x = 0, 1, \cdots, N-2$, the state transitions to $y = x+1$. State $N-1$ (the green node) is absorbing. We assume a reward of 0 is realized upon any transition from a red node and a reward of 1 is realized upon any transition from the green node. We take the horizon H to be equal to the number of states N. The initial state in any episode is 0.*

Suppose we apply LSVI with either Boltzmann or ϵ-greedy exploration. Let i^* be the first episode during which state $N-1$ is visited. It is easy to see that $\theta_{jh} = 0$ for all h and all $i < i^*$. Furthermore, with either exploration scheme, actions are sampled uniformly at random over episodes $i < i^*$. Thus, in any episode $i < i^*$, the green node will be reached if and only if the algorithm samples $a^{(2)}$ consecutively in periods $t = 0, 1, \cdots, H-2$. The probability of this event is $2^{-(H-1)} = 2^{-(N-1)} = 2^{-(|\mathcal{S}|-1)}$. It follows that $E[i^*] \geq 2^{|\mathcal{S}|-1}$. Further, since the optimal expected value over each *episode is* 1 *and the realized reward over each episode* $i < i^*$ *is* 0*, it follows that*

$$\begin{aligned}\text{Regret}(j) &\geq \sum_{i=0}^{j-1} \Pr(i < i^*) \\ &= \sum_{i=0}^{j-1} \left(1 - 2^{-(|\mathcal{S}|-1)}\right)^{i+1} \\ &= \left(2^{|\mathcal{S}|-1} - 1\right)\left(1 - \left[1 - 2^{-(|\mathcal{S}|-1)}\right]^j\right). \qquad (4.1)\end{aligned}$$

Further,

$$\liminf_{j \to \infty} \text{Regret}(j) \geq 2^{|\mathcal{S}|-1} - 1.$$

This example establishes that regret realized by LSVI with Boltzmann or ϵ-greedy exploration can grow exponentially in the number of states. This is far from what one would hope for, which is regret that is independent of the number of states and instead a low-order polynomial in the number of basis functions. Note that the lower bound established for this example applies to both coherent and agnostic learning for any choice of the generalization matrices $\Phi_0, \ldots, \Phi_{H-1}$, any regularization parameter $\lambda > 0$, and any temperature parameter $\eta \geq 0$ or $\epsilon \in [0, 1]$.

4.3 Randomized Value Functions

RL algorithms of the previous section explore through randomly perturbing greedy actions. We now consider an alternative approach to exploration that involves randomly perturbing value functions rather than actions. As a specific scheme of this kind, we propose randomized least-squares value iteration (RLSVI), which we present as Algorithm 8.[2]

To obtain an RL algorithm, we simply select greedy actions in each episode, as specified in Algorithm 9. Note that RLSVI randomly perturbs value estimates in directions of significant uncertainty to incentivize exploration. This approach relates

[2]Similarly as in Algorithm 5, when $j = 0$, we set $\hat{\theta}_{j0} = \hat{\theta}_{j1} = \cdots = \hat{\theta}_{j,H-1} = 0$.

Algorithm 8 Randomized Least-Squares Value Iteration

```
Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\{(x_{ih}, a_{ih}, r_{ih}) : i < j, h = 0, \ldots, H-1\}$
Output: $\hat{\theta}_{j0}, \ldots, \hat{\theta}_{j,H-1}$
1: $\hat{\theta}_{jH} \leftarrow 0$, $\Phi_H \leftarrow \mathbf{0}$
2: for $h = H-1, \ldots, 1, 0$ do
3:     Generate regression matrix and vector
           $$A \leftarrow \begin{bmatrix} \Phi_h(x_{0h}, a_{0h}) \\ \vdots \\ \Phi_h(x_{j-1,h}, a_{j-1,h}) \end{bmatrix} \qquad b \leftarrow \begin{bmatrix} r_{0,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\hat{\theta}_{j,h+1}\right)(x_{0,h+1}, \alpha) \\ \vdots \\ r_{j-1,h} + \max_{\alpha \in \mathcal{A}} \left(\Phi_{h+1}\hat{\theta}_{j,h+1}\right)(x_{j-1,h+1}, \alpha) \end{bmatrix}$$
4:     Estimate value function
           $$\overline{\theta}_{jh} \leftarrow (A^\top A + \lambda\sigma^2 I)^{-1} A^\top b \qquad \Sigma_{jh} \leftarrow \left(\frac{1}{\sigma^2} A^\top A + \lambda I\right)^{-1}$$
5:     Sample $\hat{\theta}_{jh} \sim N(\overline{\theta}_{jh}, \Sigma_{jh})$
6: end for
```

to Thompson sampling (see, e.g., [45, 39]). We conjecture that Algorithm 9 is statistically efficient in the sense that for any j, Regret(j) is bounded by a low-order polynomial function of K and H. Further, it is easy to see that this RL algorithm is computationally efficient in the sense that the computational requirements per time period can be bounded by a low-order polynomial function of K, H, and $|\mathcal{A}|$.

4.4 Experimental Results

In this section, we report results of applying RLSVI to Example 2, with $N = |\mathcal{S}| = H = 50$. We consider both coherent and agnostic learning contexts. Our results demonstrate dramatic efficiency gains relative to LSVI with Boltzmann or ϵ-greedy exploration. The results also illustrate how performance depends on the number of basis functions, algorithm parameters, and, in the agnostic case, the distance between the optimal value function and the span of the basis functions.

Algorithm 9 RLSVI with Greedy Action

```
Input: $\Phi_0, \dots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$
1: for $j = 0, 1, \cdots$ do
2:     Compute $\hat{\theta}_{j0}, \dots, \hat{\theta}_{j,H-1}$ using Algorithm 8
3:     Observe $x_{j0}$
4:     for $h = 0, \cdots, H-1$ do
5:         Sample $a_{jh} \sim \text{unif}\left(\arg\max_{\alpha \in \mathcal{A}} \left(\Phi_h \hat{\theta}_{jh}\right)(x_{jh}, \alpha)\right)$
6:         Observe $r_{jh}$ and $x_{j,h+1}$
7:     end for
8: end for
```

To estimate the performance of RLSVI, we average the computation results over $M = 200$ independent simulations, each over $j = 400$ episodes. Based on data produced from these simulations, we compute for each ith episode an estimate of the episode regret

$$\hat{\Delta}(i) = V_0^*(x_{i0}) - \frac{1}{M} \sum_{m=1}^{M} R^{(mi)},$$

where $R^{(mi)}$ is the reward realized over the ith episode of the mth simulation.

Each of the M independent simulations is of the same MDP. If we also used the same basis functions and algorithm parameters in each case, the accumulation

$$\sum_{i=0}^{j-1} \left(V_0^*(x_{i0}) - \frac{1}{M} \sum_{m=1}^{M} R^{(mi)} \right)$$

of our episode regret estimates would be an unbiased and consistent estimator of Regret(j). However, so that our results do not depend on a specific selection of basis functions, we will randomly sample a separate set of basis functions for each of the M simulations, using a sampling scheme that we will describe later.

As discussed in Section 4.2, for the MDP under consideration with *any* choice of basis functions and algorithm parameters, LSVI with either Boltzmann exploration or ϵ-greedy exploration, Regret$(400) \geq 400 - 6.82 \times 10^{-13} \approx 400$. This implies that the episode regret is approximately 1 for each of the first 400 episodes. This level of performance offered by the alternative algorithms will serve as baseline as we assess performance of RLSVI.

4.4.1 Coherent Learning

In the coherent learning case, $Q^*_h \in \text{span}\,[\Phi_h]$ for each period h. Recall that we use ϕ_{hk} to denote the kth basis function, which is the kth column of Φ_h. Each set of $M = 200$ simulations that we carry out to obtain estimates of episode regret is conducted in a manner specified by Algorithm 10

Algorithm 10 Coherent Learning Simulation

```
Input: $K \in \mathbb{Z}_{++}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $j \in \mathbb{Z}_{++}$, $M \in \mathbb{Z}_{++}$
Output: $\hat{\Delta}(0), \ldots, \hat{\Delta}(j-1)$
  for $m = 1, \ldots, M$ do
    for $h = 0, \ldots, H-1$ do
      for $k = 1, \ldots, K$ do
        switch $(k)$
        case 1:
          $\phi_{hk} \leftarrow Q^*_h$
        case 2:
          $\phi_{hk} \leftarrow \mathbf{1}$
        default:
          Sample $\phi_{hk} \sim N(\mathbf{0}, I)$
        end switch
      end for
    end for
    Simulate RLSVI over $j$ episodes; observe episode rewards $R^{(m0)}, \ldots, R^{(m,j-1)}$
  end for
  for $i = 0, 1, \cdots, j-1$ do
    $\hat{\Delta}(i) = V^*_0(x_{i0}) - \frac{1}{M}\sum_{m=1}^{M} R^{(mi)}$
  end for
```

We first demonstrate how the experimental results vary with K, the number of basis functions. Here, we let $\lambda = 1$, $\sigma^2 = 10^{-3}$, $j = 400$, $M = 200$, and $K = 2, 5, 10, 20$. Resulting estimates of episode regret are plotted in Figure 4.2. These results demonstrate that with up to 20 basis functions RLSVI with $\lambda = 1$ and $\sigma^2 = 10^{-3}$ generally takes less than 250 episodes to learn the optimal policy. This represents a dramatic efficiency gain relative to LSVI with Boltzmann or ϵ-greedy exploration, each of which would take more than 5.63×10^{14} episodes. The results also indicate that larger values of K call for longer learning times and larger cumulative regret. This makes sense since larger K implies weaker prior knowledge and therefore a more to learn.

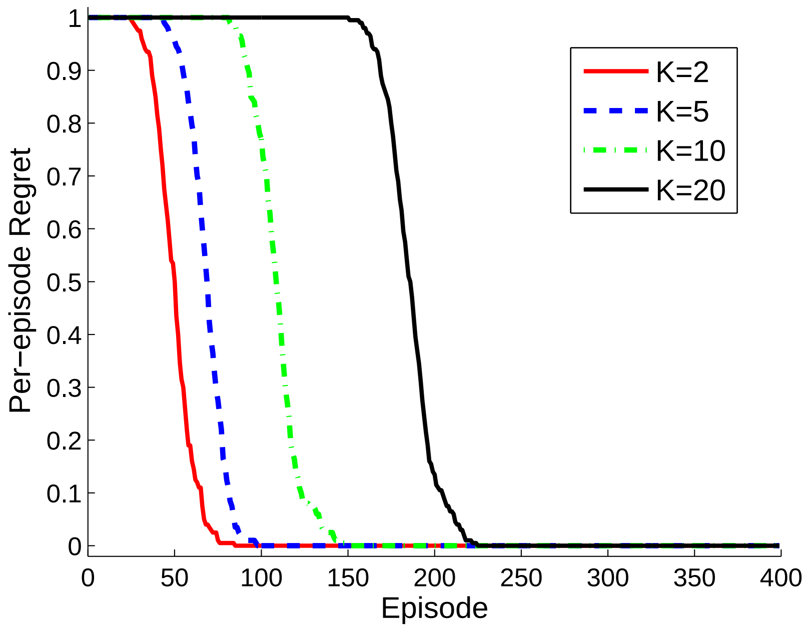

Figure 4.2: Episode regret estimates for various numbers of basis functions.

Next, we examine how results vary with the algorithm parameters λ and σ^2. Figure 4.3 plots results from simulations with $K = 20$, $\lambda = 1$, $M = 200$, $j = 400$, and $\sigma^2 = 10^{-12}, 3.5 \times 10^{-12}, 10^{-9}, 10^{-6}, 10^{-4}, 10^{-2}, 1$. These results demonstrate that for the values of σ^2 considered from 10^{-9} to 10^{-4}, RLSVI generally learns the optimal policy in around 200 episodes; on the other hand, for values of 10^{-12}, 3.5×10^{-12}, 10^{-2} and 1, RLSVI does not learn the optimal policy within 400 episodes. It makes sense that large values of σ^2 and small values of σ^2 lead to a long learning times, since large values of σ^2 induce excessive exploration, while small values of σ^2 lead to insufficient exploration.

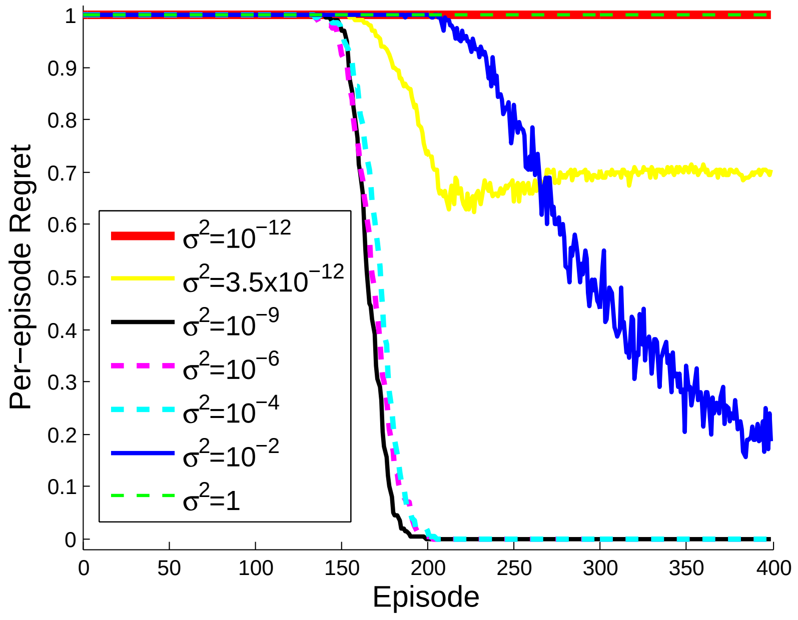

Figure 4.3: Episode regret estimates for various values of σ^2.

Figure 4.4 plots results from simulations with $K = 20$, $\sigma^2 = 10^{-4}$, $M = 200$, $j = 400$ and $\lambda = 2.5 \times 10^{-8}, 10^{-4}, 10^{-2}, 1, 10^2, 10^4$. These results indicate that for a wide range of settings for λ, RLSVI generally learns the optimal policy in less than 350 episodes. These results also suggest that large values of λ and small values of λ result in a long learning time. This makes sense, since large values of λ lead to over-regularization, which makes basis function weights very small, restricting exploration. On the other hand, small values of λ can give rise to matrices Σ_{jh} with large eigenvalues, which leads to excessive exploration.

4.4.2 Agnostic Learning

Our experiments with agnostic learning are conducted in a manner similar to those of the coherent learning case. The procedure, specified as Algorithm 11, differs only in its construction of the basis functions for each time period. Specifically, in the agnostic learning case, we choose $\phi_{h1} = \mathbf{1}$ and $\phi_{hk} = Q_h^* + \rho\psi_{hk}$ for any $k = 2, 3, \cdots, K$, where ψ_{hk} is sampled independently from $N(\mathbf{0}, I)$ and $\rho \geq 0$ is a chosen scalar. Note that if

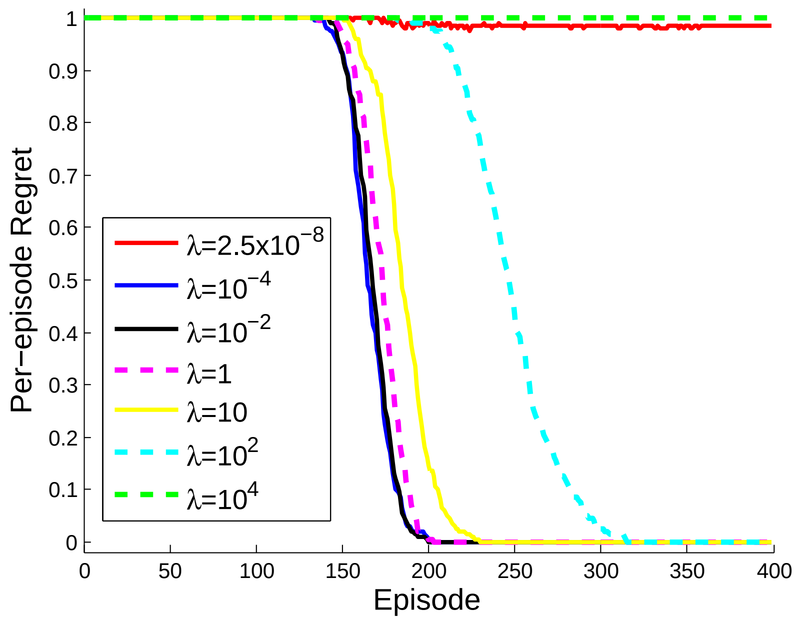

Figure 4.4: Episode regret estimates for various values of λ.

$\rho = 0$, this case reduces to a coherent learning case. As ρ grows, the basis functions are increasingly distorted and the distance between the optimal value function and the span of the basis functions is likely to grow.

Figure 4.5 plots cumulative regret estimates as a function of ρ, with $K = 11$, $\lambda = 1$, $\sigma^2 = 10^{-3}$, $M = 200$, $j = 400$ and $\rho = 0, 0.01, 0.02, \cdots, 0.1$. Also plotted is an upper bound on cumulative regret, which is also approximately equal to the level of regret realized by LSVI with Boltzmann or ϵ-greedy exploration. These results demonstrate that regret grows with ρ, but that regret realized by RLSVI over the first 400 episodes remains superior to the alternatives for a range of settings.

4.5 RLSVI in Discounted Infinite-Horizon MDPs

In this section, we propose a version of randomized least-squares value iteration (RLSVI) for reinforcement learning in infinite-horizon discounted MDPs. A discounted MDP is identified by a sextuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \gamma, P, R, \pi)$, where $\mathcal{S}$ is a state

Algorithm 11 Agnostic Learning Simulation

```
Input: $K \in \mathbb{Z}_{++}$, $\rho \in \Re_{+}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $j \in \mathbb{Z}_{++}$, $M \in \mathbb{Z}_{++}$
Output: $\hat{\Delta}(0), \ldots, \hat{\Delta}(j-1)$
  for $m = 1, \ldots, M$ do
    for $h = 0, \ldots, H-1$ do
      for $k = 1, \ldots, K$ do
        switch $(k)$
        case $1$:
          $\phi_{hk} \leftarrow \mathbf{1}$
        default:
          Sample $\psi_{hk} \sim N(\mathbf{0}, I)$
          $\phi_{hk} \leftarrow Q^*_h + \rho\psi_{hk}$
        end switch
      end for
    end for
    Simulate RLSVI over $j$ episodes; observe episode rewards $R^{(m0)}, \ldots, R^{(m,j-1)}$
  end for
  for $i = 0, 1, \cdots, j-1$ do
    $\hat{\Delta}(i) = V^*_0(x_{i0}) - \frac{1}{M}\sum_{m=1}^{M} R^{(mi)}$
  end for
```

space, $\mathcal{A}$ is an action space, $\gamma \in (0,1)$ is the discount factor, P encodes transition probabilities, R encodes reward distributions, and π is a distribution over $\mathcal{S}$. In each time $t = 0, 1, \ldots$, if the state is x_t and an action a_t is selected then a subsequent state x_{t+1} is sampled from $P(\cdot|x_t, a_t)$ and a reward r_t is sampled from $R\left(\cdot|x_t, a_t, x_{t+1}\right)$. The initial state x_0 is sampled from π.

A (stationary) policy μ is a function mapping $\mathcal{S}$ to $\mathcal{A}$. For each policy μ, we define a value function $V^\mu(x) = \mathbb{E}\left[\sum_{\tau=0}^{\infty} \gamma^\tau r_\tau | x_0 = x, \mu\right]$, where $x_{\tau+1} \sim P(\cdot|x_\tau, a_\tau)$, $r_\tau \sim R\left(\cdot|x_\tau, a_\tau, x_{\tau+1}\right)$, $x_0 = x$, and $a_\tau = \mu(x_\tau)$. The optimal value function is defined by $V^*(x) = \sup_\mu V^\mu(x)$, and a policy μ^* is said to be optimal if $V^{\mu^*} = V^*$. We restrict attention to MDPs with finite state and action spaces. Such MDPs always admit optimal policies. An action-contingent optimal value function is defined by

$$Q^*(x,a) = \mathbb{E}\left[r_t + \gamma V^*(x_{t+1}) | x_t = x, a_t = a\right], \tag{4.2}$$

where $x_{t+1} \sim P(\cdot|x_t, a_t)$ and $r_t \sim R\left(\cdot|x_t, a_t, x_{t+1}\right)$. Note that a policy μ^* is optimal if

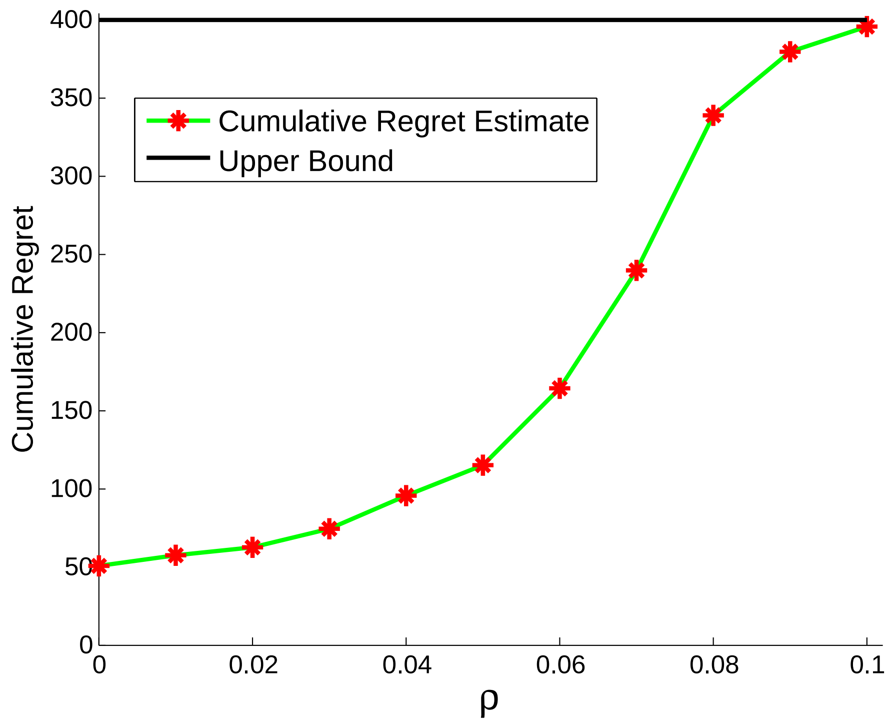

Figure 4.5: Estimate of cumulative regret over 400 episodes as a function of ρ.

and only if

$$\mu^*(x) \in \arg\max_{\alpha \in \mathcal{A}} Q^*(x, \alpha) \quad \forall x.$$

Similarly with the episodic case, a reinforcement learning algorithm generates each action a_t based on observations made up to time t, including all states, actions, and rewards observed in previous time steps, as well as the state space $\mathcal{S}$, action space $\mathcal{A}$, discount factor γ, and possible prior information. The realized discounted reward from time t on is $R^{(t)} = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$. Note that $\mathbb{E}\left[R^{(t)} \middle| x_t\right] \leq V^*(x_t)$ for all t. We will quantify the performance of a reinforcement learning algorithm in terms of the *expected cumulative regret* over the first T time periods, defined by

$$\text{Regret}(T) = \sum_{t=0}^{T-1} \mathbb{E}\left[V^*(x_t) - R^{(t)}\right]. \tag{4.3}$$

RLSVI for discounted-infinite horizon problems is presented in Algorithm 12. Similarly to Algorithm 8, Algorithm 12 randomly perturbs value estimates in directions of significant uncertainty to incentivize exploration. Note that the random perturbation vectors $w_{t+1} \sim N(\sqrt{1-\gamma^2}w_t, \gamma^2 \Sigma_{t+1})$ are sampled to ensure autocorrelation and that marginal covariance matrices of consecutive perturbations differ only slightly. In each period, a greedy action is selected. Avoiding frequent abrupt changes in the perturbation vector is important as this allows the agent to execute on multi-period plans to learn new information.

Algorithm 12 Randomized Least-Squares Value Iteration (Discounted MDP)

```
Input: $\hat{\theta}_t \in \Re^K$, $w_t \in \Re^K$, $\Phi \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\gamma \in (0,1)$, $\{(x_\tau, a_\tau, r_\tau) : \tau \le t\}$, $x_{t+1}$
Output: $\hat{\theta}_{t+1} \in \Re^K$, $w_{t+1} \in \Re^K$
1: Generate regression matrix and vector
   $$A \leftarrow \begin{bmatrix} \Phi(x_0, a_0) \\ \vdots \\ \Phi(x_t, a_t) \end{bmatrix} \qquad b \leftarrow \begin{bmatrix} r_0 + \max_{\alpha \in \mathcal{A}} \left(\Phi\hat{\theta}_t\right)(x_1, \alpha) \\ \vdots \\ r_t + \max_{\alpha \in \mathcal{A}} \left(\Phi\hat{\theta}_t\right)(x_{t+1}, \alpha) \end{bmatrix}$$
2: Estimate value function
   $$\overline{\theta}_{t+1} \leftarrow (A^\top A + \lambda\sigma^2 I)^{-1} A^\top b \qquad \Sigma_{t+1} \leftarrow \left(\frac{1}{\sigma^2} A^\top A + \lambda I\right)^{-1}$$
3: Sample $w_{t+1} \sim N(\sqrt{1-\gamma^2}w_t, \gamma^2 \Sigma_{t+1})$
4: Set $\hat{\theta}_{t+1} = \overline{\theta}_{t+1} + w_{t+1}$
```

Algorithm 13 RLSVI with Greedy Action (Discounted MDP)

```
Input: $\Phi_0, \ldots, \Phi_{H-1} \in \Re^{|\mathcal{S}||\mathcal{A}| \times K}$, $\sigma \in \Re_{++}$, $\lambda \in \Re_{++}$, $\gamma \in (0,1)$
1: Set $\hat{\theta}_0 = 0$ and observe $x_0$
2: for $t = 0, 1, \cdots$ do
3:     Sample $a_t \sim \text{unif}\left(\arg\max_{\alpha \in \mathcal{A}} \left(\Phi \hat{\theta}_t\right)(x_t, \alpha)\right)$
4:     Observe $r_t$ and $x_{t+1}$
5:     Compute $\hat{\theta}_{t+1}$ using Algorithm 12
6: end for
```

Chapter 5

Conclusion and Future Work

Finally, we conclude this dissertation and discuss some possible future work. As we have discussed in Chapter 1, fueled by modern IT, reinforcement learning (RL) is becoming increasingly important in a wide variety of applications. However, efficient RL is nontrivial and requires a carefully designed exploration scheme. One open issue in this field is to develop efficient RL algorithms that leverage value function generalization. In this dissertation, we aim to make progress in this direction.

Our contribution is twofold. First, we have developed provably efficient algorithms for

1. RL in episodic deterministic systems (Chapter 2). Our proposed algorithm, optimistic constraint propagation (OCP), is efficient in both the coherent learning cases with general value function generalization, and the state aggregation case, a special agnostic learning case.

2. RL in episodic MDPs with a finite hypothesis class (Chapter 3). We reformulate this RL problem as an MAB problem, and derive an efficient RL algorithm for this case based on an existing MAB algorithm.

Second, we have proposed randomized least-square value iteration (RLSVI), a practical algorithm in RL with linear hypothesis class (Chapter 4). To the best of our knowledge, RLSVI is the first exploration scheme that randomly perturbs value functions for general RL problems with value function generalization, and experiment results suggest that it is promising.

In the future, we plan to analyze RLSVI and test it extensively. We conjecture that it is provably statistically efficient, and hope to prove this in the future. We also plan to test it on practical RL problems, including recommendation systems (under an RL formulation), device management in smart home, and petroleum reservoir production optimization discussed in Chapter 1. We also plan to propose a version of RLSVI for the average reward RL problem.

Appendix A

Proofs

A.1 Eluder Dimension for the Sparse Linear Case

We start by defining some useful terminologies and notations. For any $\theta \in \Re^K$, any $l \leq K$ and any index set $\mathcal{I} = \{i_1, i_2, \cdots, i_l\} \subseteq \{1, 2, \cdots, K\}$ with $i_1 < i_2 < \cdots < i_l$ and $|\mathcal{I}| = l \leq K$, we use $\theta_{\mathcal{I}}$ to denote the subvector of θ associated with the index set $\mathcal{I}$, i.e. $\theta_{\mathcal{I}} = [\theta_{i_1}, \theta_{i_2} \cdots, \theta_{i_l}]^T$.

For a sequence of vectors $\theta^{(1)}, \theta^{(2)}, \cdots \in \Re^K$, we say $\theta^{(k)}$ is linearly l-independent of its predecessors if there exists an index set $\mathcal{I}$ with $|\mathcal{I}| = l$ s.t. $\theta^{(k)}_{\mathcal{I}}$ is linearly independent of $\theta^{(1)}_{\mathcal{I}}, \theta^{(2)}_{\mathcal{I}}, \cdots, \theta^{(k-1)}_{\mathcal{I}}$. Let $N = |\mathcal{S}||\mathcal{A}|$, and use Φ_j^T to denote the jth row of Φ. For any $l \leq K$, we define $\text{rank}[\Phi, l]$, the l-rank of Φ, as the length d of the longest sequence of Φ_j's such that every element is linearly l-independent of its predecessors. Recall that $\mathcal{Q}_0 = \left\{\Phi\theta : \theta \in \Re^K, \|\theta\|_0 \leq K_0\right\}$, we have the following result:

Proposition 3 *If* $2K_0 \leq K$*, then* $\dim_{\mathrm{E}}[\mathcal{Q}_0] = \text{rank}[\Phi, 2K_0]$.

Proof: We use $y = (x, a)$ to denote a state-action pair, and use $\Phi(y)^T$ to denote the row of matrix Φ associated with y. Based on our definitions of eluder dimension and l-rank, it is sufficient to prove the following lemma:

Lemma 8 *For any state-action pair* y *and for any set of state-action pairs* $Y =$

$\{y^{(1)}, y^{(2)}, \cdots, y^{(n)}\}$, *$y$ is independent of Y with respect to $\mathcal{Q}_0$ if and only if $\Phi(y)$ is linearly $2K_0$-independent of* $\{\Phi(y^{(1)}), \Phi(y^{(2)}), \cdots, \Phi(y^{(n)})\}$.

We now prove the above lemma. Note that based on the definition of independence (see Subsection 2.5.1), y is independent of Y with respect to $\mathcal{Q}_0$ if and only if there exist $Q_1, Q_2 \in \mathcal{Q}_0$ s.t. $Q_1(y^{(i)}) = Q_2(y^{(i)})$, $\forall i = 1, 2, \cdots, n$, and $Q_1(y) \neq Q_2(y)$. Based on the definition of function space $\mathcal{Q}_0$, there exist two K_0-sparse vectors $\theta^{(1)}, \theta^{(2)} \in \Re^K$ s.t. $Q_1 = \Phi\theta^{(1)}$ and $Q_2 = \Phi\theta^{(2)}$. Thus, y is independent of Y with respect to $\mathcal{Q}_0$ if and only if there exist two K_0-sparse vectors $\theta^{(1)}, \theta^{(2)} \in \Re^K$ s.t.

$$\begin{aligned} \Phi(y^{(i)})^T(\theta^{(1)} - \theta^{(2)}) &= 0 \quad \forall i = 1, 2, \cdots, n \\ \Phi(y)^T(\theta^{(1)} - \theta^{(2)}) &\neq 0 \end{aligned} \tag{A.1}$$

Based on the definition of K_0-sparsity, the above condition is equivalent to there exists a $2K_0$-sparse vector $\theta \in \Re^K$ s.t.

$$\begin{aligned} \Phi(y^{(i)})^T\theta &= 0 \quad \forall i = 1, 2, \cdots, n \\ \Phi(y)^T\theta &\neq 0 \end{aligned} \tag{A.2}$$

To see it, note that if $\theta^{(1)}, \theta^{(2)}$ are K_0-sparse, then $\theta = \theta^{(1)} - \theta^{(2)}$ is $2K_0$-sparse. On the other hand, if θ is $2K_0$-sparse, then there exist two K_0-sparse vectors $\theta^{(1)}, \theta^{(2)}$ s.t. $\theta = \theta^{(1)} - \theta^{(2)}$.

Since θ is $2K_0$-sparse, there exists a set of indices $\mathcal{I}$ s.t. $|\mathcal{I}| = 2K_0$ and $\theta_i = 0$, $\forall i \notin \mathcal{I}$. Thus, the above condition is equivalent to

$$\begin{aligned} \Phi(y^{(i)})_{\mathcal{I}}^T\theta_{\mathcal{I}} &= 0 \quad \forall i = 1, 2, \cdots, n \\ \Phi(y)_{\mathcal{I}}^T\theta_{\mathcal{I}} &\neq 0, \end{aligned} \tag{A.3}$$

which is further equivalent to $\Phi(y)_{\mathcal{I}}$ is linearly independent of

$$\Phi(y^{(1)})_{\mathcal{I}}, \Phi(y^{(2)})_{\mathcal{I}}, \cdots, \Phi(y^{(n)})_{\mathcal{I}}.$$

Since $|\mathcal{I}| = 2K_0$, from the definition of linear l-dependence, this is equivalent to $\Phi(y)$ is linearly $2K_0$-independent of $\Phi(y^{(1)}), \Phi(y^{(2)}), \cdots, \Phi(y^{(n)})$. **q.e.d.**

We now show that if Φ satisfies a weak technical condition, then $\text{rank}[\Phi, l] = l$. Specifically, for any $l \leq \min\{N, K\}$, we say Φ is l-full-rank if any submatrix of Φ with size $l \times l$ has full rank. Based on this notion, we have the following result:

Proposition 4 *For any $l \leq \min\{N, K\}$, if Φ is l-full-rank, then we have* $\text{rank}[\Phi, l] = l$.

Proof: Consider any sequence of matrix rows $\Phi^{(1)}, \Phi^{(2)}, \cdots, \Phi^{(l+1)}$ with length $l+1$, and any index set $\mathcal{I}$ with $|\mathcal{I}| = l$. Since Φ is l-full-rank, thus $\Phi^{(1)}_{\mathcal{I}}, \Phi^{(2)}_{\mathcal{I}}, \cdots, \Phi^{(l)}_{\mathcal{I}} \in \Re^l$ are linearly independent (hence forms a basis in $\Re^l$). Thus, $\Phi^{(l+1)}_{\mathcal{I}}$ is linearly dependent on $\Phi^{(1)}_{\mathcal{I}}, \Phi^{(2)}_{\mathcal{I}}, \cdots, \Phi^{(l)}_{\mathcal{I}} \in \Re^l$. Since this result holds for any $\mathcal{I}$ with $|\mathcal{I}| = l$, thus $\Phi^{(l+1)}$ is linearly l-dependent on $\Phi^{(1)}, \Phi^{(2)}, \cdots, \Phi^{(l)} \in \Re^K$. Furthermore, since this result holds for any sequence of matrix rows with length $l+1$, thus we have $\text{rank}[\Phi, l] \leq l$.

On the other hand, since Φ is l-full-rank, choose any sequence of matrix rows $\Phi^{(1)}, \Phi^{(2)}, \cdots, \Phi^{(l)}$ with length l and any index set $\mathcal{I}$ with $|\mathcal{I}| = l$, $\Phi^{(1)}_{\mathcal{I}}, \Phi^{(2)}_{\mathcal{I}}, \cdots, \Phi^{(l)}_{\mathcal{I}}$ are linearly independent. Thus, $\Phi^{(1)}, \Phi^{(2)}, \cdots, \Phi^{(l)}$ is a sequence of matrix rows s.t. every element is linearly l-independent of its predecessors. Thus, $\text{rank}[\Phi, l] \geq l$. So we have $\text{rank}[\Phi, l] = l$. **q.e.d.**

Thus, if $2K_0 \leq \min\{N, K\}$ and Φ is $2K_0$-full-rank, then we have $\dim_{\text{E}}[\mathcal{Q}_0] = \text{rank}\,[\Phi, 2K_0] = 2K_0$. Consequently, we have $\dim_{\text{E}}[\mathcal{Q}] = \dim_{\text{E}}[\mathcal{Q}_0^H] = 2K_0 H$.

A.2 Proof for Theorem 1

A.2.1 Proof for Lemma 1

Proof for Lemma 1: We prove this lemma by induction on j. First, notice that if $j = 0$, then from Algorithm 3, we have $\mathcal{Z}_0 = \varnothing$. Thus, Lemma 1(a) holds for $j = 0$.

Second, we prove that if Lemma 1(a) holds for episode j, then Lemma 1(b) holds for episode j and Lemma 1(a) holds for episode $j+1$. To see why Lemma 1(b) holds for episode j, notice that we have $Q^* \in \mathcal{Q}_{\mathcal{C}_j} \subseteq \mathcal{Q}$. Furthermore, from the induction hypothesis, $\forall z \in \mathcal{Z}_j$ and $\forall Q \in \mathcal{Q}_{\mathcal{C}_j}$, we have $Q(z) = Q^*(z)$. Since $(x_{j,t}, a_{j,t}, t)$ is dependent on $\mathcal{Z}_j$ with respect to $\mathcal{Q}$, then $\forall Q \in \mathcal{Q}_{\mathcal{C}_j} \subseteq \mathcal{Q}$, we have that $Q_t(x_{j,t}, a_{j,t}) = Q^*_t(x_{j,t}, a_{j,t})$. Hence we have $\sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x_{j,t}, a_{j,t}) = Q^*_t(x_{j,t}, a_{j,t})$, furthermore, from the OCP algorithm, we have $\sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x_{j,t}, a_{j,t}) \geq \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x_{j,t}, a)$, $\forall a \in \mathcal{A}$, thus we have

$$Q^*_t(x_{j,t}, a_{j,t}) = \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x_{j,t}, a_{j,t}) \geq \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_t(x_{j,t}, a) \geq Q^*_t(x_j, a), \quad \forall a \in \mathcal{A}, \tag{A.4}$$

where the last inequality follows from the fact that $Q^* \in \mathcal{Q}_{\mathcal{C}_j}$. Thus, $a_{j,t}$ is optimal and $Q^*_t(x_{j,t}, a_{j,t}) = V^*_t(x_{j,t})$. Thus, Lemma 1(b) holds for episode j.

We now prove Lemma 1(a) holds for episode $j+1$. We prove the conclusion by considering two different scenarios. If $t^*_j = \text{NULL}$, then $\mathcal{Z}_{j+1} = \mathcal{Z}_j$ and $\mathcal{Q}_{\mathcal{C}_{j+1}} \subseteq \mathcal{Q}_{\mathcal{C}_j}$. Thus, obviously, Lemma 1(a) holds for episode $j+1$. On the other hand, if $t^*_j \neq \text{NULL}$, we have $\mathcal{Q}_{\mathcal{C}_{j+1}} \subseteq \mathcal{Q}_{\mathcal{C}_j}$ and $\mathcal{Z}_{j+1} = \left[\mathcal{Z}_j, (x_{j,t^*_j}, a_{j,t^*_j}, t^*_j)\right]$. Based on the induction hypothesis, $\forall z \in \mathcal{Z}_j$ and $\forall Q \in \mathcal{Q}_{\mathcal{C}_{j+1}} \subseteq \mathcal{Q}_{\mathcal{C}_j}$, we have $Q(z) = Q^*(z)$. Thus, it is sufficient to prove that

$$Q_{t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = Q^*_{t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}), \quad \forall Q \in \mathcal{Q}_{\mathcal{C}_{j+1}}. \tag{A.5}$$

We prove Eqn(A.5) by considering two different cases. First, if $t^*_j = H-1$, it is sufficient to prove that

$$Q_{H-1}(x_{j,H-1}, a_{j,H-1}) = R_{H-1}(x_{j,H-1}, a_{j,H-1}) \quad \forall Q \in \mathcal{Q}_{\mathcal{C}_{j+1}}, \tag{A.6}$$

which holds by definition of $\mathcal{Q}_{\mathcal{C}_{j+1}}$ (see Algorithm 2, and recall that no constraints are conflicting if $Q^* \in \mathcal{Q}$). On the other hand, if $t^*_j < H-1$, it is sufficient to prove that for any $Q \in \mathcal{Q}_{\mathcal{C}_{j+1}}$, $Q_{t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = R_{t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) + V^*_{t^*_j+1}(x_{j,t^*_j+1})$. Recall that Algorithm 2 adds a constraint $L_{j,t^*_j} \leq Q_{t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) \leq U_{j,t^*_j}$ to $\mathcal{Q}_{\mathcal{C}_{j+1}}$ (and again, recall that no constraints are conflicting if $Q^* \in \mathcal{Q}$). Based on the definitions of L_{j,t^*_j}

and U_{j,t_j^*}, it is sufficient to prove that

$$V^*_{t_j^*+1}(x_{j,t_j^*+1}) = \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) = \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a). \tag{A.7}$$

We first prove that $V^*_{t_j^*+1}(x_{j,t_j^*+1}) = \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a)$. Specifically, we have that

$$\begin{aligned}
\sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) &= \sup_{a \in \mathcal{A}} \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) \\
&= \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}) \\
&= V^*_{t_j^*+1}(x_{j,t_j^*+1}),
\end{aligned} \tag{A.8}$$

where the second equality follows from the fact that

$$a_{j,t_j^*+1} \in \arg\max_{a \in \mathcal{A}} \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a)$$

and the last equality follows from the definition of t_j^* and Part (b) of the lemma for episode j (which we have just proved above, and holds by the induction hypothesis). Specifically, since t_j^* is the last period in episode j s.t. $(x_{j,t}, a_{j,t}, t)$ is independent of $\mathcal{Z}_j$ with respect to $\mathcal{Q}$. Thus, $(x_{j,t_j^*+1}, a_{j,t_j^*+1}, t_j^*+1)$ is dependent on $\mathcal{Z}_j$ with respect to $\mathcal{Q}$. From Lemma 1(b) for episode j, we have $V^*_{t_j^*+1}(x_{j,t_j^*+1}) = Q_{t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1})$ for any $Q \in \mathcal{Q}_{\mathcal{C}_j}$. Thus,

$$\sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}) = V^*_{t_j^*+1}(x_{j,t_j^*+1}) = \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}).$$

On the other hand, we have that

$$\begin{aligned}
\inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) &\geq \sup_{a \in \mathcal{A}} \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) \\
&\geq \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}) \\
&= V^*_{t_j^*+1}(x_{j,t_j^*+1}),
\end{aligned} \tag{A.9}$$

where the first inequality follows from the max-min inequality, the second inequality follows from the fact that $a_{j,t_j^*+1} \in \mathcal{A}$, and we have just proved the last equality above. Hence we have

$$\begin{aligned} V^*_{t_j^*+1}(x_{j,t_j^*+1}) &= \sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) \\ &\geq \inf_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{a \in \mathcal{A}} Q_{t_j^*+1}(x_{j,t_j^*+1}, a) \\ &\geq V^*_{t_j^*+1}(x_{j,t_j^*+1}). \end{aligned} \tag{A.10}$$

Thus, Eqn(A.7) holds. Hence, Lemma 1(a) holds for episode $j+1$, and by induction, we have proved Lemma 1. **q.e.d.**

A.2.2 Proof for Lemma 2

Proof for Lemma 2: Note that from Algorithm 3, if $t_j^* = \text{NULL}$, then for all $t = 0, 1, \cdots, H-1$, $(x_{j,t}, a_{j,t}, t)$ is dependent on $\mathcal{Z}_j$ with respect to $\mathcal{Q}$. Thus, from Lemma 1(b), $a_{j,t}$ is optimal for all $t = 0, 1, \cdots, H-1$. Hence we have

$$R^{(j)} = \sum_{t=0}^{H-1} R_t(x_{j,t}, a_{j,t}) = V_0^*(x_{j,0}).$$

On the other hand, $t_j^* \neq \text{NULL}$, then from Algorithm 3, $(x_{j,t_j^*}, a_{j,t_j^*}, t_j^*)$ is independent of $\mathcal{Z}_j$ and $|\mathcal{Z}_{j+1}| = |\mathcal{Z}_j| + 1$. Note $(x_{j,t_j^*}, a_{j,t_j^*}, t_j^*) \in \mathcal{Z}_{j+1}$, hence from Lemma 1(a), $\forall Q \in \mathcal{Q}_{\mathcal{C}_{j+1}}$, we have $Q_{t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) = Q^*_{t_j^*}(x_{j,t_j^*}, a_{j,t_j^*})$. **q.e.d.**

A.2.3 Proof for Theorem 1

Proof for Theorem 1: Notice that $\forall j = 0, 1, \cdots$, $R^{(j)} \leq V_0^*(x_{j,0})$ by definition. Thus, from Lemma 2, $R^{(j)} < V_0^*(x_{j,0})$ implies that $t_j^* \neq \text{NULL}$. Hence, for any $j = 0, 1, \cdots$, we have $\mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right] \leq \mathbf{1}\left[t_j^* \neq \text{NULL}\right]$. Furthermore, notice that from the definition of $\mathcal{Z}_j$, we have $\mathbf{1}\left[t_j^* \neq \text{NULL}\right] = |\mathcal{Z}_{j+1}| - |\mathcal{Z}_j|$, where $|\cdot|$ denotes the length of the given sequence. Thus for any $J = 0, 1, \cdots$, we have

$$\begin{aligned} \sum_{j=0}^{J} \mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right] &\leq \sum_{j=0}^{J} \mathbf{1}\left[t_j^* \neq \text{NULL}\right] \\ &= \sum_{j=0}^{J} \left[|\mathcal{Z}_{j+1}| - |\mathcal{Z}_j|\right] \\ &= |\mathcal{Z}_{J+1}| - |\mathcal{Z}_0| = |\mathcal{Z}_{J+1}|, \end{aligned} \tag{A.11}$$

where the last equality follows from the fact that $|\mathcal{Z}_0| = |\varnothing| = 0$. Notice that by definition (see Algorithm 3), $\forall j = 0, 1, \cdots$, $\mathcal{Z}_j$ is a sequence of elements in $\mathcal{Z}$ such that every element is independent of its predecessors with respect to $\mathcal{Q}$. Hence, from the definition of eluder dimension, we have $|\mathcal{Z}_j| \leq \dim_{\mathrm{E}}[\mathcal{Q}]$, $\forall j = 0, 1, \cdots$. Combining this result with Eqn(A.11), we have

$$\sum_{j=0}^{J} \mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right] \leq |\mathcal{Z}_{J+1}| \leq \dim_{\mathrm{E}}[\mathcal{Q}] \quad \forall J = 0, 1, \cdots .$$

Finally, notice that $\sum_{j=0}^{J} \mathbf{1}\left[V_j < V_0^*(x_{j,0})\right]$ is a non-decreasing function of J, and is bounded above by $\dim_{\mathrm{E}}[\mathcal{Q}]$. Thus,

$$\lim_{J\to\infty} \sum_{j=0}^{J} \mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right] = \sum_{j=0}^{\infty} \mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right]$$

exists, and satisfies $\sum_{j=0}^{\infty} \mathbf{1}\left[R^{(j)} < V_0^*(x_{j,0})\right] \leq \dim_{\mathrm{E}}[\mathcal{Q}]$. Hence we have

$$\left|\left\{j : R^{(j)} < V_0^*(x_{j,0})\right\}\right| \leq \dim_{\mathrm{E}}[\mathcal{Q}].$$

q.e.d.

A.3 Proof for Theorem 3 and Proposition 1

A.3.1 Proof for Lemma 4

Proof for Lemma 4: We prove Lemma 4 by induction on j. Note that when $j = 0$, $\forall (x, a, t)$, $Q^{\smiley}_{j,t}(x, a) = \infty$. Thus, Lemma 4 trivially holds for $j = 0$.

We now prove that if Lemma 4 holds for episode j, then it also holds for episode $j+1$, for any $j = 0, 1, \cdots$. To prove this result, it is sufficient to show that for any (x, a, t) whose associated optimistic Q-value has been updated in episode j (i.e. $Q^{\smiley}_{j,t}(x, a) \neq Q^{\smiley}_{j+1,t}(x, a)$), if the new optimistic Q-value $Q^{\smiley}_{j+1,t}(x, a)$ is still finite, then we have

$$|Q^{\smiley}_{j+1,t}(x, a) - Q^{*}_{t}(x, a)| \leq 2\rho(H - t).$$

Note that if $Q^{\smiley}_{j,t}(x, a) \neq Q^{\smiley}_{j+1,t}(x, a)$, then (x, a, t) must be in the same partition $\mathcal{Z}_{t,k}$ as $(x_{j,t}, a_{j,t}, t)$. Noting that

$$\sup_{Q \in \mathcal{Q}_{\mathcal{C}_j}} \sup_{b \in \mathcal{A}} Q_{t+1}(x_{j,t+1}, b) = \sup_{b \in \mathcal{A}} Q^{\smiley}_{j,t+1}(x_{j,t+1}, b),$$

from the discussion in Subsection 2.5.3, we have

$$Q^{\smiley}_{j+1,t}(x, a) = \overline{\theta}^{(j+1)}_{t,k} = \begin{cases} R_{H-1}(x_{j,H-1}, a_{j,H-1}) & \text{if } t = H - 1 \\ R_t(x_{j,t}, a_{j,t}) + \sup_{b \in \mathcal{A}} Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) & \text{if } t < H - 1 \end{cases} \tag{A.12}$$

We now prove $|Q^{\smiley}_{j+1,t}(x, a) - Q^{*}_{t}(x, a)| \leq 2\rho(H - t)$ by considering two different scenarios. First, if $t = H - 1$, then

$$Q^{\smiley}_{j+1,t}(x, a) = R_{H-1}(x_{j,H-1}, a_{j,H-1}) = Q^{*}_{H-1}(x_{j,H-1}, a_{j,H-1}).$$

From our discussion in Section 2.5, we have $|Q^{*}_{t}(x, a) - Q^{*}_{H-1}(x_{j,H-1}, a_{j,H-1})| \leq 2\rho$, which implies that $|Q^{*}_{t}(x, a) - Q^{\smiley}_{j+1,t}(x, a)| \leq 2\rho = 2\rho(H - t)$. On the other hand, if

$t < H - 1$, then

$$Q^{\smiley}_{j+1,t}(x,a) = R_t(x_{j,t}, a_{j,t}) + \sup_{b \in \mathcal{A}} Q^{\smiley}_{j,t+1}(x_{j,t+1}, b).$$

If $Q^{\smiley}_{j+1,t}(x,a) < \infty$, then $Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) < \infty$, $\forall b \in \mathcal{A}$. Furthermore, from the induction hypothesis, $Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) < \infty$, $\forall b \in \mathcal{A}$, implies that $\forall b \in \mathcal{A}$,

$$\left| Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) - Q^*_{t+1}(x_{j,t+1}, b) \right| \leq 2\rho(H - t - 1).$$

On the other hand, from the Bellman equation at $(x_{j,t}, a_{j,t}, t)$, we have that $Q^*_t(x_{j,t}, a_{j,t}) = R_t(x_{j,t}, a_{j,t}) + \sup_{b \in \mathcal{A}} Q^*_{t+1}(x_{j,t+1}, b)$. Thus,

$$\begin{aligned} \left| Q^{\smiley}_{j+1,t}(x,a) - Q^*_t(x_{j,t}, a_{j,t}) \right| &= \left| \sup_{b \in \mathcal{A}} Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) - \sup_{b \in \mathcal{A}} Q^*_{t+1}(x_{j,t+1}, b) \right| \\ &\leq \sup_{b \in \mathcal{A}} \left| Q^{\smiley}_{j,t+1}(x_{j,t+1}, b) - Q^*_{t+1}(x_{j,t+1}, b) \right| \\ &\leq 2\rho(H - t - 1). \end{aligned} \tag{A.13}$$

Moreover, since (x, a, t) and $(x_{j,t}, a_{j,t}, t)$ are in the same partition, we have

$$|Q^*_t(x,a) - Q^*_t(x_{j,t}, a_{j,t})| \leq 2\rho,$$

consequently, we have $\left| Q^{\smiley}_{j+1,t}(x,a) - Q^*_t(x,a) \right| \leq 2\rho(H - t)$. Thus, Lemma 4 holds for episode $j + 1$. By induction, we have proved Lemma 4. **q.e.d.**

A.3.2 Proof for Lemma 5

Proof for Lemma 5: Notice that from Algorithm 2, for all $t = 0, 1, \cdots, H - 1$, we have

$$Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) \geq Q^{\smiley}_{j,t}(x_{j,t}, a) \quad \forall a \in \mathcal{A}.$$

Thus, if $Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) < \infty$ for any t, then $Q^{\smiley}_{j,t}(x_{j,t}, a) < \infty$, $\forall (a,t)$. Consequently, from Lemma 4, we have that

$$\left| Q^*_t(x_{j,t}, a) - Q^{\smiley}_{j,t}(x_{j,t}, a) \right| \leq 2\rho(H-t) \quad \forall (a,t).$$

Thus, for any t, we have

$$\begin{aligned} Q^*_t(x_{j,t}, a_{j,t}) + 2\rho(H-t) &\geq Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) \\ &\geq Q^{\smiley}_{j,t}(x_{j,t}, a) \\ &\geq Q^*_t(x_{j,t}, a) - 2\rho(H-t), \quad \forall a \in \mathcal{A}, \end{aligned} \tag{A.14}$$

which implies that

$$Q^*_t(x_{j,t}, a_{j,t}) \geq \sup_{a \in \mathcal{A}} Q^*_t(x_{j,t}, a) - 4\rho(H-t) = V^*_t(x_{j,t}) - 4\rho(H-t) \quad \forall t.$$

We first prove that $V^*_0(x_{j,0}) - R^{(j)} \leq 2\rho H(H+1)$. Note that combining the above inequality with Bellman equation, we have that

$$R_t(x_{j,t}, a_{j,t}) \geq V^*_t(x_{j,t}) - V^*_{t+1}(x_{j,t+1}) - 4\rho(H-t) \quad \forall t < H-1$$

and $R_{H-1}(x_{j,H-1}, a_{j,H-1}) \geq V^*_{H-1}(x_{j,H-1}) - 4\rho$. Summing up these inequalities, we have $V^*_0(x_{j,0}) - R^{(j)} \leq 2\rho H(H+1)$.

Second, we prove that $V^*_0(x_{j,0}) - R^{(j)} \leq 6\rho H$ if the conditions of Proposition 1 hold. Note that the conditions of Proposition 1 imply that

$$U_{j,t} \geq Q^{\smiley}_{j,t}(x_{j,t}, a_{j,t}) \geq Q^{\frownie}_{j,t}(x_{j,t}, a_{j,t}) \geq L_{j,t}, \quad \forall t.$$

Note that by definition, $U_{j,H-1} = L_{j,H-1} = R_{H-1}(x_{j,H-1}, a_{j,H-1})$, and for $t < H-1$, we have $U_{j,t} = R_t(x_{j,t}, a_{j,t}) + Q^{\smiley}_{j,t+1}(x_{j,t+1}, a_{j,t+1})$, and

$$L_{j,t} \geq R_t(x_{j,t}, a_{j,t}) + \sup_{a \in \mathcal{A}} Q^{\frownie}_{j,t+1}(x_{j,t+1}, a) \geq R_t(x_{j,t}, a_{j,t}) + Q^{\frownie}_{j,t+1}(x_{j,t+1}, a_{j,t+1}),$$

where the first inequality follows from the definition of $L_{j,t}$ and max-min inequality, and the second inequality follows from the fact that $a_{j,t+1} \in \mathcal{A}$. Combining the above inequalities, we have

$$\begin{aligned} Q^{\frownie}_{j,0}(x_{j,0}, a_{j,0}) &\geq \sum_{t=0}^{H-1} R_t(x_{j,t}, a_{j,t}) = R^{(j)} \\ &\geq Q^{\smiley}_{j,0}(x_{j,0}, a_{j,0}) \geq Q^{\frownie}_{j,0}(x_{j,0}, a_{j,0}). \end{aligned} \tag{A.15}$$

Thus we have $Q^{\smiley}_{j,0}(x_{j,0}, a_{j,0}) = Q^{\frownie}_{j,0}(x_{j,0}, a_{j,0}) = R^{(j)} < \infty$. So from Lemma 4,

$$\left| R^{(j)} - Q^*_0(x_{j,0}, a_{j,0}) \right| = \left| Q^{\smiley}_{j,0}(x_{j,0}, a_{j,0}) - Q^*_0(x_{j,0}, a_{j,0}) \right| \leq 2\rho H.$$

Thus, $R^{(j)} \geq Q^*_0(x_{j,0}, a_{j,0}) - 2\rho H$. Furthermore, from the above analysis,

$$Q^*_0(x_{j,0}, a_{j,0}) \geq V^*_0(x_{j,0}) - 4\rho H.$$

Thus we have $R^{(j)} \geq V^*_0(x_{j,0}) - 6\rho H$. **q.e.d.**

A.3.3 Proof for Lemma 6

Proof for Lemma 6: $\forall j = 0, 1, \cdots$, if $t^*_j = \text{NULL}$, then by definition of t^*_j and Lemma 5, we have

$$V^*_0(x_{j,0}) - R^{(j)} \leq 2\rho H(H+1).$$

On the other hand, if $t^*_j \neq \text{NULL}$, then by definition of t^*_j, $Q^{\smiley}_{j,t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = \infty$. We now show that $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) < \infty$ for all $j' > j$, and $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = \infty$ for all $j' \leq j$.

Assume that $(x_{j,t^*_j}, a_{j,t^*_j}, t^*_j)$ belongs to partition $\mathcal{Z}_{t^*_j,k}$, thus $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = \overline{\theta}^{(j')}_{t^*_j,k}$, $\forall j'$. Based on our discussion above, $\overline{\theta}^{(j')}_{t^*_j,k}$ is monotonically non-increasing in j'. Thus, $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j})$ is monotonically non-increasing in j', and hence for any $j' \leq j$, we have $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) = \infty$. Furthermore, to prove that $Q^{\smiley}_{j',t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) < \infty$ for all $j' > j$, it is sufficient to prove that $Q^{\smiley}_{j+1,t^*_j}(x_{j,t^*_j}, a_{j,t^*_j}) < \infty$.

From Algorithm 2, the OCP algorithm will add a new constraint

$$L_{j,t_j^*} \leq Q_{t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) \leq U_{j,t_j^*}.$$

We first prove that $U_{j,t_j^*} < \infty$. To see it, notice that if $t_j^* = H-1$, then

$$U_{j,t_j^*} = U_{j,H-1} = R_{H-1}(x_{j,H-1}, a_{j,H-1}) < \infty.$$

On the other hand, if $t_j^* < H-1$, then by definition

$$U_{j,t_j^*} = R_{t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) + Q^{\smiley}_{j,t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}).$$

From the definition of t_j^*, $Q^{\smiley}_{j,t_j^*+1}(x_{j,t_j^*+1}, a_{j,t_j^*+1}) < \infty$, thus $U_{j,t_j^*} < \infty$. Consequently,

$$Q^{\smiley}_{j+1,t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) = \overline{\theta}^{(j+1)}_{t_j^*,k} = \min\{\overline{\theta}^{(j)}_{t_j^*,k}, U_{j,t_j^*}\} \leq U_{j,t_j^*} < \infty.$$

Thus, $Q^{\smiley}_{j',t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) < \infty$ for all $j' > j$.

Thus, if we consider $Q^{\smiley}_{j',t_j^*}(x_{j,t_j^*}, a_{j,t_j^*}) = \overline{\theta}^{(j')}_{t_j^*,k}$ as a function of j', then this function transits from infinity to finite values in episode j. In summary, $t_j^* \neq$ NULL implies that $\overline{\theta}^{(j')}_{t_j^*,k}$ transits from infinity to finite values in episode j. Since other $\overline{\theta}^{(j')}_{t,k}$'s might also transit from ∞ to finite values in episode j, thus $\mathbf{1}[t_j^* \neq \text{NULL}]$ is less than or equal to the number of $\overline{\theta}^{(j')}_{t,k}$'s transiting from ∞ to finite values in episode j. Note that from the monotonicity of $\overline{\theta}^{(j')}_{t,k}$, for each partition, this transition can occur at most once, and there are K partitions in total. Hence we have

$$\sum_{j=0}^{\infty} \mathbf{1}[t_j^* \neq \text{NULL}] \leq K.$$

q.e.d.

A.4 Proofs for Lemma 7 and Theorem 4

A.4.1 Proof for Lemma 7

Proof for Lemma 7: Since $\tilde{Q} \in \mathcal{Q}$, then by definition of k^*, we have

$$\mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] \geq \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{\tilde{Q}}}(x_0)\right].$$

Thus, it is sufficient to prove that

$$\mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{\tilde{Q}}}(x_0)\right] \geq \mathbb{E}_{x_0 \sim \pi}\left[V_0^{*}(x_0)\right] - 2\rho H.$$

In this lemma, we prove a stronger result by showing that

$$V_h^{\mu_{\tilde{Q}}}(x) \geq V_h^*(x) - 2\rho(H-h) \qquad \forall x \in \mathcal{S},\ \forall h = H-1, H-2, \cdots, 0. \tag{A.16}$$

We prove Eqn(A.16) by backward induction. First, for $h = H-1$ and any $x \in \mathcal{S}$, we have that

$$\begin{aligned}
V_{H-1}^{\mu_{\tilde{Q}}}(x) &\overset{\text{(a)}}{=} R\left(x, \mu_{\tilde{Q},H-1}(x)\right) \\
&\overset{\text{(b)}}{=} Q_{H-1}^*\left(x, \mu_{\tilde{Q},H-1}(x)\right) \\
&\overset{\text{(c)}}{\geq} \tilde{Q}_{H-1}\left(x, \mu_{\tilde{Q},H-1}(x)\right) - \rho \\
&\overset{\text{(d)}}{\geq} \tilde{Q}_{H-1}\left(x, \mu_{H-1}^*(x)\right) - \rho \\
&\overset{\text{(e)}}{\geq} Q_{H-1}^*\left(x, \mu_{H-1}^*(x)\right) - 2\rho = V_{H-1}^*(x) - 2\rho,
\end{aligned} \tag{A.17}$$

where μ^* is an optimal policy, equality (a) and (b) follow from the assumption that there is no reward noise, inequality (c) and (e) follow from the definition of ρ (recall that $\rho = \|Q^* - \tilde{Q}\|_\infty$), and inequality (d) follows from the fact that $\mu_{\tilde{Q}}$ is greedy to $\tilde{Q}$.

Now assume that Eqn(A.16) holds for period $h+1$, then we prove that it also holds for period h. Notice that for any $x \in \mathcal{S}$, we have

$$\begin{aligned}
V_h^{\mu_{\tilde{Q}}}(x) &\overset{\text{(a)}}{=} R(x, \mu_{\tilde{Q},h}(x)) + \mathbb{E}_{y \sim P(\cdot|x, \mu_{\tilde{Q},h}(x))}\left[V_{h+1}^{\mu_{\tilde{Q}}}(y)\right] \\
&\overset{\text{(b)}}{\geq} R(x, \mu_{\tilde{Q},h}(x)) + \mathbb{E}_{y \sim P(\cdot|x, \mu_{\tilde{Q},h}(x))}\left[V_{h+1}^{*}(y)\right] - 2\rho(H-h-1) \\
&\overset{\text{(c)}}{=} Q_h^*(x, \mu_{\tilde{Q},h}(x)) - 2\rho(H-h) + 2\rho \\
&\overset{\text{(d)}}{\geq} \tilde{Q}_h(x, \mu_{\tilde{Q},h}(x)) - 2\rho(H-h) + \rho \\
&\overset{\text{(e)}}{\geq} \tilde{Q}_h(x, \mu_h^*(x)) - 2\rho(H-h) + \rho \\
&\overset{\text{(f)}}{\geq} Q_h^*(x, \mu_h^*(x)) - 2\rho(H-h) = V_h^*(x) - 2\rho(H-h), \qquad \text{(A.18)}
\end{aligned}$$

where equality (a) follows from the Bellman equation under policy $\mu_{\tilde{Q}}$, inequality (b) follows from the induction hypothesis, equality (c) follows form the definition of Q^*, inequality (d) and (f) follow from the definition of ρ (i.e. $\rho = \|Q^* - \tilde{Q}\|_\infty$), and inequality (e) follows from the fact that $\mu_{\tilde{Q}}$ is greedy to $\tilde{Q}$.

Hence, we have proved Eqn(A.16). **q.e.d.**

A.4.2 Proof for Theorem 4

Before proceeding, we first define some useful notations. To distinguish l_k's and $\tilde{V}_k$'s in different episodes, we use $l_{k,i}$ and $\tilde{V}_{k,l_{k,i}}$ to denote l_k and $\tilde{V}_k$ *at the beginning* of episode i, for any $k = 1, 2, \cdots, K$ and any $i \geq K$. To simplify the notations, we use $c_{i,l}$ as a shorthand notation for the confidence radius and define it as

$$c_{i,l} = 2\overline{R}H\sqrt{\frac{2\log(i)}{l}} \qquad \forall i \geq K,\ \forall l = 1, 2, \cdots. \qquad \text{(A.19)}$$

We also define

$$\Delta_k = \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right],$$

which quantifies the "performance loss" of policy μ_k with respect to μ_{k^*}.

For any sub-optimal policy μ_k with $\Delta_k > 0$, the following lemma bounds the expected number of times it is applied in the first j episodes.

Lemma 9 *For any k s.t. $\Delta_k > 0$, and any $j > K$, we have*

$$\mathbb{E}\left[\sum_{i=0}^{j-1} \mathbf{1}\left\{\textit{policy } \mu_k \textit{ is applied in episode } i\right\}\right] \leq \frac{32\log(j)(\overline{R}H)^2}{\Delta_k^2} + 1 + \frac{\pi^2}{3}.$$

Proof: Our proof follows the analysis in [2]. Notice that for any positive integer L_k, we have

$$\begin{aligned}
&\sum_{i=0}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i\right\} \\
\overset{\text{(a)}}{=} \quad & 1 + \sum_{i=K}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i,\, l_{k,i} \leq L_k\right\} \\
+ \quad & \sum_{i=K}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i,\, l_{k,i} > L_k\right\} \\
\overset{\text{(b)}}{\leq} \quad & 1 + L_k + \sum_{i=K}^{j-1} \mathbf{1}\left\{\tilde{V}_{k,l_{k,i}} + c_{i,l_{k,i}} \geq \tilde{V}_{k^*,l_{k^*,i}} + c_{i,l_{k^*,i}},\, l_{k,i} > L_k\right\},
\end{aligned}$$

where equality (a) follows from the fact that policy μ_k is applied once in the first K episodes, and can be decomposed into two cases in the subsequent episodes; inequality (b) follows from the fact that

$$\sum_{i=K}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i,\, l_{k,i} \leq L_k\right\} \leq L_k,$$

(see the definition of $l_{k,i}$), and $\tilde{V}_{k,l_{k,i}} + c_{i,l_{k,i}} \geq \tilde{V}_{k^*,l_{k^*,i}} + c_{i,l_{k^*,i}}$ is a necessary (but not sufficient) condition of choosing μ_k in episode i. Furthermore, $l_{k,i} > L_k$ implies that

$i \geq K + L_k$ [1], hence we have

$$\begin{aligned}
&\sum_{i=0}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i\right\} \\
\leq\ & 1 + L_k + \sum_{i=K+L_k}^{j-1} \mathbf{1}\left\{\tilde{V}_{k,l_{k,i}} + c_{i,l_{k,i}} \geq \tilde{V}_{k^*,l_{k^*,i}} + c_{i,l_{k^*,i}},\, l_{k,i} > L_k\right\} \\
\overset{\text{(c)}}{\leq}\ & 1 + L_k + \sum_{i=K+L_k}^{j-1} \sum_{s=1}^{i+1-K} \sum_{s_k=L_k+1}^{i+1-K} \mathbf{1}\left\{\tilde{V}_{k,s_k} + c_{i,s_k} \geq \tilde{V}_{k^*,s} + c_{i,s}\right\}, \qquad \text{(A.20)}
\end{aligned}$$

where inequality (c) is the key inequality in the proof of this lemma. Specifically, at the beginning of episode i, we have $1 \leq l_{k^*,i} \leq i - (K-1) = i + 1 - K$, and conditioning on $l_{k,i} > L_k$, we also have $L_k + 1 \leq l_{k,i} \leq i + 1 - K$. Hence we have

$$\begin{aligned}
&\mathbf{1}\left\{\tilde{V}_{k,l_{k,i}} + c_{i,l_{k,i}} \geq \tilde{V}_{k^*,l_{k^*,i}} + c_{i,l_{k^*,i}},\, l_{k,i} > L_k\right\} \\
&\quad \leq \sum_{s=1}^{i+1-K} \sum_{s_k=L_k+1}^{i+1-K} \mathbf{1}\left\{\tilde{V}_{k,s_k} + c_{i,s_k} \geq \tilde{V}_{k^*,s} + c_{i,s}\right\}.
\end{aligned}$$

Notice that the bound (A.20) is independent of the random variables $l_{k,i}$'s. When $\tilde{V}_{k,s_k} + c_{i,s_k} \geq \tilde{V}_{k^*,s} + c_{i,s}$, at least one of the following three events must happen:

1. $\tilde{V}_{k,s_k} \geq \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right] + c_{i,s_k}$;

2. $\tilde{V}_{k^*,s} \leq \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - c_{i,s}$;

3. $\mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] < \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right] + 2c_{i,s_k}$.

Notice that if none of the above three events happens, then we have

$$\tilde{V}_{k^*,s} + c_{i,s} > \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_{k^*}}(x_0)\right] \geq \mathbb{E}_{x_0 \sim \pi}\left[V_0^{\mu_k}(x_0)\right] + 2c_{i,s_k} > \tilde{V}_{k,s_k} + c_{i,s_k},$$

which contradicts $\tilde{V}_{k,s_k} + c_{i,s_k} \geq \tilde{V}_{k^*,s} + c_{i,s}$.

[1] To see it, notice that $l_{k,i} > L_k$ implies that policy μ_k has been applied at least $L_k + 1$ times in episode $0, 1, \cdots, i-1$, i.e. μ_k has been applied at least L_k times in episode $K, K+1, \cdots, i-1$. This requires $(i-1) - K + 1 = i - K \geq L_k$, i.e $i \geq K + L_k$.

Using Hoeffding's inequality, for any i, s_k and s, we have

$$\begin{aligned}
\mathbb{P}\left[\tilde{V}_{k,s_k} \geq \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_k}(x_0)\right] + c_{i,s_k}\right] &\leq \exp\left(-\frac{2s_k^2 c_{i,s_k}^2}{s_k(2\overline{R}H)^2}\right) = \exp\left(-4\log(i)\right) = i^{-4} \\
\mathbb{P}\left[\tilde{V}_{k^*,s} \leq \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - c_{i,s}\right] &\leq \exp\left(-\frac{2s^2 c_{i,s}^2}{s(2\overline{R}H)^2}\right) = \exp\left(-4\log(i)\right) = i^{-4}
\end{aligned}$$

When we choose

$$L_k = \left\lceil \frac{32\log(j)(\overline{R}H)^2}{\varDelta_k^2} \right\rceil - 1, \tag{A.21}$$

then for any $s_k \geq L_k + 1$ and any $i \leq j-1$, the third event cannot happen. To see it, notice that $\log(j) > 0$ (since $j > K \geq 1$), and hence for any $s_k \geq L_k + 1$ and any $K + L_k \leq i \leq j-1$, we have $\log(i) > 0$ [2] and

$$\begin{aligned}
\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - \left\{\mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_k}(x_0)\right] + 2c_{i,s_k}\right\} &= \varDelta_k - 2c_{i,s_k} = \varDelta_k - 4\overline{R}H\sqrt{\frac{2\log(i)}{s_k}} \\
&\geq \varDelta_k - 4\overline{R}H\sqrt{\frac{2\log(i)}{L_k+1}} \\
&\geq \varDelta_k\left[1 - \sqrt{\frac{\log(i)}{\log(j)}}\right] \geq 0,
\end{aligned}$$

where the first inequality follows from $s_k \geq L_k + 1$ and the last inequality follows from $\log(i) < \log(j)$. Therefore, by choosing L_k as given in Eqn(A.21), we have

$$\begin{aligned}
&\mathbb{E}\left\{\mathbf{1}\left[\tilde{V}_{k,s_k} + c_{i,s_k} \geq \tilde{V}_{k^*,s} + c_{i,s}\right]\right\} \\
\leq\ &\mathbb{P}\left[\tilde{V}_{k,s_k} \geq V_0^{\mu_k}(x_0) + c_{i,s_k}\right] + \mathbb{P}\left[\tilde{V}_{k^*,s} \leq V_0^{\mu_{k^*}}(x_0) - c_{i,s}\right] \leq \frac{2}{i^4},
\end{aligned}$$

for any $s_k \geq L_k + 1$ and any $i \leq j-1$, where the first inequality follows from the

[2] To see why $\log(i) > 0$, notice that $\varDelta_k \leq 2\overline{R}H$ and $j \geq 2$, hence we have $L_k \geq 5$, thus $i \geq K + L_k \geq 6$.

union bound. Combining these results with Eqn(A.20), we have

$$
\begin{aligned}
&\mathbb{E}\left[\sum_{i=0}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i\right\}\right] \\
\leq\ & \left\lceil \frac{32\log(j)(\overline{R}H)^2}{\mathit{\Delta}_k^2} \right\rceil + \sum_{i=K+L_k}^{j-1} \sum_{s=1}^{i+1-K} \sum_{s_k=L_k+1}^{i+1-K} \frac{2}{i^4} \\
\overset{\text{(e)}}{\leq}\ & \frac{32\log(j)(\overline{R}H)^2}{\mathit{\Delta}_k^2} + 1 + \sum_{i=1}^{\infty} \sum_{s=1}^{i} \sum_{s_k=1}^{i} \frac{2}{i^4} \\
=\ & \frac{32\log(j)(\overline{R}H)^2}{\mathit{\Delta}_k^2} + 1 + \sum_{i=1}^{\infty} \frac{2}{i^2} = \frac{32\log(j)(\overline{R}H)^2}{\mathit{\Delta}_k^2} + 1 + \frac{\pi^2}{3},
\end{aligned}
$$

where the inequality (e) follows from the fact $K = |\mathcal{Q}| \geq 1$. **q.e.d.**

To simplify the exposition, we define

$$
Z_{j,k} = \mathbb{E}\left[\sum_{i=0}^{j-1} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i\right\}\right]. \tag{A.22}
$$

Thus, Lemma 9 provides a gap-dependent bound on $Z_{j,k}$. We now prove Theorem 4.

Proof for Theorem 4: Notice that the expected cumulative regret with respect to $V_0^{\mu_{k^*}}(x_0)$ in the first j episodes is

$$
\begin{aligned}
&\sum_{i=0}^{j-1} \mathbb{E}\left[V_0^{\mu_{k^*}}(x_0) - R^{(i)}\right] \\
=&\sum_{i=0}^{j-1} \left\{ \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - \sum_{k=1}^{K} \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_k}(x_0)\right] \mathbb{P}\left\{\mu_k \text{ is applied in episode } i\right\} \right\} \\
=&\sum_{i=0}^{j-1} \sum_{k=1}^{K} \mathbb{P}\left\{\mu_k \text{ is applied in episode } i\right\} \mathit{\Delta}_k \\
=&\sum_{k=1}^{K} \mathit{\Delta}_k \left[\sum_{i=0}^{j-1} \mathbb{P}\left\{\mu_k \text{ is applied in episode } i\right\}\right],
\end{aligned} \tag{A.23}
$$

which is $\sum_{k=1}^{K} \Delta_k Z_{j,k}$, based on the definition of $Z_{j,k}$. From Lemma 9, for any k s.t. $\Delta_k > 0$ and any $j > K$, we have

$$\Delta_k Z_{j,k} \leq \frac{32\log(j)(\overline{R}H)^2}{\Delta_k} + \left(1 + \frac{\pi^2}{3}\right)\Delta_k. \tag{A.24}$$

If we multiply both sides by $\Delta_k Z_{j,k}$, then we have

$$\begin{aligned}
(\Delta_k Z_{j,k})^2 &\leq 32\log(j)(\overline{R}H)^2 Z_{j,k} + \left(1 + \frac{\pi^2}{3}\right)\Delta_k^2 Z_{j,k} \\
&\overset{\text{(a)}}{\leq} 32\log(j)(\overline{R}H)^2 Z_{j,k} + \left(1 + \frac{\pi^2}{3}\right)(2\overline{R}H)^2 Z_{j,k} \\
&= (\overline{R}H)^2\left[32\log(j) + (4 + \frac{4}{3}\pi^2)\right] Z_{j,k} \\
&\overset{\text{(b)}}{\leq} 64(\overline{R}H)^2 \log(j) Z_{j,k},
\end{aligned}$$

where the inequality (a) follows from the fact that

$$\Delta_k = \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right] - \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_k}(x_0)\right] \leq 2\overline{R}H;$$

the inequality (b) follows from the fact that $j > K \geq 1$, and from algebra, for $j \geq 2$, $32\log(j) \geq 32\log(2) > (4 + \frac{4}{3}\pi^2)$. Hence we have

$$\Delta_k Z_{j,k} \leq 8\overline{R}H\sqrt{\log(j) Z_{j,k}}, \tag{A.25}$$

for any k s.t. $\Delta_k > 0$ and any $j > K$. Obviously, the above inequality trivially holds when $\mathit{\Delta}_k = 0$, thus we have

$$\begin{aligned}
\sum_{i=0}^{j-1} \mathbb{E}\left[V_0^{\mu_{k^*}}(x_0) - R^{(i)}\right] &= \sum_{k=1}^{K} \mathit{\Delta}_k Z_{j,k} \leq \sum_{k=1}^{K} 8\overline{R}H\sqrt{\log(j)}\sqrt{Z_{j,k}} \\
&= 8\overline{R}H\sqrt{\log(j)}\sum_{k=1}^{K}\sqrt{Z_{j,k}} \\
&\overset{\text{(c)}}{\leq} 8\overline{R}H\sqrt{\log(j)K}\sqrt{\sum_{k=1}^{K} Z_{j,k}} \\
&\overset{\text{(d)}}{=} 8\overline{R}H\sqrt{\log(j)jK}, \qquad \text{(A.26)}
\end{aligned}$$

where inequality (c) follows from Cauchy-Schwarz inequality, and equality (d) follows from the fact that

$$\sum_{k=1}^{K} Z_{j,k} = \mathbb{E}\left[\sum_{i=0}^{j-1}\sum_{k=1}^{K} \mathbf{1}\left\{\text{policy } \mu_k \text{ is applied in episode } i\right\}\right] = \mathbb{E}\left[\sum_{i=0}^{j-1} 1\right] = j.$$

Finally, we have that

$$\begin{aligned}
&\text{Regret}(j) \\
&= \sum_{i=0}^{j-1} \mathbb{E}\left[V_0^*(x_0) - R^{(i)}\right] \\
&= \left\{\mathbb{E}_{x_0\sim\pi}\left[V_0^*(x_0)\right] - \mathbb{E}_{x_0\sim\pi}\left[V_0^{\mu_{k^*}}(x_0)\right]\right\} j + \sum_{i=0}^{j-1} \mathbb{E}\left[V_0^{\mu_{k^*}}(x_0) - R^{(i)}\right] \\
&\leq 2\rho H j + 8\overline{R}H\sqrt{Kj\log(j)}, \qquad \text{(A.27)}
\end{aligned}$$

where the last inequality follows from Lemma 7 and the above result. **q.e.d.**

Bibliography

[1] Yasin Abbasi-Yadkori and Csaba Szepesvári. Regret bounds for the adaptive control of linear quadratic systems. *Journal of Machine Learning Research - Proceedings Track*, 19:1–26, 2011.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.

[3] Peter Auer and Ronald Ortner. Logarithmic online regret bounds for undiscounted reinforcement learning. In *NIPS*, pages 49–56, 2006.

[4] Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Regret bounds for reinforcement learning with policy advice. *CoRR*, abs/1305.1027, 2013.

[5] K. Aziz and A. Settari. *Petroleum Reservoir Simulation.* Applied Science Publishers, 1979.

[6] G. Barbose, C. Goldman, and B. Neenan. A survey of utility experience with real time pricing. *Lawrence Berkeley National Laboratory: Lawrence Berkeley National Laboratory*, 2004.

[7] Peter L. Bartlett and Ambuj Tewari. REGAL: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI2009)*, pages 35–42, June 2009.

[8] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 3 edition, 2005.

[9] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 4 edition, 2012.

[10] Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, September 1996.

[11] S. Borenstein, M. Jaske, and A. Rosenfeld. Dynamic pricing, advanced metering, and demand response in electricity markets. *UC Berkeley: Center for the Study of Energy Markets*, October 2002.

[12] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.

[13] S. Braithwait and K. Eakin. The role of demand response in electric power market design. *Edison Electric Institute*, 2002.

[14] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[15] Adam Coates, Pieter Abbeel, and Andrew Y. Ng. Autonomous helicopter flight using reinforcement learning. In *Encyclopedia of Machine Learning*, pages 53–61. Springer, 2010.

[16] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[17] Daniela Pucci de Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, 29(3):462–478, 2004.

[18] Richard Dearden, Nir Friedman, and Stuart J. Russell. Bayesian q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[19] D. Echeverría Ciaurri, O.J. Isebor, and L.J. Durlofsky. Application of derivative-free methodologies for generally constrained oil production optimization problems. *International Journal of Mathematical Modelling and Numerical Optimization*, 2:134–161, 2011.

[20] Geoffrey Gordon. Online fitted reinforcement learning. In *Advances in Neural Information Processing Systems 8*, pages 1052–1058. MIT Press, 1995.

[21] Morteza Ibrahimi, Adel Javanmard, and Benjamin Van Roy. Efficient reinforcement learning for high dimensional linear quadratic systems. In *NIPS*, 2012.

[22] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.

[23] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems An Introduction*. Cambridge University Press, 2011.

[24] Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.

[25] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[26] Michael J. Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. In *IJCAI*, pages 740–747, 1999.

[27] Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

[28] T.L Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4 – 22, 1985.

[29] Tor Lattimore, Marcus Hutter, and Peter Sunehag. The sample-complexity of general reinforcement learning. In *ICML*, 2013.

[30] Lihong Li and Michael Littman. Reducing reinforcement learning to kwik online regression. *Annals of Mathematics and Artificial Intelligence*, 2010.

[31] Lihong Li, Michael L. Littman, and Thomas J. Walsh. Knows what it knows: a framework for self-aware learning. In *ICML*, pages 568–575, 2008.

[32] Yuxi Li, Csaba Szepesvári, and Dale Schuurmans. Learning exercise policies for american options. In *AISTATS*, pages 352–359, 2009.

[33] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *ICML*, pages 673–680, 2006.

[34] D. O'Neill, M. Levorato, A. J. Goldsmith, and U. Mitra. Residential demand response using reinforcement learning. In *IEEE SmartGridComm*, Gaithersburg, Maryland, USA, OCT 2010.

[35] Ronald Ortner and Daniil Ryabko. Online regret bounds for undiscounted continuous reinforcement learning. In *NIPS*, 2012.

[36] Warren Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley-Interscience, 2007.

[37] Warren Powell and Ilya Ryzhov. *Optimal Learning.* John Wiley and Sons, 2011.

[38] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical report, 1994.

[39] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *CoRR*, abs/1301.2609, 2013.

[40] P. Sarma, L. J. Durlofsky, K. Aziz, and W. H. Chen. Efficient real-time reservoir management using adjoint-based optimal control and model updating. *Computational Geosciences*, 10:3–36, 2006.

[41] Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In *NIPS*, pages 361–368, 1994.

[42] Er L. Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888, 2006.

[43] Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*. MIT Press, March 1998.

[44] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

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

[46] John N. Tsitsiklis and Benjamin Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94, 1996.

[47] Benjamin Van Roy. Performance loss bounds for approximate value iteration with state aggregation. *Math. Oper. Res.*, 31(2):234–244, 2006.

[48] Benjamin Van Roy and Zheng Wen. Generalization and exploration via randomized value functions. *CoRR*, abs/1402.0635, 2014.

[49] Benjamin Van Roy and Xiang Yan. Manipulation robustness of collaborative filtering. *Management Science*, 56(11):1911–1929, 2010.

[50] Zheng Wen, Louis J. Durlofsky, Benjamin Van Roy, and Khalid Aziz. Use of approximate dynamic programming for production optimization. In *SPE Proceedings*, the Woodlands, Texas, USA, February 2011.

[51] Zheng Wen, Louis J. Durlofsky, Benjamin Van Roy, and Khalid Aziz. Approximate dynamic programming for optimizing oil production. In Frank L. Lewis and Derong Liu, editors, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Wiley-IEEE Press, 2012.

[52] Zheng Wen, Branislav Kveton, Brian Eriksson, and Sandilya Bhamidipati. Sequential bayesian search. In *ICML (2)*, pages 226–234, 2013.

[53] Zheng Wen, Daniel O'Neill, and Hamid Reza Maei. Optimal demand response using device based reinforcement learning. *CoRR*, abs/1401.1549, 2014.

[54] Zheng Wen and Benjamin Van Roy. Efficient exploration and value function generalization in deterministic systems. In *NIPS*, pages 3021–3029, 2013.

[55] Jemery Wyatt. *Exploration and Inference in Learning from Reinforcement*. PhD thesis, University of Edinburgh, 1997.

Zheng Wen

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Benjamin Van Roy) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Stephen Boyd)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ramesh Johari)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Daniel O'Neill)

Approved for the Stanford University Committee on Graduate Studies.