PAC-optimal, Non-parametric Algorithms and Bounds for Exploration in Concurrent MDPs with Delayed Updates

by

Jason Pazis

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Ronald Parr, Supervisor

_____
Vincent Conitzer

_____
George Konidaris

_____
Mauro Maggioni

_____
Peter Stone

Dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in the Department of Computer Science
in the Graduate School of Duke University
2015

Abstract

PAC-optimal, Non-parametric Algorithms and Bounds for Exploration in Concurrent MDPs with Delayed Updates

by

Jason Pazis

Department of Computer Science
Duke University

Date: _____

Approved:

_____
Ronald Parr, Supervisor

_____
Vincent Conitzer

_____
George Konidaris

_____
Mauro Maggioni

_____
Peter Stone

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University
2015

Abstract

As the reinforcement learning community has shifted its focus from heuristic methods to methods that have performance guarantees, PAC-optimal exploration algorithms have received significant attention. Unfortunately, the majority of current PAC-optimal exploration algorithms are inapplicable in realistic scenarios: 1) They scale poorly to domains of realistic size. 2) They are only applicable to discrete state-action spaces. 3) They assume that experience comes from a single, continuous trajectory. 4) They assume that value function updates are instantaneous. The goal of this work is to bridge the gap between theory and practice, by introducing an efficient and customizable PAC optimal exploration algorithm, that is able to explore in multiple, continuous or discrete state MDPs simultaneously. Our algorithm does not assume that value function updates can be completed instantaneously, and maintains PAC guarantees in realtime environments. Not only do we extend the applicability of PAC optimal exploration algorithms to new, realistic settings, but even when instant value function updates are possible, our bounds present a significant improvement over previous single MDP exploration bounds, and a drastic improvement over previous concurrent PAC bounds. We also present Bellman error MDPs, a new analysis methodology for online and offline reinforcement learning algorithms, and TCE, a new, fine grained metric for the cost of exploration.

Contents

List of Figures

List of Abbreviations and Symbols

Symbols

$\mathcal{S}$	A state space.
$\mathcal{A}$	An action space.
P	A Markovian transition model.
R	A reward function.
γ	A discount factor.
π	A policy.
π^*	An optimal policy.
$V^\pi(s)$	The value of state s under policy π.
$V^*(s)$	The value of state s under the optimal policy.
$Q^\pi(s,a)$	The value of state-action (s,a) under policy π.
$Q^*(s,a)$	The value of state-action (s,a) under the optimal policy.
B^π	The Bellman operator for policy π.
B	The optimal Bellman operator.
$\tilde{B}^\pi$	The approximate optimistic Bellman operator for policy π.
$R_{\max}$	An upper bound on the reward function.
$Q_{\max}$	An upper bound on the value function.
$d()$	A distance function.
$\mathcal{N}_{\mathcal{SAM}}(d_{known})$	The covering number.
$\lvert\mathcal{A}\rvert$	An upper bound on the number of discrete actions the agent can take in any state.

$u(s, a, M)$	An approximation unit.
U	An approximation set.
Q_U	The value function of approximation set U.
k	The maximum number of sample pointers in an approximation unit.
k_m	An integer parameter.
k_p	The number of MDPs we are exploring in parallel.
K_a	A set of integer values less than or equal to k.
$k_a(s, a, M)$	The number of active samples in $u(s, a, M)$.
ϵ_b	An exploration bonus.
$P^{\pi}_{\gamma,s,a,M}(\bar{s}, \bar{a})$	The discounted probability of reaching $(\bar{s}, \bar{a}, M)$ from (s, a, M).
t_i	A timestep.
$D_{(s,a,r,s',\mathcal{A}_{s'}M_i),M_j}$	A sample delay.

Abbreviations

RL	Reinforcement Learning.
MDP	Markov Decision Process.
TCE	Total Cost of Exploration.

Acknowledgements

As I reflect back on how other people have influenced my progress as a scientist and as a person, I realize that I have much be grateful for, and a lot of people to be grateful to. Without the profoundly positive influence of others, I would never have started, let alone completed my PhD.

I would like to start by thanking my former advisor Michail G. Lagoudakis, whose well-written thesis served as my introduction to reinforcement learning. I remember walking into his office wondering if he would find my ideas interesting, and being surprised by how enthusiastic he was about helping me. Not only do I owe Michail the decision to come to Duke, but also the decision to pursue a PhD.

Most students consider themselves lucky if they find *one* good advisor. During my studies, I have been fortunate enough to have two excellent advisors. I would like to thank my advisor Ronald Parr, who had some difficult shoes to fill, yet has been able to meet and exceed my expectations. During my time at Duke Ron has given me the freedom to pursue my own ideas, even when that meant stepping outside of his comfort zone. His intelligence and ability to provide constructive feedback make him invaluable as a collaborator, while his calm demeanor and good sense of humor make him a pleasure to work with.

In addition to my advisors, many other faculty members have helped me not only directly with feedback on my ideas, but also indirectly by teaching me principles on how to approach research problems in general. I would like to start by thanking

Vince Conitzer and Mauro Maggioni. The seeds for many of the ideas presented in this thesis were planted while I was attending their excellent courses. I would also like to thank Peter Stone and Emma Brunskill. Their feedback has pushed me to go further. Last but not least I would like to thank George Konidaris. His fresh perspective and advice has been invaluable not only in completing my thesis, but also in pursuing the next step.

I would like to thank Marie-Jo for putting up with my hectic schedule during this last year, and for her support and enthusiasm. I am grateful to Ios, Alex, Ben and everyone else at Duke for being great friends and making the past few years an enjoyable experience rather than a chore. I would also like to thank Paul Buda for making my internship experience a pleasant and productive one, and for his invaluable advice.

Finally, I would like to thank a number of organizations that supported me through my studies. In particular I would like to thank NSF for grants IIS-0713435, IIS-1147641 and IIS-1218931, as well as ICML and AAAI. While sometimes as students we tend to forget about who is paying the bills, much of this research would not have been possible without their support.

1

Introduction

In a world where only the fittest survive, the ability to adapt to one's environment can make the difference between survival and extinction. One of the primary avenues of adaptation is evolution. By evolution we mean the process by which a species adapts to its environment by natural selection: Naturally occurring variations that are better suited to the environment are more likely to survive to the next generation. This process has been in place since life first appeared on this planet and is still active today.

Unfortunately, when compared with the lifespan of an individual organism, evolution is a very slow process. For a new, beneficial variation to become established, or for a harmful variation to be phased out of the gene pool, multiple generations are required. Thus, while evolution favors variations that have a better chance of survival, it does not help an individual become better adapted to their own environment.

In order for an individual to adapt to a new or changing environment, they have to be able to learn from experience. Learning from experience allows an organism to adapt to situations that were not faced by its ancestors often enough to force successful reactions to be hardcoded into its genes, and can have a dramatic impact

on an individual's chances of survival. One could argue that genes which give an organism the ability to learn from experience are the product of an evolutionary process which favors organisms that can adapt to new and changing environments.

A path similar to the one followed by living organisms has been followed by the field of artificial intelligence. The first few generations of robots/algorithms were unable to adapt. Their responses to stimuli were constrained to the reactions that their creators had preprogrammed. Even planning algorithms would keep following the same plan every time they were faced with the same situation, even if that same plan had consistently failed in the past.

Even though such robots have proven to be very useful in constrained environments such as manufacturing plants, researchers quickly recognized that true autonomy requires algorithms that can learn from experience.

1.1 Focus

This work is devoted to the study of learning how to act from experience. In particular, we will be focusing on the problem of simultaneous exploration and exploitation in sequential decision making problems (**exploration** for short). In an exploration setting we are presented with an unknown environment, and have to balance taking actions that will help us learn more about the environment (which will help us perform better in the future), against taking actions that we expect to have better payoff in the short term.

We will present a new metric on the effectiveness of different exploration algorithms, a new analysis methodology that simplifies the analysis of algorithms, and a concrete algorithm that will allow us to explore efficiently.

1.2 Sequential decision making

Sequential decision making problems are commonly modeled as Markov Decision Processes (MDPs), and tackled by algorithms from the field of reinforcement learning (RL), a subfield of machine learning (a subfield of artificial intelligence). Sequential decision making problems differ from simpler "one-shot" decision problems in that there is some notion of a state, and actions influence that state. To make the distinction clearer, let us look at examples from both sequential and non-sequential problems:

- A classical "one-shot" problem would be a spam filter. Classifying a particular email as spam (or not), does not alter the probability that the next email received will be spam.
- By contrast, in a patient treatment problem, administering a certain drug at one point in the process may have a strong impact on the patient's status at the next step.

What makes the particular class of sequential decision making problems we will be studying both interesting and challenging, is that we usually do not have examples of "good" or "bad" behavior. Instead what we will have, or what we may be able to collect from interaction with the process, are samples. These samples will give us some idea of the effects of different actions from various states. Using this limited information, our task is to not only try to find the action that looks best at this time-step, but to try to balance long and short term goals.

Problems that fall under this category include problems as diverse as scheduling a fleet of delivery trucks, flying an autonomous helicopter, or even controlling the electric power distribution of an entire country.

1.3 Exploration in sequential decision making problems

Both sequential and non-sequential decision making problems exist for which a set of samples from which to learn is given to us. Unfortunately this is not always the case; there are many problems of practical interest for which acquiring samples is part of the problem. In non-sequential settings the problem is comparatively straightforward: At every decision point, the decision maker can decide both if and where they are going to explore or exploit.

By contrast, in the class of sequential decision making problems we will be examining, the decision maker is not free to explore or exploit any part of the state-action space at will. Instead, it has to follow the system's dynamics, making decisions not only based on the ability to find something new about the environment at the current step, but also on the potential of its actions to lead to parts of the state-action space for which limited information is available.

1.3.1 Heuristic exploration methods

The first class of exploration methods to appear in the literature can be broadly characterized as heuristic exploration methods. While many heuristic algorithms for exploration exist (such as ϵ-greedy, or Boltzmann exploration) they follow the same general pattern: At every timestep the algorithm chooses to act greedily (exploit) or act randomly (explore) based on some probability distribution. In the simplest case the probability that the algorithm will act randomly remains constant at every step, and if the goal is pure exploration, the probability can be equal to one.

Unfortunately, it is easy to design examples for which heuristic exploration algorithms can take a number of steps that is exponential in the size of the state-action space before they explore adequately. Consider for example the process pictured in Figure 1.1 (sometimes referred to as a combination lock (Li, 2009)). Assume that the

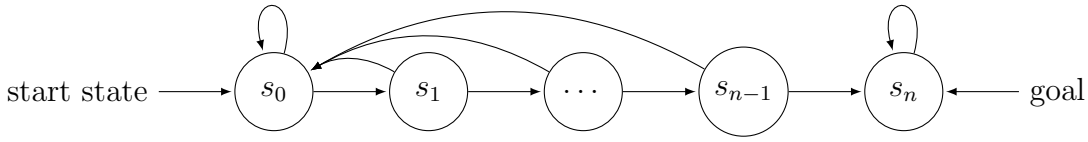

FIGURE 1.1: Example of a process for which heuristic exploration algorithms can perform poorly. Vertices represent states and edges actions.

learner starts from state s_0 and selects actions uniformly at random. The number of steps until the goal state s_n is discovered can be exponential in the number of states n.

Due to this shortcoming, heuristic exploration algorithms offer only asymptotic performance guarantees (if any guarantees are provided at all). Asymptotic guarantees tell us that as time tends to infinity our algorithm will perform close to optimal. While these might be better than no guarantees at all, their usefulness in real world problems is debatable.

1.3.2 PAC-optimal exploration methods

The algorithms and bounds presented in this thesis belong to a second class of exploration methods called **PAC-optimal exploration** methods. Characterizing an algorithm as **PAC**, which stands for Probably Approximately Correct, is a concise way to describe an algorithm that is guaranteed with some probability $1 - \delta$ (probably) to achieve performance within some constant ϵ (approximately) of the optimal (correct).

As the reinforcement learning community has shifted its focus from heuristic methods to methods that have performance guarantees, PAC-optimal exploration algorithms have received significant attention. Unfortunately, the majority of current PAC-optimal exploration algorithms are inapplicable in realistic scenarios:

- They scale poorly to domains of realistic size.

- They are only applicable to discrete state-action spaces.
- They assume that experience comes from a single, continuous trajectory.
- They assume that value function updates are instantaneous.

The goal of this work is to bridge the gap between theory and practice, by introducing a customizable algorithm that can maintain strong theoretical guarantees while exploring in multiple MDPs in parallel, even when value function updates are not instantaneous, and each MDP has a large and/or continuous state-action space.

1.4 Outline

- Chapter 2 provides an introduction to Markov decision processes, reinforcement learning, and other concepts used throughout the thesis.
- Chapter 3 introduces the concept of Bellman error MDPs, a new analysis methodology for RL algorithms.
- Chapter 4 introduces TCE, a new, fine-grained metric for comparing exploration algorithms on the sum of the differences in expected accumulated discounted reward between the optimal policy and our algorithm's policy.
- Chapter 5 describes the setting we will be analyzing our algorithm in, and presents a number of definitions and assumptions that hold throughout the thesis.
- Chapter 6 presents a customizable concurrent exploration algorithm, defines our concurrency, time, and delay models, and analyses our algorithm's space and time complexity.
- Chapter 7 analyses the TCE of our algorithm for three different parameter settings.

- Chapter 8 discusses how to explore in discrete state-action spaces, how to explore in continuous action spaces, and how to incorporate prior knowledge.
- Chapter 9 provides an overview of related work in 1) Bellman error bounds, 2) cost of exploration metrics, 3) concurrent learning in multiple MDPs, 4) PAC optimal exploration in continuous and discrete state-action spaces, and 5) other forms of exploration.
- Chapter 10 provides experimental results on the pendulum swing-up problem.
- Chapter 11 concludes this work by summarizing our contribution, and offers directions for future work.

2

Background

Let $\mathcal{X}$ be the domain of x. Throughout this thesis, $\forall x$ will serve as a shorthand for $\forall x \in \mathcal{X}$. For example we will use $\forall(s, a)$ as a shorthand for $\forall(s, a) \in (\mathcal{S}, \mathcal{A})$.

2.1 Markov decision processes (MDPs)

Markov decision processes (**MDPs**) (Puterman, 1994) provide a discrete-time mathematical decision-making modeling framework, that is particularly useful when modeling processes for which the outcome is a function of the agent's actions perturbed by external influences (such as noise). MDPs have found extensive use in areas such as economics, control, manufacturing, and reinforcement learning.

An MDP is a 5-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where

- $\boldsymbol{\mathcal{S}}$ is the state space of the process. It can be finite set, or it can be infinite as is the case when there is some state variable that can take values in a continuous range. The current state s is a complete description of the environment at the current timestep.

- $\boldsymbol{\mathcal{A}}$ is the action space of the process. Just like the state space, it can be finite

or infinite. The set of actions consist of the possible choices an agent has at each state. Note that not all actions have to be available at every state. We will use $\boldsymbol{\mathcal{A}_s}$ to denote the set of actions available at state s.

- $\boldsymbol{P}$ is a Markovian transition model. $p(s'|s,a)$ denotes the probability density of transitioning to state s' when taking action a in state s. The Markov property dictates that the probability density of transitioning to state s' when taking action a in state s depends only on s and a, and not on the history of the process.

- $\boldsymbol{R}$ is a reward function. $R(s,a,s')$ is a real valued, scalar reward for taking action a in state s and transitioning to state s'. Like the transition model, the rewards do not depend on the history of the process.

- $\boldsymbol{\gamma} \in [0,1)$ is a discount factor for future rewards. It expresses the fact that we care more about rewards received now than in the future. The reward we receive at the current timestep is weighted by 1, while the reward received t steps in the future is exponentially discounted by γ^t. In the extreme case where $\gamma = 0$ the problem degenerates to picking the action that will yield the largest immediate reward (supervised learning).

2.2 Policies, value functions, Bellman operators, and Bellman backups

A *policy* $\boldsymbol{\pi}$ for an MDP is a (deterministic or stochastic) mapping $\pi : \mathcal{S} \mapsto \mathcal{A}$ from states to actions. For deterministic policies, $\pi(s)$ denotes the action choice in state s.

The value $\boldsymbol{V^{\pi}(s)}$ of a state s under a policy π is defined as the expected accumulated discounted reward obtained when the process begins in state s and all decisions

are made according to policy π.

$$\boldsymbol{V^{\pi}(s)} = E_{a_t \sim \pi; s_t \sim P}\left[\sum_{t=0}^{\infty} \gamma^t R\big(s_t, a_t, s_{t+1}\big) \Big| s_0 = s\right].$$

V^{π} can also be defined recursively via the Bellman equation:

$$\boldsymbol{V^{\pi}(s)} = \int_{s'} p(s'|s, \pi(s)) \left(R(s, \pi(s), s') + \gamma V^{\pi}(s')\right).$$

Similarly, the value $\boldsymbol{Q^{\pi}(s, a)}$ of a state-action pair (s, a) under policy π is defined as the expected accumulated discounted reward when the process begins in state s by taking action a and all decisions thereafter are made according to policy π. For a fixed policy π the Bellman operator for Q is defined as:

$$\boldsymbol{B^{\pi}Q(s, a)} = \int_{s'} p(s'|s, a)\Big(R(s, a, s') + \gamma Q(s', \pi(s'))\Big),$$

while the optimal Bellman operator is defined as:

$$\boldsymbol{BQ(s, a)} = \int_{s'} p(s'|s, a)\Big(R(s, a, s') + \gamma \sup_{a'} Q(s', a')\Big).$$

A single application of a Bellman operator is called a **Bellman backup**.

An optimal policy $\boldsymbol{\pi^*}$ yields the optimal value function $\boldsymbol{V^*(s)}$ for every state s. Every MDP has at least one deterministic, optimal policy although it may not be unique (multiple deterministic or stochastic policies can be optimal, yielding equal expected total discounted reward through different actions). The optimal value function $V^*(s)$ can be defined recursively via the Bellman optimality equation:

$$\boldsymbol{V^*(s)} = \sup_{a} \left\{ \int_{s'} p(s'|s, a) \left(R(s, a, s') + \gamma V^*(s')\right) \right\}.$$

The Bellman optimality equation for Q becomes:

$$\boldsymbol{Q^*(s, a)} = \int_{s'} p(s'|s, a) \left(R(s, a, s') + \gamma \sup_{a'} \{Q^*(s', a')\}\right).$$

Note that $V^*(s), Q^*(s,a)$ as well as $V^\pi(s)$ and $Q^\pi(s,a)$ for any fixed π, are unique.

If V^* is known, an optimal policy can be extracted only if the full MDP model of the process is also known, to allow for one-step look-aheads. On the other hand, if Q^* is known, a greedy policy, that selects actions that maximize Q^* in each state is an optimal policy and can be extracted without the MDP model.

2.3 Convergence of the Bellman operator

Applying the Bellman operator to any value function repeatedly, converges to the true value function as the number of applications tends to infinity: $B^i Q(s,a) \to Q^*(s,a)$ as $i \to \infty$, and $(B^\pi)^i Q(s,a) \to Q^\pi(s,a)$ as $i \to \infty$.

For a finite number of applications of the Bellman operator, when $Q_{\min} \leqslant Q(s,a) \leqslant Q_{\max}$ and $Q_{\min} \leqslant Q^\pi(s,a) \leqslant Q_{\max}$ $\forall (s,a)$ we have that $\forall (s,a)$:

$$|Q^\pi(s,a) - (B^\pi)^i Q(s,a)| \leqslant \gamma^i (Q_{\max} - Q_{\min}).$$

Similarly, when $Q_{\min} \leqslant Q(s,a) \leqslant Q_{\max}$ and $Q_{\min} \leqslant Q^*(s,a) \leqslant Q_{\max}$ $\forall (s,a)$ we have that $\forall (s,a)$:

$$|Q^\pi(s,a) - B^i Q(s,a)| \leqslant \gamma^i (Q_{\max} - Q_{\min}).$$

2.4 Contractions and fixed points

The reason that repeated applications of a Bellman operator converge to the true value is that the Bellman operator is a γ-contraction in maximum norm:

Definition 2.4.1. *An operator $\hat{B}$ is called a* **γ-contraction** *for $0 \leqslant \gamma < 1$ if*

$$||\hat{B}Q_1 - \hat{B}Q_2||_\infty \leqslant \gamma ||Q_1 - Q_2||_\infty \ \forall \ Q_1, Q_2.$$

When applied to a complete metric space, an operator that is a γ-contraction in maximum norm has a unique fixed point (it converges to a unique value).

2.5 Value iteration

Value iteration, one of the simplest and most widely used algorithms for computing the optimal value function, takes advantage of the contraction property of the Bellman operator. It works by repeatedly performing Bellman backups using the Bellman optimality operator:

1. $\forall(s, a)$: initialize $Q(s, a)$ to an arbitrary value in $[Q_{\min}, Q_{\max}]$.
2. $\forall(s, a)$: set $Q(s, a)$ to $BQ(s, a)$.
3. If convergence criteria are not satisfied, go to step 2.
4. Return Q.

2.6 Reinforcement learning (RL)

In reinforcement learning (Kaelbling et al., 1996; Sutton and Barto, 1998), a learner interacts with a stochastic process modeled as an MDP and typically observes the state and immediate reward at every step; however, the transition model P and reward function R are not known. The goal is to learn a near optimal policy using experience collected through interaction with the process. At each step of interaction, the learner observes the current state s, chooses an action a, and observes the reward received r, the resulting next state s', and the set of available actions in $\mathcal{A}_{s'}$, essentially sampling the transition model and reward function of the process. Thus experience comes in the form of $(s, a, r, s', \mathcal{A}_{s'})$ samples.

2.7 Value function approximation

In many real world applications, the number of state-action pairs is too large (or even infinite if the state or action spaces are continuous), rendering exact representation

of the value function impractical. In addition to the fact that some spaces are too large to be represented exactly, we may not have samples for every state-action pair, or processing all the samples may exceed our computational resources. In such cases we are forced to use some form of function approximation.

Many function approximation methods from supervised learning have been used with reinforcement learning, including neural networks, decision trees and forests, nearest neighbors, and linear combinations of (possibly non-linear) basis functions, and kernels. These methods offer different tradeoffs regarding representational power, convergence when combined with particular learning algorithms, generalization ability, number of parameters to be learned and human effort required in picking appropriate parameters, with no method having a clear advantage in all situations.

2.8 Concentration inequalities

One of the main mathematical tools that we will be using throughout this thesis is concentration inequalities. Concentration inequalities provide guarantees that with high probability and under sufficient conditions, sums or even functions of random variables are close to their expectation. In particular, we will make extensive use of the multiplicative form of Chernoff's bound, and McDiarmid's inequality.

The multiplicative form of Chernoff's bound is very useful in cases where we know (or have a bound on) the expected value of the sum of a number of independent Bernoulli random variables, but do not want the number of variables appearing in our bound.

Lemma 2.8.1. (**Chernoff's bound**)
Let t_i for $i = 0 \rightarrow l$ be independent (but not necessarily identically distributed) random variables in $\{0, 1\}$, with $P(t_i = 1) = p_i$ and $\sum_{i=0}^{l} p_i = \mu$. Then:

$$P\left(\sum_{i=0}^{l} t_i \leqslant (1-\epsilon)\mu\right) \leqslant e^{\frac{-\epsilon^2\mu}{2}},$$

and

$$P\left(\sum_{i=0}^{l} t_i \geqslant (1+\epsilon)\mu\right) \leqslant e^{\frac{-\epsilon^2\mu}{3}}.$$

McDiarmid's inequality is very useful in cases where we know (or have a bound on) the expected value of a function of independent random variables, but the only other information we have about the function is a bound on how much it can change if we change one of its inputs.

Lemma 2.8.2. (**McDiarmid's inequality** (McDiarmid, 1989))
Let $X_1, X_2, \ldots X_n$ be independent (but not necessarily identically distributed) random variables, and f be a function for which replacing x_i changes the value of f by at most c_i:

$$\sup_{x_1, x_2, \ldots x_n, \hat{x}_i} |f(x_1, x_2, \ldots x_n) - f(x_1, x_2, \ldots, x_{i-1}, \hat{x}_i, x_{i+1}, \ldots x_n)| \leqslant c_i, \ \forall \ i \ \in [1, n].$$

We have that:

$$P\left(f(X_1, X_2, \ldots X_n) - E[f(X_1, X_2, \ldots X_n)] \geqslant \epsilon\right) \leqslant e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}},$$

$$P\left(E[f(X_1, X_2, \ldots X_n)] - f(X_1, X_2, \ldots X_n) \geqslant \epsilon\right) \leqslant e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}},$$

$$P\left(|f(X_1, X_2, \ldots X_n) - E[f(X_1, X_2, \ldots X_n)]| \geqslant \epsilon\right) \leqslant 2e^{-\frac{2\epsilon^2}{\sum_{i=1}^{n} c_i^2}}.$$

Notice that no further assumptions are made about function f. It could for example be the case that f has been generated from the same set of samples we are trying to get a bound on.

We will also use Cantelli's inequality (also known as the one sided Chebyshev's inequality). Cantelli's inequality is useful when the square root of the variance of a random variable is significantly lower than the range of values it can cake.

Lemma 2.8.3. (**Cantelli's inequality**)

Let X be a random variable with mean $E[X] = \mu$ and variance $\sigma^2 = var[X]$. Then:

$$P\left(X \geqslant \mu + \epsilon\right) \leqslant \frac{\sigma^2}{\sigma^2 + \epsilon^2},$$

and

$$P\left(X \leqslant \mu - \epsilon\right) \leqslant \frac{\sigma^2}{\sigma^2 + \epsilon^2}.$$

2.9 Sample complexity

There have been many definitions of sample complexity in RL. For PAC-optimal exploration algorithms in discounted MDPs, the most commonly used one is the following (Kakade, 2003):

Definition 2.9.1. *Let π be an arbitrarily complex, possibly non-stationary, possibly history dependent policy (such as the policy followed by an exploration algorithm), and let $(s_1, s_2, s_3, \dots)$ be a random path generated by π. The* **sample complexity** *of exploration is the number of time-steps t such that $V^{\pi}(s_t) < V^{*}(s_t) - \epsilon$.*[1]

We will be defining a more fine-grained sample complexity metric in Section 4.3.

2.10 Efficient PAC optimal exploration

The notion of efficient PAC optimal exploration has also evolved over the years. We will be using the following definition (Strehl et al., 2009):

Definition 2.10.1. *An algorithm is said to be* **efficient PAC-MDP** (*Probably Approximately Correct in Markov Decision Processes*) *if, for any $\epsilon > 0$ and $0 < \delta < 1$, its sample complexity, its per-timestep computational complexity, and its space complexity, are less than some polynomial in the relevant quantities $(S, A, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma})$, with probability at least $1 - \delta$.*

[1] Note that $V^{\pi}(s_t)$ denotes the expected discounted accumulated reward of the arbitrarily complex policy π from state s at time t, rather than the expectation of some stationary snapshot of π.

Over the years, some researchers have dropped the polynomial per-timestep computational and space complexity requirements and have used the following definition instead (notice that these algorithms are called PAC-MDP, not efficient PAC-MDP):

Definition 2.10.2. *An algorithm is said to be* **PAC-MDP** (*Probably Approximately Correct in Markov Decision Processes*) *if, for any* $\epsilon > 0$ *and* $0 < \delta < 1$*, its sample complexity is less than some polynomial in the relevant quantities* $(S, A, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma})$*, with probability at least* $1 - \delta$*.*

While it is theoretically interesting to see what we can achieve when computational and space complexity requirements are lifted, algorithms whose computational and/or space complexity is super-polynomial in the relevant quantities are of limited use in practical applications.

3

Bellman Error MDPs

This chapter introduces the concept of Bellman error MDPs, a new analysis tool for RL algorithms. Existing work on bounds based on the Bellman error has focused on one-step max-norm errors (Williams and Baird, 1993). Unfortunately, for the same reason that the maximum expected discounted accumulated reward of an MDP can be much smaller than $\frac{\mathcal{R}_{\max}}{1-\gamma}$, bounds based on the max-norm of the Bellman error can be loose. Bellman error MDPs are much more flexible and intuitive, allowing us to prove stronger bounds with less effort. While this work focuses on PAC optimal exploration, Bellman error MDPs can be used to prove bounds for other online or offline RL algorithms.

3.1 Definitions

Let M be an MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with Bellman operator B^π for policy π, and Q an approximate value function for M. Additionally, $\forall s, a, \pi$ let $0 \leqslant Q^\pi(s,a) \leqslant Q_{\max}$. Let the Bellman error MDP $\boldsymbol{M_{\epsilon(\pi,Q)}}$ be an MDP which differs from M only in its reward function which is defined as $\boldsymbol{R_{\epsilon(\pi,Q)}(s,a)} = Q(s,a) - B^\pi Q(s,a)$ (notice that $Q(s,a)$ and B^π are the approximate value function and Bellman operator of the

original MDP). We will use $\boldsymbol{B^{\pi}_{\epsilon(\pi,Q)}}$ to denote $M_{\epsilon(\pi,Q)}$'s Bellman operator under policy π, and $\boldsymbol{Q^{\pi}_{\epsilon(\pi,Q)}}$ to denote its value function.

3.2 Bellman error MDP theory

Theorem 3.2.1 is the main theorem of this chapter. Most results on Bellman error MDPs follow directly from this theorem and basic properties of MDPs.

Theorem 3.2.1. *The return of π over M is equal to Q minus the return of π over $M_{\epsilon(\pi,Q)}$*[1]*:*

$$Q^{\pi}(s,a) = Q(s,a) - Q^{\pi}_{\epsilon(\pi,Q)}(s,a) \ \forall (s,a) \in (\mathcal{S},\mathcal{A}).$$

Proof. We will use induction to prove our claim: If 0 denotes a value function that is zero everywhere, all we need to prove is that $(B^{\pi})^i Q(s,a) = Q(s,a) - (B^{\pi}_{\epsilon(\pi,Q)})^i 0$ and take the limit as $i \to \infty$.

The base case $B^{\pi}Q(s,a) = Q(s,a) - B^{\pi}_{\epsilon(\pi,Q)}0$ is given by hypothesis. Assuming that $(B^{\pi})^i Q(s,a) = Q(s,a) - (B^{\pi}_{\epsilon(\pi,Q)})^i 0$ holds for i, we will prove that it also holds for $i+1$:

$$\begin{aligned}
(B^{\pi})^{i+1}Q(s,a) =& B^{\pi}(B^{\pi})^i Q(s,a) \\
=& \int_{s'} P(s'|s,a)\Big(R(s,a,s') + \gamma (B^{\pi})^i Q(s',\pi(s'))\Big) \\
=& \int_{s'} P(s'|s,a)\Big(R(s,a,s') + \gamma\Big(Q(s',\pi(s')) - (B^{\pi}_{\epsilon(\pi,Q)})^i 0\Big)\Big) \\
=& \int_{s'} P(s'|s,a)\Big(R(s,a,s') + \gamma Q(s',\pi(s'))\Big) - \\
& \qquad \int_{s'} P(s'|s,a)\left(\gamma (B^{\pi}_{\epsilon(\pi,Q)})^i 0\right) \\
=& B^{\pi}Q(s,a) - \int_{s'} P(s'|s,a)\left(\gamma (B^{\pi}_{\epsilon(\pi,Q)})^i 0\right)
\end{aligned}$$

[1] Note that this is true for any policy π not just the greedy policy over Q.

$$= Q(s,a) - R_{\epsilon(\pi)}(s,a) - \int_{s'} P(s'|s,a)\left(\gamma(B^{\pi}_{\epsilon(\pi,Q)})^i 0\right)$$

$$= Q(s,a) - (B^{\pi}_{\epsilon(\pi)})^{i+1} 0$$

If we now take the limit as $i \to \infty$ we have the original claim:

$$\lim_{i\to\infty}(B^{\pi})^i Q(s,a) = Q(s,a) - (B^{\pi}_{\epsilon(\pi,Q)})^i 0 \to$$

$$Q^{\pi}(s,a) = Q(s,a) - Q^{\pi}_{\epsilon(\pi,Q)}(s,a)$$

□

Corollary 3.2.2 bounds the range of the Bellman error MDP value function, a property that will prove very useful in Chapter 7. It follows from Theorem 3.2.1 and the fact that $0 \leqslant Q^{\pi}(s,a) \leqslant Q_{\max}$.

Corollary 3.2.2. *Let* $0 \leqslant Q(s,a) \leqslant Q_{\max}\ \forall(s,a)$. *Then:*

$$-Q_{\max} \leqslant Q^{\pi}_{\epsilon(\pi,Q)}(s,a) \leqslant Q_{\max}\ \forall(s,a,\pi).$$

Lemma 3.2.3 below (a consequence of Theorem 3.2.1), proves that the difference between the optimal and the greedy policy over Q is bounded above by the inverse difference of the value of those policies in their respective Bellman error MDPs.

Lemma 3.2.3. $\forall(s,a,Q)$*:*

$$V^*(s) - V^{\pi^Q}(s) \leqslant Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,\pi^Q(s)) - Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,\pi^*(s))$$

Proof.

$$Q(s,\pi^*(s)) \leqslant Q(s,\pi^Q(s)) \Rightarrow$$

$$Q^{\pi^*}(s,\pi^*(s)) + Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,\pi^*(s)) \leqslant Q^{\pi^Q}(s,\pi^Q(s)) + Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,\pi^Q(s)) \Rightarrow$$

$$Q^{\pi^*}(s,\pi^*(s)) - Q^{\pi^Q}(s,\pi^Q(s)) \leqslant Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,\pi^Q(s)) - Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,\pi^*(s)) \Rightarrow$$

$$V^*(s) - V^{\pi^Q}(s) \leqslant Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,\pi^Q(s)) - Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,\pi^*(s))$$

□

Lemma 3.2.4 proves that if a value function can be decomposed into a weighted sum of value functions, the expected value of a policy in its Bellman error MDP is equal to the weighted sum of the expected value of the same policy in the Bellman error MDPs of the decomposed value functions.

Lemma 3.2.4. *Let* $Q(s,a) = \sum_{i=1}^{n} (w_i Q_i(s,a))$ *and* $\sum_{i=1}^{n} w_i = 1$. *Then* $\forall (s,a) \in (\mathcal{S}, \mathcal{A})$*:*

$$Q^{\pi}_{\epsilon(\pi,Q)}(s,a) = \sum_{i=1}^{n} \left(w_i Q^{\pi}_{\epsilon(\pi,Q_i)}(s,a) \right).$$

Furthermore, if $0 \leqslant Q_i(s,a) \leqslant Q_{\max}\ \forall (s,a)$*, then:*

$$-Q_{\max} \leqslant Q^{\pi}_{\epsilon(\pi,Q_i)}(s,a) \leqslant Q_{\max},\ \forall (s,a,\pi)$$

Proof. For the first part of the lemma we have:

$$\begin{aligned}
&\textstyle\sum_{i=1}^{n} \left(w_i Q^{\pi}_{\epsilon(\pi,Q_i)}(s,a) \right) = \\
&\quad = \sum_{i=1}^{n} \left(w_i \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[Q_i(s',a') - B^{\pi} Q_i(s',a') \right] \right) \\
&\quad = \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[\sum_{i=1}^{n} (w_i Q_i(s',a')) - \sum_{i=1}^{n} (w_i B^{\pi} Q_i(s',a')) \right] \\
&\quad = \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[Q(s',a') - \sum_{i=1}^{n} \left(w_i P(\bar{s}|s',a') \left(R(s',a',\bar{s}) + \gamma Q_i(\bar{s}, \pi(\bar{s})) \right) \right) \right] \\
&\quad = \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[Q(s',a') - P(\bar{s}|s',a') \left(R(s',a',\bar{s}) + \gamma \sum_{i=1}^{n} \left(w_i Q_i(\bar{s}, \pi(\bar{s})) \right) \right) \right] \\
&\quad = \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[Q(s',a') - P(\bar{s}|s',a') \left(R(s',a',\bar{s}) + \gamma Q(\bar{s}, \pi(\bar{s})) \right) \right] \\
&\quad = \sum_{t=0}^{\infty} \gamma^t E_{(s',a'|\pi,t)} \left[Q(s',a') - B^{\pi} Q(s',a') \right]
\end{aligned}$$

$$= Q^{\pi}_{\epsilon(\pi,Q)}(s,a)$$

The second part follows directly from Corollary 3.2.2. □

Lemma 3.2.5 proves an upper bound on the expected value of the greedy policy in its Bellman error MDP when we do not have a uniform bound on the Bellman error over all state-actions.

Lemma 3.2.5. *Let $X_1, \ldots, X_i, \ldots, X_n$ be sets of state-actions where $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_i \ \forall (s,a) \in X_i$, $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_{\pi^Q} \ \forall (s,a) \notin \cup_{i=1}^{n} X_i$, and $\epsilon_{\pi^Q} \leqslant \epsilon_i \forall i$. Let $p_i^e(s,a,t)$ for $t \in [0, T-1]$ be Bernoulli random variables expressing the probability of taking a state-action (s_t, a_t) at step t, for which $(s_t, a_t) \in X_i$, when starting from state-action (s,a) and following π^Q thereafter. Then:*

$$Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,a) \leqslant \frac{\epsilon_{\pi^Q}}{1-\gamma} + \epsilon_e,$$

where $\epsilon_e = \sum_{i=1}^{n} \left(\sum_{t=0}^{T-1} \left(\gamma^t p_i(s,a,t)\right) \left(\epsilon_i - \epsilon_{\pi^Q}\right) \right) + \gamma^T Q_{\max}$.

Proof. The expected reward in the Bellman error MDP for step $t \in [0, T-1]$ is upper bounded by $\epsilon_{\pi^Q} + \sum_{i=1}^{n} \left(p_i(s,a,t)(\epsilon_i - \epsilon_{\pi^Q})\right)$. From Corollary 3.2.2 the value of any state-action that may be encountered at step T in the Bellman error MDP is upper bounded by $Q_{\max}$. From these two facts we have that:

$$\begin{aligned} Q^{\pi^Q}_{\epsilon(\pi^Q,Q)}(s,a) &\leqslant \sum_{t=0}^{T-1} \left(\gamma^t \left(\epsilon_{\pi^Q} + \sum_{i=1}^{n} \left(p_i^e(s,a,t)(\epsilon_i - \epsilon_{\pi^Q})\right) \right) \right) + \gamma^T Q_{\max} \\ &\leqslant \frac{\epsilon_{\pi^Q}}{1-\gamma} + \sum_{i=1}^{n} \left(\sum_{t=0}^{T-1} \left(\gamma^t p_i^e(s,a,t)\right) \left(\epsilon_i - \epsilon_{\pi^Q}\right) \right) + \gamma^T Q_{\max}. \end{aligned}$$

□

Lemmas 3.2.6 and 3.2.7 prove bounds on the difference in expected value between the optimal policy and the greedy policy over Q when we do not have a uniform

bound on the Bellman error over all state-actions. They avoid the square peg in round hole problem encountered when one tries to analyze exploration algorithms using max-norm bounds. We will make extensive use of Lemma 3.2.7 in Chapter 7.

Lemma 3.2.6. *Let* $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant -\epsilon_* \ \forall (s,a)$, $X_1, \ldots, X_i, \ldots, X_n$ *be sets of state-actions where* $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_i \ \forall (s,a) \in X_i$, $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_{\pi^Q} \ \forall (s,a) \notin \cup_{i=1}^{n} X_i$, *and* $\epsilon_{\pi^Q} \leqslant \epsilon_i \forall i$. *Let* $p_i(s,a,t)$ *for* $t \in [0, T-1]$ *be Bernoulli random variables expressing the probability of taking a state-action* (s_t, a_t) *at step* t, *for which* $(s_t, a_t) \in X_i$, *when starting from state-action* (s,a) *and following* π^Q *thereafter. Then:*

$$V^*(s) - V^{\pi^Q}(s) \leqslant \frac{\epsilon_* + \epsilon_{\pi^Q}}{1-\gamma} + \epsilon_e,$$

where $\epsilon_e = \sum_{i=1}^{n} \left(\sum_{t=0}^{T-1} \left(\gamma^t p_i(s,a,t)\right) \left(\epsilon_i - \epsilon_{\pi^Q}\right) \right) + \gamma^T Q_{\max}$.

Proof. Since $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant -\epsilon_* \ \forall (s,a)$, we have that $Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,a) \geqslant -\frac{\epsilon_*}{1-\gamma}$. Substituting the above along with Lemma 3.2.5 into Lemma 3.2.3 concludes our proof. □

Lemma 3.2.7. *Let* $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant -\epsilon_* \ \forall (s,a)$, $X_1, \ldots, X_i, \ldots, X_n$ *be sets of state-actions where* $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_i \ \forall (s,a) \in X_i$, $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon_{\pi^Q} \ \forall (s,a) \notin \cup_{i=1}^{n} X_i$, *and* $\epsilon_{\pi^Q} \leqslant \epsilon_i \forall i$. *Let* $T_H = \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil$ *and define* $H = \{1, 2, 4, \ldots, 2^i\}$ *where* i *is the largest integer such that* $2^i \leqslant T_H$. *Define* $p_{h,i}(s,a)$ *for* $h \in [0, T_H - 1]$ *be Bernoulli random variables expressing the probability of encountering exactly* h *state-actions for which* $(s,a) \in X_i$ *when starting from state-action* (s,a) *and following* π^Q *thereafter for a total of* $\min\{T, T_H\}$ *steps. Finally let* $p^e_{h,i}(s_{t,j}) = \sum_{m=h}^{2h-1} p_{m,i}(s_{t,j})$. *Then:*

$$V^*(s) - V^{\pi^Q}(s) \leqslant \frac{\epsilon_* + \epsilon_{\pi^Q}}{1-\gamma} + \epsilon_s + \epsilon_e,$$

where $\epsilon_e = 2\sum_{i=1}^{n}\left(\sum_{h\in H}\left(hp_{h,i}(s,a)\right)\left(\epsilon_i - \epsilon_{\pi^Q}\right)\right) + \gamma^T Q_{\max}$.

Proof. Let $p_i(s,a,t)$ for $t \in [0, T-1]$ be Bernoulli random variables expressing the probability of taking a state-action (s_t, a_t) at step t, for which $(s_t, a_t) \in X_i$, when starting from state-action (s,a) and following π^Q thereafter. We have that $\forall i$:

$$\sum_{t=0}^{T-1}\left(\gamma^t p_i(s,a,t)\right) \leqslant \sum_{t=0}^{T-1} p_i(s,a,t) = \sum_{h=1}^{T}\left(hp_{h,i}(s,a)\right) \leqslant 2\sum_{h\in H}\left(hp^e_{h,i}(s,a)\right).$$

We also have that:

$$\gamma^{\min\{T,T_H\}} Q_{\max} \leqslant \gamma^{T_H} Q_{\max} + \gamma^T Q_{\max} \leqslant \epsilon_s + \gamma^T Q_{\max},$$

and the result follows from Lemma 3.2.6. □

3.3 Using Bellman error MDPs

An important consequence of Lemma 3.2.3 is that in order to get a bound on the difference in performance between the optimal policy and the greedy policy over a value function, we only need a lower bound on the performance of an optimal policy and an upper bound on the performance of the greedy policy in their respective Bellman error MDPs.

The algorithm presented in Chapter 6 relies on an optimistic approximation scheme. Because of that, we will be able to get a good bound on the performance of the optimal policy in its Bellman error MDP from the start, while our bound on the performance of the greedy policy will improve as more samples are gathered.

3.4 Max-norm bounds

Previously known bounds based on the max-norm of the Bellman error can now be derived as a direct consequence of the fact that in a discounted MDP where the reward function is in $[R_{\min}, R_{\max}]$, the value function is bounded by $\left[\frac{R_{\min}}{1-\gamma}, \frac{R_{\max}}{1-\gamma}\right]$.

Lemma 3.4.1. (Theorem 4.1 and Corollary 4.1 in Williams and Baird (1993))
Let $-\epsilon \leqslant Q(s,a) - BQ(s,a) \leqslant \epsilon \ \forall (s,a) \in (\mathcal{S}, \mathcal{A})$ *Then:*

$$V^{\pi^Q}(s) \geqslant V^*(s) - \frac{2\epsilon}{1-\gamma}.$$

If additionally $Q(s,a) - BQ(s,a) \geqslant 0 \ \forall (s,a)$*, and/or* $V^*(s) \leqslant Q(s, \pi^Q(s)) \ \forall s$*, then:*

$$V^{\pi^Q}(s) \geqslant V^*(s) - \frac{\epsilon}{1-\gamma}.$$

Proof. Since $B^{\pi^Q}Q(s,a) = BQ(s,a)$ and $B^{\pi^*}Q(s,a) \leqslant BQ(s,a)$, we have that $Q(s,a) - B^{\pi^Q}Q(s,a) \leqslant \epsilon$ and $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant -\epsilon$, which means that the reward of the Bellman error MDP for the greedy policy is bounded above by ϵ, and the reward of the Bellman error MDP for an optimal policy is bounded below by $-\epsilon$. The result for the first part of the lemma follows from Lemma 3.2.3.

For the second part of the lemma we additionally have that $Q(s,a) - BQ(s,a) \geqslant 0$ implies $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant 0$ which means that the reward of the Bellman error MDP for an optimal policy is bounded below by 0 and the result follows from Lemma 3.2.3.

From Theorem 3.2.1 we have that $Q^{\pi^*}(s, \pi^Q(s)) = Q(s, \pi^Q(s)) - Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s, \pi^Q(s))$. $V^*(s) \leqslant Q(s, \pi^Q(s))$ implies $Q(s, \pi^Q(s)) - Q^{\pi^*}(s, \pi^Q(s)) \geqslant 0$, which implies $Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s, \pi^Q(s)) \geqslant 0 \ \forall s$. Once again the result follows from Lemma 3.2.3. □

Lemma 3.4.2. (Lemma 4.6 in Williams and Baird (1993))
If $Q(s,a) - BQ(s,a) \geqslant 0 \ \forall (s,a) \in (\mathcal{S}, \mathcal{A})$ *Then:*

$$Q(s, \pi^Q(s)) \geqslant V^*(s).$$

Proof. Since $B^{\pi^*}Q(s,a) \leqslant BQ(s,a)$, we have that $Q(s,a) - B^{\pi^*}Q(s,a) \geqslant 0$, which means that the reward of the Bellman error MDP for an optimal policy is bounded below by 0 and $Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,a) \geqslant 0$. From Theorem 3.2.1 we have that $Q^{\pi^*}(s,a) = Q(s,a) - Q^{\pi^*}_{\epsilon(\pi^*,Q)}(s,a)$ which implies $Q(s,a) \geqslant Q^{\pi^*}(s,a) \ \forall (s,a)$. Since $Q(s, \pi^Q(s)) \geqslant Q(s, \pi^*(s)) \ \forall s$, the result follows. □

4

The Total Cost of Exploration

This chapter introduces a new fine-grained metric for comparing exploration algorithms on the sum of the differences in expected accumulated discounted reward between the optimal policy and an exploration algorithm's policy: the Total Cost of Exploration (TCE).

Before we present TCE, we will give a brief overview of the two most popular metrics available in the literature today.

4.1 Regret

Regret is a metric that shares the fine-grained nature of TCE. While it has gained significant traction in other exploration settings (Ortner and Ryabko, 2012), it is unsuitable for our setting.

Regret for undiscounted MDPs is typically defined as the difference between the expected accumulated reward of an optimal policy and the accumulated reward achieved by the algorithm in question. Since in discounted MDPs we care about discounted accumulated reward instead, regret is defined as the difference between the expected accumulated discounted reward of an optimal policy and the discounted

accumulated reward achieved by the algorithm in question. For MDPs with non-negative rewards, regret for the discounted setting is upper bounded by $\frac{R_{\max}}{1-\gamma}$. There are at least two ways in which regret in discounted MDPs can be arbitrarily close to this upper bound:

1. A common and arguably necessary (Jaksch et al., 2010) assumption in regret analysis is a finite mixing time/diameter. By assuming a finite diameter, we assume that our algorithm cannot get trapped in a "bad" set of states. Unfortunately this is a very strong assumption that is not true for many real world domains (such as for any robotic system that can get damaged or trapped, any living organism that can get injured or simply bored and stop using our system, and any financial entity that can go broke)[1]. As a simple example of why the lack of finite diameter can yield regret arbitrarily close to $\frac{R_{\max}}{1-\gamma}$, consider an MDP with two states (Figure 4.1). The starting state has two actions: the first is self-transitioning with a reward of $R_{\max}$, while the second has zero reward and transitions to an absorbing state with zero reward. If the algorithm is unlucky and chooses the second action first, its regret will be $\frac{R_{\max}}{1-\gamma}$.

2. Even if we assume a finite diameter, it is easy to design discounted MDPs for which regret is arbitrarily close to $\frac{R_{\max}}{1-\gamma}$. Consider an MDP where the starting state has two actions (Figure 4.2): The first is self-transitioning with reward $R_{\max}$, while the second one has zero reward and transitions to an n long chain of states each of which has zero reward and the last of which transitions back to the starting state. If the algorithm is unlucky and chooses the second action first, its regret will be at least $(1-\gamma^n)\frac{R_{\max}}{1-\gamma}$, where n can be as large as $|S|$. The situation can become even worse if we add stochasticity.

[1] One scenario for which the finite mixing time/diameter assumption holds is when we are dealing with a simulation problem. In many simulated domains the existence of a reset action that can bring us back to a unique start state is a reasonable assumption.

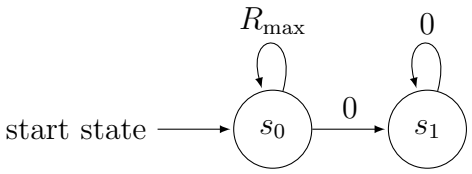

FIGURE 4.1: Example of an MDP with infinite diameter.

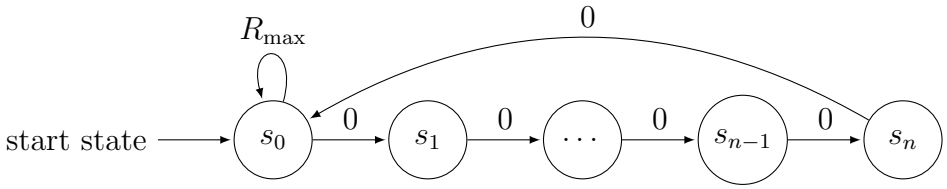

FIGURE 4.2: Example of an MDP with finite diameter for which discounted regret can be arbitrarily close to the upper bound.

4.2 Sample complexity for exploration

In the discounted MDP exploration setting, or when we cannot assume a finite diameter, the most commonly used metric is the number of time-steps t such that $V^{\pi}(s_t) < V^{*}(s_t) - \epsilon$, commonly referred to as "sample complexity". One of the main drawbacks of sample complexity is that it is very coarse: It only counts the number of significantly suboptimal steps, and does not discriminate between the magnitude of different errors.

4.3 The total cost of exploration

When interacting with the real world we typically cannot assume a finite diameter, yet we may care about more than just the number of highly suboptimal steps an exploration algorithm might incur. For example, if we are using our algorithm to explore using a robot or some other physical system, it would be more useful to have an upper bound on the amount of wear and tear of the robot, rather than a bound on the number of times the robot will be completely destroyed.

We now define a metric that does not require unrealistic assumptions, is suitable for measuring exploration performance in discounted MDPs, and provides fine-grained information:

Definition 4.3.1. *Let π be an arbitrarily complex, possibly non-stationary, possibly history dependent policy (such as the policy followed by an exploration algorithm), $(s_1, s_2, s_3, \dots)$ be a random path generated by π, ϵ a positive constant, $\mathcal{T}$ the (possibly infinite) set of time steps for which $V^\pi(s_t) < V^*(s_t) - \epsilon$, and define[2]:*

$$\epsilon_e(t) = V^*(s_t) - V^\pi(s_t) - \epsilon, \ \forall \ t \in \mathcal{T}.$$

$$\epsilon_e(t) = 0, \ \forall \ t \notin \mathcal{T}.$$

The Total Cost of Exploration (**TCE**) *is defined as the undiscounted infinite sum:*

$$\sum_{t=0}^{\infty} \epsilon_e(t).$$

It is now easy to see that for any policy for which the TCE is bounded, the number of at least ϵ-suboptimal steps is also bounded:

Lemma 4.3.2. *For any policy π for which the TCE is bounded, there will be at most:*

$$\frac{\sum_{t=0}^{\infty} \epsilon_e(t)}{\epsilon_{threshold}}$$

steps such that:

$$V^\pi(s_t) < V^*(s_t) - \epsilon - \epsilon_{threshold},$$

where $\epsilon_{threshold}$ is any positive constant.

In Chapter 7 we will prove sample complexity bounds based on TCE bounds that are at least as good as or better than other available sample complexity bounds

[2] Note that $V^\pi(s_t)$ denotes the expected discounted accumulated reward of the arbitrarily complex policy π from state s at time t, rather than the expectation of some stationary snapshot of π.

with respect to all key quantities. The same is unfortunately not true in the other direction. Given a sample complexity bound for an exploration algorithm for which $V^{\pi}(s_t) < V^{*}(s_t) - \epsilon$ for at most n time steps, the best we can say is that $\sum_{t=0}^{\infty} \epsilon_e(t) \leqslant n(V_{\max} - \epsilon)$.

Note that just like sample complexity, TCE does not make any assumptions on the algorithm or the MDP to which it is applied. For example, no assumptions are made as to whether the algorithm distinguishes between "known" and "unknown" states, or whether the MDP has a finite diameter. While beyond the scope of this work, many existing algorithms could be analyzed in terms of TCE.

4.3.1 Lower bound

We can use Lemma 4.3.2 to convert existing "number of suboptimal steps" sample complexity bounds to TCE bounds. Sample complexity is known to be log-linear in the size of the state-action space $|SA|$ (Strehl et al., 2009), quadratic in $\frac{1}{\epsilon}$, and cubic in $\frac{1}{1-\gamma}$ (Lattimore and Hutter, 2012), for an overall lower worst case bound of:

$$\Omega\left(\frac{R_{\max}|SA|\log|SA|}{\epsilon^2(1-\gamma)^3}\right).$$

Through Lemma 4.3.2 this implies a lower worst case bound of:

$$\Omega\left(\frac{R_{\max}|SA|\log|SA|}{\epsilon(1-\gamma)^3}\right)$$

for TCE. In Theorem 7.4.2 we come within $\frac{1}{1-\gamma}$ of this lower bound (ignoring logarithmic factors). This also implies that we come within $\frac{1}{1-\gamma}$ of the lower bound for "number of suboptimal steps" sample complexity.

5

PAC Optimal Exploration: Setting, Definitions and Assumptions

In the typical PAC optimal exploration setting, the state-action space is discrete and finite, value function updates happen instantaneously, and experience comes from a single, continuous trajectory. In practice, most realistic scenarios violate one or more of these assumptions. Since the goal of this work is to develop an exploration algorithm that can be used in real world problems, we will not be making any of the above assumptions.

5.1 Setting

5.1.1 Large and infinite state-action spaces

The first major roadblock preventing most PAC-optimal exploration methods from being applicable in practice is that they scale poorly to domains of realistic size. Not only do they assume that the state-action space is discrete and finite, which renders them inapplicable to continuous space domains, but their sample complexity becomes prohibitive in all but the smallest toy problems. Both drawbacks are a result of their inability to generalize.

By contrast, generalization is a central building block of the algorithm presented in Chapter 6. Our generalization scheme allows us to maintain PAC-optimal guarantees and results in an algorithm that performs well in practice.

5.1.2 Value function update delays

In many real world problems value function updates are not instantaneous. Consider the following two examples:

1. A fast-paced realtime system, whose time-step is too short to allow updating the value function between every pair of steps.

2. A mobile system where instead of computing the value function locally, we send new samples to a remote platform which returns the updated value function with some (possibly stochastic) delay.

Delayed value function updates are the norm rather than the exception in real life, yet they are not adequately addressed by current PAC-optimal exploration methods.

As we will show in our analysis, the worst case sample complexity of the algorithm presented in Chapter 6 degrades gracefully with increasing value function update delays.

5.1.3 Finite computational resources

In an effort to achieve improved sample complexity, some algorithms remember their entire history. In real life our resources are finite[1], so in Chapter 6 we will present an efficient PAC-MDP algorithm (see Definition 2.10.1).

5.1.4 Concurrent exploration

Another situation that appears often in real life is when we want to explore in multiple MDPs at the same time. This scenario encompasses exploring in multiple "copies" of

[1] Even though there exist applications for which storing the entire transition history is possible, reprocessing that history at every step is typically impractical.

the same MDP simultaneously, or exploring in multiple MDPs with varying degrees of similarity. While both of these problems are interesting, the second one presents additional difficulties. It might be tempting to try to cluster MDPs into discrete clusters, and then treat each cluster as if all MDPs in the cluster are identical. Unfortunately this approach runs into the same scalability issues as discrete state-action exploration algorithms that lack the ability to generalize. There are many cases where we are dealing with MDPs that are different enough that we do not want to treat them as identical, but we would still like to be able to take advantage of any similarities present. Consider for example the case where we have multiple MDPs that are similar in most of the state-action space, but differ drastically in a small but critical section. One of the ways this situation can occur is when we have multiple MDPs with the same dynamics but different goals.

As we will see, generalizing over MDPs is no different than generalizing over state-actions. Our approach to generalization will allow us to seamlessly handle multiple MDPs, taking advantage of any degree of similarity present.

5.2 Algorithm definitions and assumptions

We now present a number of definitions and assumptions that hold throughout the thesis.

Let $\mathcal{X}$ be the domain of x. Throughout this thesis, $\forall x$ will serve as a shorthand for $\forall x \in \mathcal{X}$. For example we will use $\forall(s,a)$ as a shorthand for $\forall(s,a) \in (\mathcal{S}, \mathcal{A})$.

In the following $\boldsymbol{s, \bar{s}, \tilde{s}, s'}$ are used to denote various states, $\boldsymbol{a, \bar{a}, \tilde{a}, a'}$ are used to denote actions, and $\boldsymbol{M, \bar{M}, \tilde{M}, M'}$ are used to denote MDPs.

We will use $\boldsymbol{(s,a,M)}$ to denote state-action (s,a) in MDP M, $\boldsymbol{Q(s,a,M)}$ to denote the Q value function of (s,a) in MDP M, $\boldsymbol{BQ(s,a,M)}$ to denote the application of the Bellman operator for MDP M to $Q(s,a,M)$, and $\boldsymbol{(s,a,r,s',\mathcal{A}_{s'},M)}$ to denote a $(s,a,r,s',\mathcal{A}_{s'})$ sample collected from MDP M.

We assume that all value functions Q live in a complete metric space. We also assume that all rewards are bounded, and without loss of generality that they lie in $\boldsymbol{[0, R_{\max}]}$.[2]

Definition 5.2.1. $\boldsymbol{Q_{\max}}$ *denotes an upper bound on the maximum expected discounted reward from any state-action in the family of MDPs we are considering:*

$$0 \leqslant Q^{\pi}(s, a, M) \leqslant \boldsymbol{Q_{\max}} \leqslant \frac{R_{\max}}{1-\gamma} \ \forall (s, a, M, \pi).$$

We also define $\boldsymbol{\hat{Q}_{\max}} = R_{\max} + \gamma Q_{\max}$.

Definition 5.2.2. $\boldsymbol{d(s, a, M, \bar{s}, \bar{a}, \bar{M})}$ *is defined to be the distance of state-action* (s, a) *in MDP* M*, to state-action* $(\bar{s}, \bar{a})$ *in MDP* $\bar{M}$ *in some well-defined distance metric. We also define the shorthand:*

$$\boldsymbol{d(s, a, M, \bar{s}, \bar{a}, \bar{M}, d_{known})} = \max\{0, d(s, a, M, \bar{s}, \bar{a}, \bar{M}) - d_{known}\}.$$

Examples of distance metrics one could use include weighted norms on the state-action-MDP space, norms on linear and non-linear transformations of the state-action-MDP space, as well as norms on features of the state-action-MDP space.

Definition 5.2.3. *The* **covering number** $\boldsymbol{\mathcal{N}_{SAM}(d_{known})}$ *of a state-action-MDP space is the size of the largest minimal set*[3] C *of state-action pairs, such that for any* (s, a) *reachable from the starting state(s) of any MDP* M *in the set of MDPs we are exploring in, there exists* $(\bar{s}, \bar{a}, \bar{M}) \in C$ *such that:*

$$d(s, a, M, \bar{s}, \bar{a}, \bar{M}) \leqslant \boldsymbol{d_{known}}.$$

[2] It is easy to satisfy this assumption in all MDPs with bounded rewards by simply shifting the reward space.

[3] A minimal set having property X is a set such that if any one of its elements is removed it will cease to have property X. There can be multiple such sets, of different cardinalities. Among the minimal sets having property X, the largest minimal set has the greatest cardinality.

The covering number generalizes the concept of cardinality of a state-action-MDP space to the continuous setting. In the discrete setting $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ is bounded above by the cardinality of the state-action-MDP space. Adjusting d_{known} allows us to trade off sample, space, and computational complexity with approximation precision in both continuous and discrete spaces.

Definition 5.2.4. $|\mathcal{A}|$ *is defined as the upper bound on the number of discrete actions the agent can take in any state. For simplicity of exposition we assume that* $|\mathcal{A}|$ *is finite. In Chapter 8 we will show how any MDP with a continuous action space can be treated as if it had a discrete action space.*

Definition 5.2.5. *The* **sample set** *is defined as a mutable set of up to* $k\mathcal{N}_{\mathcal{SAM}}(d_{known})$ $(s_i, a_i, r_i, s_i', \mathcal{A}_{s_i'}, M_i)$ *samples.*

The sample set is constrained to at most $k\mathcal{N}_{\mathcal{SAM}}(d_{known})$ samples for space and computational efficiency.

Definition 5.2.6. *An* **approximation unit** u *is defined to be a 2-tuple comprised of the following:*

1. *A real value.*
2. *A mutable set of up to* $\boldsymbol{k}$ *pointers to samples in the sample set.*

An approximation unit is identified by a state-action-MDP triple as $\boldsymbol{u(s, a, M)}$.

As the name signifies, an approximation unit is the basic unit of approximation used by our algorithm.

Definition 5.2.7. *An* **approximation set** $\boldsymbol{U}$ *is defined as a mutable set of* 0 *to* $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ *approximation units. No two approximation units in an approximation set can have the same identifying state-action-MDP triple.*

Definition 5.2.8. *Let* $\boldsymbol{K_a}$ *be a set of integer values, each of which is less than or equal to* k*. Let* $u(s,a,M)$ *be an approximation unit pointing to* k_0 *samples* $(s_i, a_i, r_i, s'_i, \mathcal{A}_{s'_i}, M_i)$ *for* $i \in [1, k_0]$*. The function* $\boldsymbol{k_a(u(s,a,M))}$ *returns the largest value* k_a *in* $\boldsymbol{K_a}$ *such that* $k_a \leqslant k_0$*. If such a value does not exist,* $\boldsymbol{k_a(u(s,a,M))}$ *returns* 0*. We will call the first* $k_a(u(s,a,M))$ *samples of* $u(s,a,M)$ *the* **active samples** *of* $u(s,a,M)$*. Since no two approximation units in an approximation set can have the same identifying state-action-MDP triple, we will use* $\boldsymbol{k_a(s,a,M)}$ *to denote* $k_a(u(s,a,M))$*.*

K_a is one of the parameters that will allow us to customize our exploration algorithm. At any point in time, only the active samples participate in the value function produced by our algorithm. K_a can be anything between the two extremes of $\{k\}$ and $\{1, 2, 3, \ldots, k\}$, and as we will see in Chapter 7 different settings offer different tradeoffs.

Definition 5.2.9. *Let* $\boldsymbol{k_m} \geqslant 1$ *be an integer parameter, and* $\boldsymbol{\epsilon_b}$ *be a real value which we will call the* **exploration bonus**. *Let* $u(s,a,M)$ *be an approximation unit for which* $k_a(s,a,M) \geqslant k_m$*. The function* $\boldsymbol{F^{\pi}(Q, u(s,a,M))}$ *is defined as:*

$$\boldsymbol{F^{\pi}(Q, u(s,a,M))} = \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} + median\big\{G^{\pi}(Q, u(s,a,M), 1), \ldots, G^{\pi}(Q, u(s,a,M), k_m)\big\},$$

where

$$\boldsymbol{G^{\pi}(Q, u(s,a,M), j)} = \frac{k_m}{k_a(s,a,M)} \sum_{i=1+(j-1)\frac{k_a(s,a,M)}{k_m}}^{j\frac{k_a(s,a,M)}{k_m}} \left(r_i + \gamma Q(s'_i, \pi(s'_i), M_i) \right),$$

and $(s_i, a_i, r_i, s'_i, \mathcal{A}_{s'_i}, M_i)$ *is the* i*-th sample pointed to by* $u(s,a,M)$*. We will use* $\boldsymbol{F(Q, u(s,a,M))}$ *to denote* $F^{\pi^Q}(Q, u(s,a,M))$*.*

F^π splits the active samples of an approximation unit into k_m groups (if $k_m > 1$), computes the average of the sample values in each group, and returns the median of the averages.

Definition 5.2.10. *If U contains at least one approximation unit $u(s, a, M)$ for which $k_a(s, a, M) > 0$, the function $\boldsymbol{N(U, s, a, M)}$ returns the identifying state-action-MDP triple of the approximation unit $u(\bar{s}, \bar{a}, \bar{M}) \in U$ with $k_a(\bar{s}, \bar{a}, \bar{M}) > 0$ such that*

$$d(s, a, M, \bar{s}, \bar{a}, \bar{M}, d_{known}) + \frac{\epsilon_b}{\sqrt{k_a(\bar{s}, \bar{a}, \bar{M})}}$$

is minimized. Ties are broken by an arbitrary deterministic function.

Instead of returning the identifying state-action-MDP triple of the nearest approximation unit with respect to $d(s, a, M, \bar{s}, \bar{a}, \bar{M}, d_{known})$, $N(U, s, a, M)$ takes into account the number of active samples of every approximation unit.

Definition 5.2.11. *For state-action (s, a) in MDP M, the approximate optimistic Bellman operator $\boldsymbol{\tilde{B}^\pi}$ for policy π is defined as:*

$$\boldsymbol{\tilde{B}^\pi Q(s, a, M)} = \min\big\{Q_{\max}, F^\pi(Q, u(N(U, s, a, M))) + d(s, a, M, N(U, s, a, M), d_{known})\big\}.$$

We will use $\boldsymbol{\tilde{B}Q(s, a, M)}$ to denote $\tilde{B}^{\pi^Q} Q(s, a, M)$. When U is the empty set, $\tilde{B}^\pi Q(s, a, M) = Q_{\max}$.

The approximate optimistic Bellman operator is used in the inner loop of our exploration algorithm.

Definition 5.2.12. *The value function $\boldsymbol{Q_U}$ of an approximation set U is defined as:*

$$\boldsymbol{Q_U(s, a, M)} = \min\{Q_{\max}, u_v(N(U, s, a, M)) + d(s, a, M, N(U, s, a, M), d_{known})\}.$$

Where $u_v(N(U, s, a, M))$ is the value stored in approximation unit $u(N(U, s, a, M))$. If U is the empty set $Q_U(s, a, M) = Q_{\max}$ $\forall(s, a, M)$.

Definition 5.2.13. $\boldsymbol{\epsilon_c} \geqslant 0$ *is the minimal non-negative constant satisfying:*

$$\forall(s, a, M, \bar{s}, \bar{a}, \bar{M}, \pi, Q_{\tilde{U}}), |B^{\pi} Q_{\tilde{U}}(s, a, M) - B^{\pi} Q_{\tilde{U}}(\bar{s}, \bar{a}, \bar{M})| \leqslant$$
$$\boldsymbol{\epsilon_c} + d(s, a, M, \bar{s}, \bar{a}, \bar{M}, d_{known}).$$

Note that while d_{known} is a user-defined parameter, algorithm execution does not require ϵ_c to be known.

Definition 5.2.13 describes one of the most important concepts this work is based on. It will allow us to show that as long as $B^{\pi} Q_{\tilde{U}}(s, a, M)$ is similar for nearby state-action-MDP triples, our algorithm will perform well. Contrary to previous work (Pazis and Parr, 2013), we can guarantee this even when $B^{\pi} Q_{\tilde{U}}(s, a, M)$ is not Lipschitz continuous, or when our algorithm is learning from multiple MDPs rather than a single MDP.

Definition 5.2.14. σ *is the minimal non-negative constant satisfying:*

$$\forall(s, a, M, \pi, Q_{\tilde{U}}), \sqrt{var\left[B^{\pi} Q_{\tilde{U}}(s, a, M)\right]} \leqslant \sigma,$$

where the variance is over the possible next states (s, a, M) *can transition to.*

5.3 Useful lemmas

Lemma 5.3.1. *Let* t_i *for* $i = 0 \rightarrow l$ *be the outcomes of independent (but not necessarily identically distributed) random variables in* $\{0, 1\}$*, with* $P(t_i = 1) \geqslant p_i$*. If* $\frac{2}{m} \ln \frac{1}{\delta} < 1$ *and:*

$$\sum_{i=0}^{l} p_i \geqslant \frac{m}{1 - \sqrt{\frac{2}{m} \ln \frac{1}{\delta}}},$$

then $\sum_{i=0}^{l} t_i \geqslant m$ *with probability at least* $1 - \delta$*.*

Proof. Let $\sum_{i=0}^{l} p_i = \mu$. From Chernoff's bound we have that the probability that

$\sum_{i=0}^{l} t_i$ is less than $(1-\epsilon)\mu$ for $0 < \epsilon < 1$ is upper bounded by:

$$\delta = P\left(\sum_{i=0}^{l} t_i \leqslant (1-\epsilon)\mu\right) \leqslant e^{\frac{-\epsilon^2\mu}{2}}. \tag{5.1}$$

Setting $(1-\epsilon)\mu = m$ we have that $\mu = \frac{m}{1-\epsilon}$. Substituting into equation 5.1 above and taking logarithms we have:

$$\begin{aligned}
\ln\delta \leqslant & \frac{-\epsilon^2 \frac{m}{1-\epsilon}}{2} \Rightarrow \\
\frac{\epsilon^2}{1-\epsilon} \leqslant & \frac{2}{m}\ln\frac{1}{\delta} \Rightarrow \\
\epsilon^2 < & \frac{2}{m}\ln\frac{1}{\delta} \Rightarrow \\
\epsilon < & \sqrt{\frac{2}{m}\ln\frac{1}{\delta}}.
\end{aligned}$$

Substituting into $\mu = \frac{m}{1-\epsilon}$ concludes our proof. □

Lemma 5.3.1 is an improvement over Li's Lemma 56 (Li, 2009) in two ways: While Li's lemma requires that $\mu > 2m$ for all $\delta < 1$, for realistically large values of m lemma 5.3.1 yields values much closer to m. As an example, for $\delta = 10^{-9}$ and $k = 10^3, 10^6$ and 10^9 our bound yields $1.2556m, 1.0065m$ and $1.0002m$ respectively. More importantly, our bound allows for the success probability of each trial to be different, which will allow us to prove TCE PAC bounds, rather than just bounds on the number of significantly suboptimal steps.

Versions of Lemma 5.3.2 below have appeared in the literature before. It is included here for completeness:

Lemma 5.3.2. *Let $\hat{B}$ be a γ-contraction with fixed point $\hat{Q}$, and Q the output of*

$$\frac{\ln\frac{\epsilon}{(Q_{\max}-Q_{\min})}}{\ln\gamma}$$

iterations[4] *of value iteration using* $\hat{B}$. *Then if* $Q_{\min} \leqslant \hat{Q}(s,a,M) \leqslant Q_{\max}$ *and* $Q_{\min} \leqslant Q_0(s,a,M) \leqslant Q_{\max}\ \forall\ (s,a,M)$, *where* $Q_0(s,a,M)$ *is the initial value for* (s,a,M)*:*

$$-\epsilon \leqslant Q(s,a,M) - \hat{B}Q(s,a,M) \leqslant \epsilon\ \forall (s,a,M).$$

Proof. If $\gamma = 0$ $\hat{B}Q_0(s,a,M) = \hat{Q}(s,a,M)\ \forall (s,a,M)$. Otherwise, we have that:

$$||\hat{Q} - Q_0||_\infty \leqslant Q_{\max} - Q_{\min}.$$

Let Q_n be the value function at the n-th step of value iteration. Since $\hat{B}$ is a γ-contraction with fixed point $\hat{Q}$, for $n \geqslant 1$:

$$\begin{aligned}
||\hat{Q} - \hat{B}Q_n||_\infty &= ||\hat{B}\hat{Q} - \hat{B}Q_n||_\infty \\
&\leqslant \gamma||\hat{Q} - Q_{n-1}||_\infty \\
&= \gamma||\hat{Q} - \hat{B}Q_{n-2}||_\infty \\
&\leqslant \gamma^2||\hat{Q} - Q_{n-2}||_\infty \\
&\dots \\
&\leqslant \gamma^n||\hat{Q} - Q_0||_\infty \Rightarrow \\
||\hat{Q} - \hat{B}Q_n||_\infty &\leqslant \gamma^n(Q_{\max} - Q_{\min}).
\end{aligned}$$

We want

$$\epsilon \leqslant \gamma^n(Q_{\max} - Q_{\min}),$$

which after redistributing and taking the γ logarithm becomes:

$$\begin{aligned}
n &\leqslant \log_\gamma \frac{\epsilon}{Q_{\max} - Q_{\min}} \\
&= \frac{\ln \frac{\epsilon}{(Q_{\max} - Q_{\min})}}{\ln \gamma}.
\end{aligned}$$

□

[4] Note that $\frac{\ln \frac{\epsilon}{(Q_{\max} - Q_{\min})}}{\ln \gamma} \leqslant \frac{1}{1-\gamma} \ln \frac{Q_{\max} - Q_{\min}}{\epsilon}$.

6

Concurrent PAC Optimal Exploration: A Flexible Algorithm

Given the definitions in Chapter 5 we are now ready to present our exploration algorithm. The results presented in this chapter hold for any choice of K_a, ϵ_b, and k_m. As we will see in our analysis in Chapter 7, we can customize the properties of our algorithm by varying K_a, ϵ_b, and k_m.

6.1 The concurrent exploration algorithm

When exploring in k_p concurrent MDPs, our algorithm consists of k_p (one per MDP) instantiations of the policy execution process (Algorithm 1), and a single learner[1] (Algorithm 2). Policy execution processes communicate with the learner by adding samples to a queue for inclusion in the sample set, and the learner communicates with the policy execution processes by publishing $\tilde{\boldsymbol{U}}$, a set of degenerate[2] approximation

[1] Our results readily generalize to the case where we have multiple learners, even in the case where the order in which samples are processed is different for each learner. For simplicity of exposition we will assume that only a single learner exists.

[2] Since only values of published approximation units are ever accessed, approximation units in $\tilde{U}$ do not contain sample pointers.

units that define the approximate value function $\boldsymbol{Q_{\tilde{U}}}$.

Algorithm 1 Policy execution (k_p instantiations, one for each concurrent MDP):

Initialize $\tilde{U}$ to the empty set.
loop
  From state s in MDP M, using the latest available published $\tilde{U}$:
    Perform action $a = \arg\sup_{\tilde{a}} Q_{\tilde{U}}(s, \tilde{a}, M)$
    Receive reward r
    Transition to state s', and observe $\mathcal{A}_{s'}$.
  Submit $(s, a, r, s', \mathcal{A}_{s'}, M)$ to the inclusion candidate queue.
end loop

Algorithm 2 Learner

Initialize U to the empty set, publish a snapshot of U as $\tilde{U}$, and set update to false.
loop
  Set U_{old} to U.
  while the inclusion candidate queue is not empty **do**
    Pop $(s, a, r, s', \mathcal{A}_{s'}, M)$ from the queue.
    if $d(s, a, M, \bar{s}, \bar{a}, \bar{M}) > d_{known}\ \forall\ u(\bar{s}, \bar{a}, \bar{M}) \in U$ **then**
      Add a new approximation unit $u(s, a, M)$ to U, initialize it with up to k pointers to any $(s_i, a_i, r_i, s_i', \mathcal{A}_{s_i'}, M_i)$ samples in the sample set for which $d(s, a, M, s_i, a_i, M_i) \leqslant d_{known}$, and set update to true.
    end if
    if $d(s, a, M, \bar{s}, \bar{a}, \bar{M}) \leqslant d_{known}$ for some $u(\bar{s}, \bar{a}, \bar{M}) \in U$ containing fewer than k samples **then**
      Add $(s, a, r, s', \mathcal{A}_{s'}, M)$ to the sample set, and a pointer to each $u(\bar{s}, \bar{a}, \bar{M}) \in U$ containing fewer than k samples for which $d(s, a, M, \bar{s}, \bar{a}, \bar{M}) \leqslant d_{known}$. If this increases the number of active samples for at least one approximation unit, set update to true.
    end if
  end while
  if update is true **then**
    Set Q_U to $\tilde{B}Q_{U_{old}}$.
    Publish a snapshot of U as $\tilde{U}$.
    if $-\boldsymbol{\epsilon_a} \leqslant Q_{U_{old}}(s, a, M) - Q_U(s, a, M) \leqslant \boldsymbol{\epsilon_a}\ \forall\ u(s, a, M) \in U$ **then**
      Set update to false.
    end if
  end if
end loop

Algorithm 1 is a greedy policy execution algorithm. At every step, it uses the latest available published $\tilde{U}$, selects the greedy action, collects a sample, and submits it to the learner for processing.

At its core, Algorithm 2 is based on value iteration. Samples submitted by the policy execution processes are processed sequentially in lines 4 through 12. If a sample is more than d_{known} away from the nearest approximation unit in the approximation set, a new approximation unit is added. If the sample is within d_{known} of an approximation unit that has fewer than k samples, the sample is added to the sample set, and a pointer to the sample is added to all approximation units within d_{known} distance that have fewer than k samples. Lines 13 through 19 perform a single step of value iteration and publish the updated value function. They are repeated until the one step Bellman error of Q_U drops below ϵ_a.

6.2 Implementation considerations

There are couple of implementation issues we should elaborate on:

- Setting Q_U in line 14 of the learner is performed by setting the values of each $u(s, a, M) \in U$ for which $k_a(s, a, M) > 0$ to $F(Q_{U_{old}}, u(s, a, M))$.

- Computing $F(Q_{U_{old}}, u(s, a, M))$ does not require recomputing $N(U, s', a', M)$. By maintaining $|\mathcal{A}|$ pointers to approximation units per sample (one for each $a' \in A_{s'}$), $F(Q_{U_{old}}, u(s, a, M))$ can be computed efficiently. To initialize the pointers for a particular sample, $N(U, s', a', M)$ needs to be called up to $|\mathcal{A}|$ times (once for each $a' \in A_{s'}$) when the sample is added to the sample set, and the pointers need to be updated every time the number of active samples for an approximation unit changes (at a much lower cost than recomputing $N(U, s', a', M)$).

6.3 Concurrency, time, and delay models

We will keep our definitions of concurrency, timestep, and value function update delays general, so as to cover as many real-world situations as possible.

Definition 6.3.1. $\boldsymbol{k_p \geqslant 1}$ *denotes the maximum number of MDPs we are exploring in parallel.*

Definition 6.3.2. Timestep $\boldsymbol{t_i}$ *of MDP* M_i *is defined as the* t_i*-th action our policy takes in MDP* M_i*.*

Definition 6.3.3. Global time $\boldsymbol{t_g}$ *is a measure of time that is common between all MDPs.*

We do not assume that actions in different MDPs are synchronized, that they happen at a constant rate with respect to global time, or that all MDPs have the same start time. For example, it could be the case that in global time $[t_g(s), t_g(f)]$ the policy at MDP M_1 takes actions $2, 3$, the policy at MDP M_2 takes actions $7, 8, 9, 10$, while no actions have been taken yet in MDPs M_3 and M_4.

Assumption 6.3.4. *For ease of exposition we assume that samples are processed by the learner in the same order (with respect to global time) as the actions that generate them. This assumption can be easily removed, but it makes exposition significantly simpler.*

Definition 6.3.5. *Let action* (s, a) *be taken at MDP* M_i *at global time* $t_g(s)$ *generating sample* $(s, a, r, s', \mathcal{A}_{s'}, M_i)$. $\boldsymbol{D_{(s,a,r,s',\mathcal{A}_{s'}M_i),M_j}}$ *is the delay for sample* $(s, a, r, s', \mathcal{A}_{s'}, M_i)$ *with respect to MDP* M_j*. If processing* $(s, a, r, s', \mathcal{A}_{s'}, M_i)$ *does not cause the update flag of the algorithm to be set,* $D_{(s,a,r,s',\mathcal{A}_{s'},M_i),M_j} = 0$ *for all* M_j*. Otherwise* $D_{(s,a,r,s',\mathcal{A}_{s'},M_i),M_j}$ *is the number of actions taken in MDP* M_j *in* $[t_g(s), t_g(u_j))$*, where* $t_g(u_j)$ *is the time* M_j *receives an updated value function* Q_U *for which* $-\boldsymbol{\epsilon_a} \leqslant Q_{U_{old}}(s, a, M) - Q_U(s, a, M) \leqslant \boldsymbol{\epsilon_a}\ \forall\ u(s, a, M) \in U$ *on line 16 of Algorithm 2 after* $(s, a, r, s', \mathcal{A}_{s'}, M_i)$ *has been processed. Every action taken in MDP* M_j *during* $[t_g(s), t_g(u_j))$ *is called a* **delay step** *for MDP* M_j*.*

6.4 Space and time complexity

Lemma 6.4.1. *The space complexity of Algorithm 1 is:*

$$O\left(\mathcal{N}_{\mathcal{SAM}}(d_{known})\right).$$

per concurrent MDP.

Proof. Algorithm 1 only needs access to the value and number of active samples of each approximation unit, and from the definition of the covering number and Algorithm 2 we have that at most $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ approximation units can be added. □

Lemma 6.4.2. *The space complexity of Algorithm 2 is:*

$$O\left(k|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})\right).$$

Proof. From the definition of the covering number we have that at most $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ approximation units can be added. From the definition of an approximation unit and Algorithm 2 we have that at most k samples can be added per approximation unit, and each sample requires at most $O(|\mathcal{A}|)$ space. □

Lemma 6.4.3. *$\tilde{B}$ is a γ-contraction in maximum norm.*

Proof. Suppose $||Q_1 - Q_2||_\infty = \epsilon$. For any (s, a, M) we have:

$$\begin{aligned}
\tilde{B}Q_1(s,a,M) = \; & \min\big\{Q_{\max}, F(Q_1, N(U,s,a,M))+ \\
& \qquad d(s,a,M,N(U,s,a,M),d_{known})\big\} \\
\leqslant \; & \min\big\{Q_{\max}, F(Q_2, N(U,s,a,M)) + \gamma\epsilon+ \\
& \qquad d(s,a,M,N(U,s,a,M),d_{known})\big\} \\
\leqslant \; & \gamma\epsilon + \min\big\{Q_{\max}, F(Q_2, N(U,s,a,M))+ \\
& \qquad d(s,a,M,N(U,s,a,M),d_{known})\big\} \\
= \; & \gamma\epsilon + \tilde{B}Q_2(s,a,M)
\end{aligned}$$

$$\Rightarrow \tilde{B}Q_1(s,a,M) \leqslant \gamma\epsilon + \tilde{B}Q_2(s,a,M).$$

Similarly we have that $\tilde{B}Q_2(s,a,M) \leqslant \gamma\epsilon + \tilde{B}Q_1(s,a,M)$ which completes our proof.

□

Lemma 6.4.4. *The per step computational complexity of Algorithm 1 is bounded above by:*

$$O\left(|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})\right).$$

Proof. A naive search for the nearest approximation unit of each of the at most $|\mathcal{A}|$ actions needs to perform at most $O\left(\mathcal{N}_{\mathcal{SAM}}(d_{known})\right)$ operations. □

Since every step of Algorithm 1 results in a sample being processed by Algorithm 2, we will be bounding the per sample computational complexity of Algorithm 2:

Lemma 6.4.5. *Let c be the maximum number of approximation units a single sample can be added to*[3]*. The per sample computational complexity of Algorithm 2 is bounded above by:*

$$O\left(ck|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known}) + \frac{k|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})}{\ln\gamma}\ln\frac{\epsilon}{Q_{\max}}\right).$$

Proof. Lines 4 through 12 will be performed only once per sample. Line 6 requires at most $O(\mathcal{N}_{\mathcal{SAM}}(d_{known}))$ operations. Line 7 requires at most $O(k\mathcal{N}_{\mathcal{SAM}}(d_{known}))$ operations. Line 9 can reuse the result of line 6 and only check if the approximation unit added in line 7 (if any) has k samples. Line 10 requires at most $O(\mathcal{N}_{\mathcal{SAM}}(d_{known}))$ operations.

Given a pointer to $N(U,s',a',M)$ for every next state action MDP triple, line 14 requires at most $O\left(k|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})\right)$ operations($O\left(k|\mathcal{A}|\right)$ for each approximation unit), and since $\tilde{B}$ is a γ-contraction in maximum norm (Lemma 6.4.3), it will

[3] c will depend on the dimensionality of the state-action-MDP space. For domains that are big enough to be interesting it will be significantly smaller than other quantities of interest.

be performed at most $\frac{\ln \frac{\epsilon}{(Q_{\max})}}{\ln \gamma}$ times per sample[4] (Lemma 5.3.2), for a total cost of $O\left(\frac{k|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})}{\ln \gamma} \ln \frac{\epsilon_a}{Q_{\max}}\right)$. This step dominates the computational cost of the learner.

Lines 14, 15 and 16 require at most $O\left(\mathcal{N}_{\mathcal{SAM}}(d_{known})\right)$ operations each, and will be performed at most $\frac{\ln \frac{\epsilon}{(Q_{\max})}}{\ln \gamma}$ times per sample.

Maintaining a pointer to $N(U, s', a', M)$ for every next state action MDP triple requires the following: 1) Computing $N(U, s', a', M)$ every time a sample is added to the sample set, for a cost of up to $O(|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known}))$ operations per sample. 2) Updating all $N(U, s', a', M)$ pointers every time the number of active samples for an approximation unit changes: There can exist at most $k|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known})$ pointers in the sample set, and a single sample can affect the number of active samples of at most c approximation units. Each pointer can be updated against a particular approximation unit in constant time, thus the cost of maintaining the pointers is at most $O(ck|\mathcal{A}|\mathcal{N}_{\mathcal{SAM}}(d_{known}))$ operations per sample. □

[4] Note that the per sample cost does not increase even if multiple samples enter through step 3 before publishing $\tilde{U}$.

7

PAC Analysis

This chapter analyses TCE for different settings of algorithm 2. In all cases, our analysis strategy follows a number of discrete steps:

- In the first step we bound the probability that an individual approximation unit, belonging to a particular published $\tilde{U}$, will have Bellman error of unacceptably high magnitude with respect to the number of active samples it contains.

- In the second step we use the union bound to bound the probability that there exists at least one approximation in $\tilde{U}$ for some published $\tilde{U}$, with Bellman error of unacceptably high magnitude with respect to the number of active samples it contains.

- In the third step we bound the probability that any (s, a, M) will have Bellman error of unacceptably high magnitude.

- In the fourth step we bound the number of times approximation units with less than k active samples can be encountered.

- Finally, by combining the results from the previous steps with the theory developed in the previous chapters we are able to prove TCE bounds.

In the following $Q_{\tilde{U}}$ will to refer to the approximate value function that our algorithm publishes. Much of our analysis is based on comparing the effects of applying different Bellman operators to $Q_{\tilde{U}}$. This is an analysis tool, a hypothetical question of what would happen if we applied some Bellman operator to $Q_{\tilde{U}}$, rather than part of the algorithm.

7.1 $k_m = 1$ specific lemmas

When $k_m = 1$, Lemma 7.1.1 bounds the difference between $F^\pi(Q_{\tilde{U}}, u(s,a,M))$ and its expected value as a function of $k_a(s,a,M)$. We will use this lemma in Theorems 7.4.1 and 7.4.2 to bound the Bellman error of individual approximation units.

Lemma 7.1.1. *Let $k_m = 1$. Then for a fixed $u(s,a,M)$ belonging to a fixed $\tilde{U}$:*

$$P\left(F^\pi(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^\pi(Q_{\tilde{U}}, u(s,a,M))\right] \leqslant -\epsilon_0\right) \leqslant e^{-\frac{2\epsilon_0^2 k_a(s,a,M)}{\hat{Q}_{max}^2}},$$

and:

$$P\left(F^\pi(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^\pi(Q_{\tilde{U}}, u(s,a,M))\right] \geqslant \epsilon_0\right) \leqslant e^{-\frac{2\epsilon_0^2 k_a(s,a,M)}{\hat{Q}_{max}^2}}.$$

Proof. If $k_a(s,a,M) = 0$ then $e^{-\frac{2\epsilon_0^2 k_a(s,a,M)}{\hat{Q}_{max}^2}} = 1$ and the above is trivially true. Otherwise, let Y be the set of $k_a(s,a,M)$ samples used by $F^\pi(\tilde{Q}, u(s,a,M))$ at (s,a,M) and define $f(x_1, \ldots x_{k_a}) = F^\pi(Q_{\tilde{U}}, u(s,a,M))$, where $x_1, \ldots x_{k_a}$ are realizations of independent (from the Markov property) variables whose outcomes are possible next states, one for each state-action-MDP in Y. The outcomes of the variables (which is where the Markov property ensures independence) are the next states the transitions lead to, not the state-action-MDP triples the samples originate from. The

state-action-MDP triples the samples originate from are fixed with respect to f, and no assumptions are made about their distribution. Since $\tilde{U}$ is fixed, $Q_{\tilde{U}}$ is fixed with respect to f (we are examining the effects of a single application of $F^{\pi}(Q_{\tilde{U}}, u(s,a,M))$ to the fixed function $Q_{\tilde{U}}$, while varying the next-states that samples in $u(s,a,M)$ land on). $\forall\ i\ \in [1, k_a(s,a,M)]$:

$$\sup_{x_1,\ldots x_k,\hat{x}_i} |f(x_1,\ldots x_{k_a}) - f(x_1,\ldots,x_{i-1}\hat{x}_i, x_{i+1}\ldots x_{k_a(s,a,M)})| = c_i \leqslant \frac{\hat{Q}_{max}}{k_a(s,a,M)},$$

and

$$\sum_{i=1}^{k_a(s,a,M)} (c_i)^2 \leqslant k_a(s,a,M)\frac{\hat{Q}_{max}^2}{k_a(s,a,M)^2} = \frac{\hat{Q}_{max}^2}{k_a(s,a,M)}.$$

From McDiarmid's inequality we have:

$$P\left(F^{\pi}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi}(Q_{\tilde{U}}, u(s,a,M))\right] \leqslant -\epsilon_0\right)$$

$$\leqslant P\left(f(x_1,\ldots x_{k_a}) - \mathrm{E}\left[f(x_1,\ldots x_{k_a})\right] \leqslant -\epsilon_0\right)$$

$$\leqslant e^{-\frac{2\epsilon_0^2}{\sum_{i=1}^{k_a(s,a,M)}(c_i^2)}}$$

$$\leqslant e^{-\frac{2\epsilon_0^2 k_a(s,a,M)}{\hat{Q}_{max}^2}},$$

and:

$$P\left(F^{\pi}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi}(Q_{\tilde{U}}, u(s,a,M))\right] \geqslant \epsilon_0\right)$$

$$\leqslant P\left(f(x_1,\ldots x_{k_a}) - \mathrm{E}\left[f(x_1,\ldots x_{k_a})\right] \geqslant \epsilon_0\right)$$

$$\leqslant e^{-\frac{2\epsilon_0^2}{\sum_{i=1}^{k_a(s,a,M)}(c_i^2)}}$$

$$\leqslant e^{-\frac{2\epsilon_0^2 k_a(s,a,M)}{\hat{Q}_{max}^2}}.$$

□

7.2 Median trick specific lemmas

When $k_m > 1$ our algorithm uses what is known as the median trick (Alon et al., 1999). When using the median trick, instead of taking the average over all active samples in an approximation unit, we split them into k_m groups, take the average over each group, and then select the median of the averages. This allows our bounds to scale with the variance of the Bellman operator, rather than the range of values the Bellman operator can return.

Lemma 7.2.1. *Let σ be defined as in Definition* 5.2.14*, and $\epsilon_b = \sigma\sqrt{4k_m}$. For a fixed (s, a, M), $\tilde{U}$ and some fixed $j \in [1, k_m]$:*

$$P\left(G^\pi(Q_{\tilde{U}}, u(s,a,M), j) - B^\pi Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right) \leqslant \frac{1}{5},$$

and:

$$P\left(G^\pi(Q_{\tilde{U}}, u(s,a,M), j) - B^\pi Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right) \leqslant \frac{1}{5}.$$

Proof. From Definition 5.2.13 we have that:

$$B^\pi Q_{\tilde{U}}(s,a,M) - \epsilon_c \leqslant \mathrm{E}\left[G^\pi(Q_{\tilde{U}}, u(s,a,M), j)\right],$$

and:

$$\mathrm{E}\left[G^\pi(Q_{\tilde{U}}, u(s,a,M), j)\right] \leqslant B^\pi Q_{\tilde{U}}(s,a,M) + \epsilon_c,$$

where the expectation is over the next-states that samples in $u(s, a, M)$ used by G^π land on.

Let Y be the set of $\frac{k_a(s,a,M)}{k_m}$ samples used by $G^\pi(Q_{\tilde{U}}, u(s,a,M), j)$ at (s, a, M). Define $Z_1, \ldots Z_{\frac{k_a(s,a,M)}{k_m}}$ to be random variables, one for each sample in Y. The distribution of Z_i is the distribution of possible values that $r_i + \gamma \max_{a'} Q_{\tilde{U}}(s'_i, a', M_i)$ can take, given (s_i, a_i, M_i), $Q_{\tilde{U}}$ and the transition model of M_i, where (s_i, a_i, M_i)

is the state-action-MDP triple of the i-th sample in Y. From the Markov property we have that $Z_1, \ldots Z_{\frac{k_a(s,a,M)}{k_m}}$ are independent random variables[1]. From Definition 5.2.14 we have that the variance of $Z_i \ \forall \ i$ is upper bounded by σ^2. Define $X = \sum_{i=1}^{\frac{k_a(s,a,M)}{k_m}} \left(\frac{k_m}{k_a(s,a,M)} Z_i \right)$. We have that $var[X] \leqslant \frac{\sigma^2 k_m}{k_a(s,a,M)}$.

From Cantelli's inequality we have:

$$
\begin{aligned}
& P\left(G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi} Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right) \\
& \leqslant P\left(G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) - \mathrm{E}\left[G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) \right] \leqslant -\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right) \\
& \leqslant \frac{\frac{\sigma^2 k_m}{k_a(s,a,M)}}{\frac{\sigma^2 k_m}{k_a(s,a,M)} + \left(\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right)^2} \\
& = \frac{\frac{\sigma^2 k_m}{k_a(s,a,M)}}{\frac{\sigma^2 k_m}{k_a(s,a,M)} + \frac{4\sigma^2 k_m}{k_a(s,a,M)}} \\
& = \frac{1}{5},
\end{aligned}
$$

and:

$$
\begin{aligned}
& P\left(G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi} Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right) \\
& \leqslant P\left(G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) - \mathrm{E}\left[G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j) \right] \geqslant \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right) \\
& \leqslant \frac{\frac{\sigma^2 k_m}{k_a(s,a,M)}}{\frac{\sigma^2 k_m}{k_a(s,a,M)} + \left(\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} \right)^2}
\end{aligned}
$$

[1] The state-action-MDP triples the samples originate from as well as $Q_{\tilde{U}}$ and the transition model of M_i are fixed with respect to Z_i, and no assumptions are made about their distribution. The only source of randomness is the the transition model of M_i.

$$= \frac{\frac{\sigma^2 k_m}{k_a(s,a,M)}}{\frac{\sigma^2 k_m}{k_a(s,a,M)} + \frac{4\sigma^2 k_m}{k_a(s,a,M)}}$$

$$= \frac{1}{5}.$$

□

Based on Lemma 7.2.1 we can now bound the probability that an individual approximation unit belonging to a particular published $\tilde{U}$ will have Bellman error of unacceptably high magnitude:

Lemma 7.2.2. *Let σ be defined as in Definition* 5.2.14, $\epsilon_b = \sigma\sqrt{4k_m}$, *and* $k_m = \left\lceil 5.6 \ln \frac{4\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta} \right\rceil$. *When $u(s,a,M)$ contains $k_a(s,a,M)$ active samples and for a fixed $\tilde{U}$:*

$$P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c\right) \leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2},$$

and

$$P\left(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + 2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right)$$
$$\leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}.$$

Proof. Let Y be the set of $k_a(s,a,M)$ samples used by $F^{\pi}(\tilde{Q}, u(s,a,M))$ at (s,a,M). Define $Z_1, \ldots Z_{k_a(s,a,M)}$ to be random variables, one for each sample in Y. The distribution of Z_i is the distribution of next states s_i', given (s_i, a_i, M_i), $Q_{\tilde{U}}$ and the transition model of M_i, where (s_i, a_i, M_i) is the state-action-MDP triple of the i-th sample in Y. From the Markov property, we have that $Z_1, \ldots Z_{k_a(s,a,M)}$ are independent random variables (similarly to Lemma 7.2.1). Let x_j be a realization

of X_j, where X_j's distribution is the joint distribution of all Z_i corresponding to samples that participate in $G^{\pi}(Q_{\tilde{U}}, u(s,a,M), j)$.

We define $f^{\pi^*}(x_1, \dots x_{k_m})$ to be the function that counts the number of j's such that:

$$G^{\pi^*}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}},$$

and $f^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}(x_1, \dots x_{k_m})$ to be the function that counts the number of j's such that:

$$G^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}.$$

From Lemma 7.2.1 we have that:

$$E[f^{\pi^*}(x_1, \dots x_{k_m})] \leqslant \frac{k_m}{5},$$

and:

$$E[f^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}(x_1, \dots x_{k_m})] \leqslant \frac{k_m}{5}.$$

$\forall\ i\ \in [1, k_m]$:

$$\sup_{x_1, \dots x_k, \hat{x}_i} |f^{\pi^*}(x_1, \dots x_{k_a}) - f^{\pi^*}(x_1, \dots, x_{i-1}\hat{x}_i, x_{i+1} \dots x_{k_a(s,a,M)})| \leqslant 1,$$

and:

$$\sup_{x_1, \dots x_k, \hat{x}_i} |f^{\pi^{Q_{\tilde{U}}}}(x_1, \dots x_{k_a}) - f^{\pi^{Q_{\tilde{U}}}}(x_1, \dots, x_{i-1}\hat{x}_i, x_{i+1} \dots x_{k_a(s,a,M)})| \leqslant 1.$$

From McDiarmid's inequality we have:

$$P\left(f^{\pi^*}(x_1, \dots x_{k_m}) \geqslant \frac{k_m}{2}\right) \leqslant P\left(f^{\pi^*}(x_1, \dots x_{k_m}) - E[f^{\pi^*}(x_1, \dots x_{k_m})] \geqslant \frac{3k_m}{10}\right)$$

$$\leqslant e^{-\frac{2\left(\frac{3k_m}{10}\right)^2}{k_m}}$$

$$= e^{-\frac{9k_m^2}{50}}$$

$$\leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2},$$

and:

$$P\left(f^{\pi^{Q_{\tilde{U}}}}(x_1,\ldots x_{k_m}) \geqslant \frac{k_m}{2}\right) \leqslant P\left(f^{\pi^{Q_{\tilde{U}}}}(x_1,\ldots x_{k_m}) - E[f^{\pi^{Q_{\tilde{U}}}}(x_1,\ldots x_{k_m})] \geqslant \frac{3k_m}{10}\right)$$

$$\leqslant e^{-\frac{2\left(\frac{3k_m}{10}\right)^2}{k_m}}$$

$$= e^{-\frac{9k_m^2}{50}}$$

$$\leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}.$$

Since the probability that

$$G^{\pi^*}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}$$

for at least $\frac{k_m}{2}$ j's is upper bounded by $\frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$, and the probability that:

$$G^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M), j) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}$$

for at least $\frac{k_m}{2}$ j's is upper bounded by $\frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$, the result follows from Definition 5.2.9. □

7.3 General lemmas

In the following, $\epsilon^-\left(\tilde{U}, u(s,a,M)\right)$ and $\epsilon^+\left(\tilde{U}, u(s,a,M)\right)$ denote constants specific to a particular approximation set $\tilde{U}$ and approximation unit $u(s,a,M)$.

Given a bound on the probability that an individual approximation unit has Bellman error of unacceptably high magnitude, Lemma 7.3.1 uses the union bound

to bound the probability that there exists at least one approximation unit in $\tilde{U}$ for some published $\tilde{U}$ with Bellman error of unacceptably hight magnitude during a non-delay step.

Lemma 7.3.1. *If for any particular (fixed) $\tilde{U}$ and $u(s,a,M)$ during a non-delay step:*

$$P\Big(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \epsilon^-\left(\tilde{U}, u(s,a,M)\right)\Big) \leqslant \frac{\delta}{4|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2},$$

and:

$$P\Big(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \epsilon^+\left(\tilde{U}, u(s,a,M)\right)\Big) \leqslant \frac{\delta}{4|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2},$$

the probability that for any published $\tilde{U}$ there exists at least one $u(s,a,M) \in \tilde{U}$ such that:

$$F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \epsilon^-\left(\tilde{U}, u(s,a,M)\right) \tag{7.1}$$

or:

$$F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \epsilon^+\left(\tilde{U}, u(s,a,M)\right) \tag{7.2}$$

during a non-delay step, is upper bounded by $\frac{\delta}{2}$.

Proof. When $\tilde{U}$ is the empty set no approximation units exist, so the above is trivially true. Otherwise, at most $|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})$ distinct non-empty $\tilde{U}$ exist during non-delay steps. Thus, there are at most $2|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2$ ways for at least one of the at most $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ approximation units to fail at least once during non-delay steps ($|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2$ ways each for equation 7.1 or equation 7.2 to be true at least once), each with a probability at most $\frac{\delta}{4|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$. From the

union bound, we have that the probability that for any published $\tilde{U}$ there exists at least one $u(s,a,M) \in \tilde{U}$ such that equation 7.1 or 7.2 is true during a non-delay step, is upper bounded by $\frac{\delta}{2}$. $\square$

Based on Lemma 7.3.1 we can now bound the probability that any (s,a,M) will have Bellman error of unacceptably high magnitude during a non-delay step:

Lemma 7.3.2. *If for any particular (fixed) $\tilde{U}$ and $u(s,a,M)$ during a non-delay step:*

$$P\Big(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c - \epsilon^-\left(\tilde{U}, u(s,a,M)\right)\Big) \leqslant \frac{\delta}{4|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$$

and:

$$P\Big(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + \epsilon^+\left(\tilde{U}, u(s,a,M)\right)\Big) \leqslant \frac{\delta}{4|K_a|\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$$

where $\epsilon^-\left(\tilde{U}, u(s,a,M)\right) \geqslant 0$ and $\epsilon^+\left(\tilde{U}, u(s,a,M)\right)\Big) \geqslant 0$, then:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) > -\epsilon_a - 2\epsilon_c - \epsilon^-\left(\tilde{U}, u(s,a,M)\right)$$

and:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) < \epsilon_a + 2\epsilon_c + \epsilon^+\left(\tilde{U}, u(s,a,M)\right) + 2d(s,a,M,N(\tilde{U},s,a,M),d_{known})$$

$\forall(s,a,M,\tilde{U})$ simultaneously with probability $1-\frac{\delta}{2}$.

Proof. When $\tilde{U}$ is the empty set, $Q_{\tilde{U}}(s,a,M) = Q_{\max}$. Since $B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant Q_{\max}$, $Q_{\tilde{U}}(s,a,M) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \geqslant -\epsilon_a - 2\epsilon_c - \epsilon^-\left(\tilde{U}, u(s,a,M)\right)$. Otherwise, $\forall(s,a,M,\tilde{U})$ with probability $1-\frac{\delta}{2}$:

$B^{\pi^*}Q_{\tilde{U}}(s,a,M) = \min\left\{Q_{\max}, B^{\pi^*}Q_{\tilde{U}}(s,a,M)\right\}$

$$
\begin{aligned}
&\leqslant \min\left\{Q_{\max}, B^{\pi^*}Q_{\tilde{U}}(N(\tilde{U},s,a,M))+\right.\\
&\qquad\qquad \left. d(s,a,M,N(\tilde{U},s,a,M),d_{known})+\epsilon_c\right\}\\
&< \min\left\{Q_{\max}, F^{\pi^*}(Q_{\tilde{U}},u(N(\tilde{U},s,a,M)))+\epsilon_c+\epsilon^-\left(\tilde{U},u(s,a,M)\right)+\right.\\
&\qquad\qquad \left. d(s,a,M,N(\tilde{U},s,a,M),d_{known})+\epsilon_c\right\}\\
&\leqslant \tilde{B}^{\pi^*}Q_{\tilde{U}}(s,a,M)+2\epsilon_c+\epsilon^-\left(\tilde{U},u(s,a,M)\right)\\
&\leqslant \tilde{B}Q_{\tilde{U}}(s,a,M)+2\epsilon_c+\epsilon^-\left(\tilde{U},u(s,a,M)\right)\\
&\leqslant Q_{\tilde{U}}(s,a,M)+\epsilon_a+2\epsilon_c+\epsilon^-\left(\tilde{U},u(s,a,M)\right).
\end{aligned}
$$

In step 1 we used the fact that $B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant Q_{\max}$, in step 2 we used Definition 5.2.13, in step 3 we used Lemma 7.3.1, in step 4 we used Definition 5.2.11, in step 5 we used the fact that $\tilde{B}Q_{\tilde{U}}(s,a,M) \geqslant \tilde{B}^{\pi^*}Q_{\tilde{U}}(s,a,M)$, and in step 6 we used the fact that $-\epsilon_a \leqslant Q_{\tilde{U}}(s,a,M) - \tilde{B}Q_{\tilde{U}}(s,a,M)$.

When no approximation units containing active samples exist, $d(s,a,M,N(\tilde{U},s,a,M))$ is not defined. Otherwise, $\forall(s,a,M,\tilde{U})$ with probability $1-\frac{\delta}{2}$:

$$
\begin{aligned}
&B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M)\\
&\quad= \min\left\{Q_{\max}, B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M)\right\}\\
&\quad\geqslant \min\left\{Q_{\max}, B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(N(\tilde{U},s,a,M))-d(s,a,M,N(\tilde{U},s,a,M),d_{known})-\epsilon_c\right\}\\
&\quad> \min\left\{Q_{\max}, F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}},u(N(\tilde{U},s,a,M)))-\epsilon_c-\epsilon^+\left(\tilde{U},u(s,a,M)\right)-\right.\\
&\qquad\qquad \left.\epsilon_c-d(s,a,M,N(\tilde{U},s,a,M),d_{known})\right\}\\
&\quad\geqslant \min\left\{Q_{\max}, F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}},u(N(\tilde{U},s,a,M)))+d(s,a,M,N(\tilde{U},s,a,M),d_{known})\right\}-\\
&\qquad\qquad 2\epsilon_c-\epsilon^+\left(\tilde{U},u(s,a,M)\right)-2d(s,a,M,N(\tilde{U},s,a,M),d_{known})
\end{aligned}
$$

$$\geqslant \tilde{B}^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s,a,M) - 2\epsilon_c - \epsilon^+\left(\tilde{U}, u(s,a,M)\right) - 2d(s,a,M,N(\tilde{U},s,a,M),d_{known})$$

$$= \tilde{B}Q_{\tilde{U}}(s,a,M) - 2\epsilon_c - \epsilon^+\left(\tilde{U}, u(s,a,M)\right) - 2d(s,a,M,N(\tilde{U},s,a,M),d_{known})$$

$$\geqslant Q_{\tilde{U}}(s,a,M) - \epsilon_a - 2\epsilon_c - \epsilon^+\left(\tilde{U}, u(s,a,M)\right) - 2d(s,a,M,N(\tilde{U},s,a,M),d_{known}).$$

In step 1 we used the fact that $B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s,a,M) \leqslant Q_{\max}$, in step 2 we used Definition 5.2.13, in step 3 we used Lemma 7.3.1, in step 5 we used Definition 5.2.11, in step 6 we used the fact that $\tilde{B}^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s,a,M) = \tilde{B}Q_{\tilde{U}}(s,a,M)$, and in step 7 we used the fact that $Q_{\tilde{U}}(s,a,M) - \tilde{B}Q_{\tilde{U}}(s,a,M) \leqslant \epsilon_a$.

Note that both the first half of Lemma 7.3.1 (used in the fist half of the proof) and the second half (used in the second half of the proof) hold simultaneously with probability $\frac{\delta}{2}$, thus we don't need to take a union bound over the individual probabilities. □

Lemma 7.3.3 bounds the number of times we can encounter approximation units with fewer than k active samples.

Lemma 7.3.3. *Let $(s_{1,j}, s_{2,j}, s_{3,j}, \dots)$ for $j \in \{1, \dots, k_p\}$ be the random paths generated in MDPs $M_1, \dots, M_{k_p}$ respectively on some execution of Algorithm 1. Let $\tilde{U}(t,j)$ be the approximation set used by Algorithm 1 at step t in MDP M_j. Let $\tau(t,j)$ be the number of steps from step t in MDP M_j to the next delay step, or to the first step t' for which $\tilde{U}(t,j) \neq \tilde{U}(t',j)$, whichever comes first. Let $T_H = \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil$ and define $H = \{1, 2, 4, \dots, 2^i\}$ where i is the largest integer such that $2^i \leqslant T_H$. Let k_a^- be the largest value in K_a that is strictly smaller than k_a, or 0 if such a value does not exist. Let $X_{k_a}(t,j)$ be the set of state-action-MDP triples at step t in MDP M_j such that no approximation unit with at least k_a active samples exists within d_{known} distance, and at least one approximation unit with at least k_a^- active samples exists within d_{known} distance ($k_a^- = 0$ covers both the case where an approximation unit with*

0 *active samples exists within* d_{known} *distance, and the case where no approximation units exist within* d_{known} *distance). Define* $p_{h,k_a}(s_{t,j})$ *for* $k_a \in K_a$ *to be Bernoulli random variables that express the following conditional probability: Given the state of the approximation set and sample candidate queue at step* t *for MDP* M_j*, exactly* h *state-action-MDP triples in* $X_{k_a}(t,j)$ *are encountered in MDP* M_j *during the next* $\min\{T_H, \tau(t,j)\}$ *steps. Let* $p^e_{h,k_a}(s_{t,j}) = \sum_{i=h}^{2h-1} p_{i,k_a}(s_{t,j})$*. Finally let* T_j *be the set of non-delay steps (see Definition 6.3.5) for MDP* M_j*. If* $\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil}{k_p \mathcal{N}_{\mathcal{SAM}}(d_{known})} \ln \frac{2|K_a|}{\delta} < 1$ *and* $\mathcal{N}_{\mathcal{SAM}}(d_{known}) \geqslant 2$*, with probability* $1 - \frac{\delta}{2}$*:*

$$\sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (h p^e_{h,k_a}(s_{t,j}))$$

$$< \frac{(k_a - k_a^- + k_p)\left(1 + \log_2 \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil\right) \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2|K_a|}{\delta}}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$\forall\ k_a \in K_a$ *and* $\forall\ h \in H$ *simultaneously.*

Proof. From the Markov property we have that, $p^e_{h,k_a}(s_{t,j})$ variables at least T_H steps apart (of the same or different j) are independent[2]. Define T_i^H for $i \in \{0, 1, \ldots, T_H - 1\}$ to be the (infinite) set of timesteps for which $t \in \{i, i + T_H, i + 2T_H, \ldots\}$.

$k_a - k_a^-$ samples will be added to an approximation unit with k_a^- active samples before it progresses to k_a active samples. Additionally, at most $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ approximation units can be added to the approximation set, and we are exploring in k_p MDPs in parallel. Thus, at most $(k_a - k_a^- + k_p - 1)\mathcal{N}_{\mathcal{SAM}}(d_{known})$ state-action-MDP triples within d_{known} distance of approximation units with k_a^- active samples can be encountered on non-delay steps[3].

[2] Careful readers may notice that what happens at step t affects which variables are selected at future timesteps. This is not a problem. We only care that the *outcomes* of the variables are independent given their selection.

[3] This covers the worst case, which could happen if every time a state-action-MDP triple within

Let us assume that there exists an $i \in \{0, 1, \ldots, T_H - 1\}$ and $h \in H$ such that:

$$\sum_{j=1}^{k_p} \sum_{t \in T_j \cap T_i^H} p^e_{h,k_a}(s_{t,j}) \geqslant \frac{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}{h\left(1 - \sqrt{\frac{2h}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} \ln \frac{2|K_a|}{\delta}}\right)}.$$

From Lemma 5.3.1 it follows that with probability at least $1 - \frac{\delta}{2|K_a|}$, at least $(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})$ state-action-MDP triples within d_{known} distance of approximation units with k_a^- active samples will be encountered on non-delay steps, which is a contradiction. It must therefore be the case that:

$$\sum_{j=1}^{k_p} \sum_{t \in T_j \cap T_i^H} p^e_{h,k_a}(s_{t,j}) < \frac{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}{h\left(1 - \sqrt{\frac{2h}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} \ln \frac{2|K_a|}{\delta}}\right)}$$

with probability at least $1 - \frac{\delta}{2|K_a|}$ for all $i \in \{0, 1, \ldots, T_H - 1\}$ and $h \in H - \{T_H\}$ simultaneously, which implies that:

$$\begin{aligned}
&\sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (h p^e_{h,k_a}(s_{t,j})) \\
&\qquad < \frac{(k_a - k_a^- + k_p)|H|T_H\mathcal{N}_{\mathcal{SAM}}(d_{known})}{1 - \sqrt{\frac{2T_H}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} \ln \frac{2|K_a|}{\delta}}} \\
&\qquad \leqslant \frac{(k_a - k_a^- + k_p)\left(1 + \log_2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil\right)\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2|K_a|}{\delta}}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}
\end{aligned}$$

with probability at least $1 - \frac{\delta}{2|K_a|}$ for all $h \in H$ simultaneously.

From the union bound we have that since k_a can take $|K_a|$ values, with probability $1 - \frac{\delta}{2}$:

$$\sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (h p^e_{h,k_a}(s_{t,j}))$$

---

d_{known} distance of an approximation unit with $k_a - 1$ samples is encountered by one of the MDPs, the remaining $k_p - 1$ MDPs encounter a state-action-MDP triple within d_{known} of the same approximation unit simultaneously.

$$< \frac{(k_a - k_a^- + k_p)\left(1 + \log_2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2|K_a|}{\delta}}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$\forall\ k_a \in K_a$ and $\forall\ h \in H$ simultaneously. □

7.4 TCE bounds

Theorems 7.4.1, 7.4.2, and 7.4.3 decompose errors into the following four sources:

1. ϵ_s is the error caused by the fact that we are using only a finite set of samples (at most k) to estimate the mean, thus we only have an estimate.

2. ϵ_c, a function of d_{known}, is the error caused by the fact that since the space is continuous we cannot have samples for every state-action.

3. ϵ_a is the error caused by the fact that we are only finding an ϵ_a-approximation, rather than the true fixed point of $\tilde{B}$.

4. Finally, $\epsilon_e(t, j)$ is the error caused by the fact that at time t there may exist state-actions in MDP M_j that do not have an approximation unit with k samples within d_{known} distance, and value function updates may not be instantaneous.

7.4.1 Analysis of the simplest settings

We start by analyzing Algorithm 2 in the simplest possible setting: k_m is set to 1, and K_a and ϵ_b are set in such a way that Algorithm 2 waits for an approximation unit to accumulate k samples before using it.

Theorem 7.4.1. *Let $(s_{1,j}, s_{2,j}, s_{3,j}, \dots)$ for $j \in \{1, \dots, k_p\}$ be the random paths generated in MDPs $M_1, \dots, M_{k_p}$ respectively on some execution of Algorithm 1, and $\tilde{\boldsymbol{\pi}}$ be the (non-stationary) policy followed by algorithm 1. Let $\boldsymbol{\epsilon_b} = 0$, $\boldsymbol{k_m} = 1$,*

$\boldsymbol{k} \geqslant \frac{\hat{Q}_{\max}^2}{2\epsilon_s^2(1-\gamma)^2} \ln\left(\frac{4\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}\right)$, $\boldsymbol{K_a} = \{k\}$, ϵ_c *be defined as in Definition 5.2.13,* ϵ_a *be defined as in Algorithm 2,* k_p *be defined as in Definition 6.3.1, and* $\boldsymbol{T_j}$ *be the set of non-delay steps (see Definition 6.3.5) for MDP* M_j*. If* $\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} < 1$ *and* $\mathcal{N}_{\mathcal{SAM}}(d_{known}) \geqslant 2$*, with probability at least* $1-\delta$*, for all* t, j*:*

$$V^{\tilde{\pi}}(s_{t,j}, M_j) \geqslant V^*(s_{t,j}, M_j) - \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} - 3\epsilon_s - \epsilon_e(t,j),$$

where:

$$\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j) = \sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j) + \sum_{j=1}^{k_p}\sum_{t\notin T_j}\epsilon_e(t,j),$$

with:

$$\begin{aligned}
&\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j) \\
&< \frac{(2k+3k_p)\left(1+\log_2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})Q_{\max}}{1-\sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} \\
&\approx \tilde{O}\left(\frac{\left(\frac{\hat{Q}_{\max}^2}{\epsilon_s^2(1-\gamma)^2}+k_p\right)\mathcal{N}_{\mathcal{SAM}}(d_{known})Q_{\max}}{(1-\gamma)}\right)
\end{aligned}$$

and:

$$\sum_{t\notin T_j}\epsilon_e(t,j) \leqslant \left(Q_{\max}\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j}\right),$$

where $D_{i,j}$ *is the delay of the* k*-th sample in the* i*-th approximation unit with respect to MDP* M_j*. Note that the probability of success* $1-\delta$ *holds for all timesteps in all MDPs simultaneously, and* $\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j)$ *is an undiscounted infinite sum. Unlike* $\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)$ *which is a sum over all MDPs,* $\sum_{t\notin T_j}\epsilon_e(t,j)$ *gives a separate bound on the total cost due to value function update delays for each MDP separately.*

Proof. From Definition 5.2.13 we have that:

$$B^{\pi^*}Q_{\tilde{U}}(s,a,M) - \epsilon_c \leqslant \mathrm{E}\left[F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M))\right]$$

and:

$$\mathrm{E}\left[F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M))\right] \leqslant B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) + \epsilon_c,$$

where the expectations are over the next-states that samples in $u(s,a,M)$ used by F^{π^*} and $F^{\pi^{Q_{\tilde{U}}}}$ respectively land on.

From Lemma 7.1.1 and the fact that $|K_a| = 1$ we have:

$$\begin{aligned}
&P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}\tilde{Q}(s,a,M) \leqslant -\epsilon_c - (1-\gamma)\epsilon_s\right) \\
&\qquad \leqslant P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M))\right] \leqslant -(1-\gamma)\epsilon_s\right) \\
&\qquad \leqslant e^{\frac{-2(1-\gamma)^2\epsilon_s^2 k}{\hat{Q}_{\max}^2}} \\
&\qquad = \frac{\delta}{4\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}
\end{aligned}$$

and:

$$\begin{aligned}
&P\left(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + (1-\gamma)\epsilon_s\right) \\
&\qquad \leqslant P\left(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M))\right] \geqslant (1-\gamma)\epsilon_s\right) \\
&\qquad \leqslant e^{\frac{-2(1-\gamma)^2\epsilon_s^2 k}{\hat{Q}_{\max}^2}} \\
&\qquad = \frac{\delta}{4\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}.
\end{aligned}$$

Substituting the above along with $(1-\gamma)\epsilon_s$ for $\epsilon^-\left(\tilde{U}, u(s,a,M)\right)$ and $\epsilon^+\left(\tilde{U}, u(s,a,M)\right)$ in Lemma 7.3.2 we have that with probability at least $1 - \frac{\delta}{2}$:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) > -\epsilon_a - 2\epsilon_c - (1-\gamma)\epsilon_s \tag{7.3}$$

for all $(s, a, M, \tilde{U})$ during non-delay steps, and:

$$Q_{\tilde{U}}(s, a, M) - B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s, a, M) < \epsilon_a + 2\epsilon_c + (1-\gamma)\epsilon_s \tag{7.4}$$

for all $(s, a, M, \tilde{U})$ for which $d(s, a, M, N(s, a, M)) \leqslant d_{known}$ during non-delay steps.

Let T_H, H, $\tilde{U}(t, j)$, $\tau(t, j)$, and $p^e_{h,k_a}(s_{t,j})$ be defined as in Lemma 7.3.3. Even though $\tilde{\pi}$ is non-stationary, it is comprised of stationary segments. Starting from step t in MDP M_j, $\tilde{\pi}$ is equal to $\pi^{Q_{\tilde{U}(t,j)}}$ for at least $\tau(t, j)$ steps. Substituting equations 7.3 and 7.4 into Lemma 3.2.7 we have that with probability at least $1 - \frac{\delta}{2}$, for all M_j and $t \in T_j$:

$$V^*(s_{t,j}, M_j) - V^{\tilde{\pi}}(s_{t,j}, M_j) \leqslant \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} + 3\epsilon_s + \epsilon_e(t, j),$$

where

$$\epsilon_e(t, j) = \left(\gamma^{\tau(t,j)} + 2\sum_{h \in H}(hp^e_{h,k_a}(s_{t,j}))\right) Q_{\max}. \tag{7.5}$$

With probability $1 - \delta$:

$$\begin{aligned}
&\sum_{j=1}^{k_p}\sum_{t \in T_j}\epsilon_e(t, j) \\
&\quad = \sum_{j=1}^{k_p}\sum_{t \in T_j}\left(\gamma^{\tau(t,j)} + 2\sum_{h \in H}(hp^e_{h,k_a}(s_{t,j}))\right) Q_{\max} \\
&\quad = \sum_{j=1}^{k_p}\sum_{t \in T_j}\gamma^{\tau(t,j)} Q_{\max} + 2\sum_{j=1}^{k_p}\sum_{t \in T_j}\sum_{h \in H}(hp^e_{h,k_a}(s_{t,j}))Q_{\max} \\
&\quad < \frac{k_p \mathcal{N}_{\mathcal{SAM}}(d_{known}) Q_{\max}}{(1-\gamma)} \\
&\qquad + \frac{2(k+k_p)\left(1 + \log_2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}
\end{aligned}$$

$$\leqslant \frac{(2k+3k_p)\left(1+\log_2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})Q_{\max}}{1-\sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}},$$

where in step 1 we substituted equation 7.5, and in step 3 we used the fact that there can be at most $\mathcal{N}_{\mathcal{SAM}}(d_{known})$ policy changes, $|K_a| = 1$, and Lemma 7.3.3. Equation 7.5 and Lemma 7.3.3 each hold with probability $1-\frac{\delta}{2}$, which results to an overall probability of at least $1-\delta$ for our bound on $\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)$ to hold.

Since for realistically large $\mathcal{N}_{\mathcal{SAM}}(d_{known})$:

$$1-\sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}} \approx 1,$$

we have that:

$$\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)$$

$$< \frac{(2k+3k_p)\left(1+\log_2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})Q_{\max}}{1-\sqrt{\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2}{\delta}}{(k+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\approx \tilde{O}\left(\frac{\left(\frac{\hat{Q}^2_{\max}}{\epsilon_s^2(1-\gamma)^2}+k_p\right)\mathcal{N}_{\mathcal{SAM}}(d_{known})Q_{\max}}{(1-\gamma)}\right)$$

with probability $1-\delta$, where we have substituted k with

$$c\frac{\hat{Q}^2_{\max}}{2\epsilon_s^2(1-\gamma)^2}\ln\left(\frac{4\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}\right)$$

for some constant c.

From the definition of a delay step (Definition 6.3.5) we have that the maximum number of delay steps in MDP M_j is $\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j}$. Allowing for the algorithm to perform arbitrarily badly ($\epsilon_e(t,j) = Q_{\max}$) on every delay step we have that $\forall j$:

$$\sum_{t\notin T_j}\epsilon_e(t,j) \leqslant Q_{\max}\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j}.$$

□

7.4.2 Decoupling optimism and number of samples gathered

Setting K_a and ϵ_b as we did in Theorem 7.4.1 causes our algorithm to wait for an approximation unit to accumulate k samples before using it. By contrast, Theorem 7.4.2 sets K_a and ϵ_b in such a way that approximation units are used from the moment they are added to the approximation set. This results in a TCE bound with square root dependence on k (versus linear dependence on k for Theorem 7.4.1), effectively shaving a factor of $\frac{\hat{Q}_{\max}}{\epsilon_s(1-\gamma)}$ from our bounds.

Theorem 7.4.2. *Let $(s_{1,j}, s_{2,j}, s_{3,j}, \dots)$ for $j \in \{1, \dots, k_p\}$ be the random paths generated in MDPs $M_1, \dots, M_{k_p}$ respectively on some execution of Algorithm 1, and $\boldsymbol{\tilde{\pi}}$ be the (non-stationary) policy followed by algorithm 1. Let $\boldsymbol{k_m} = 1$, $\boldsymbol{\epsilon_b} = \hat{Q}_{\max}\sqrt{\frac{\ln \frac{4[1+\log_2 k]\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{2}}$, $\boldsymbol{k} \geqslant \frac{\hat{Q}^2_{\max}}{2\epsilon_s^2(1-\gamma)^2} \ln\left(\frac{4[1+\log_2 k]\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}\right)$, $\boldsymbol{K_a} = \{2^0, 2^1, 2^2, \dots, 2^i\} \cup \{k\}$ where i is the largest positive integer such that $2^i < k$, ϵ_c be defined as in Definition 5.2.13, ϵ_a be defined as in Algorithm 2, k_p be defined as in Definition 6.3.1, and $\boldsymbol{T_j}$ be the set of non-delay steps (see Definition 6.3.5) for MDP M_j. If $\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2[1+\log_2 k]}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} < 1$ and $\mathcal{N}_{\mathcal{SAM}}(d_{known}) \geqslant 2$, with probability at least $1-\delta$, for all t, j:*

$$V^{\tilde{\pi}}(s_{t,j}, M_j) \geqslant V^*(s_{t,j}, M_j) - \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} - 3\epsilon_s - \epsilon_e(t,j),$$

where:

$$\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j) = \sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j) + \sum_{j=1}^{k_p}\sum_{t\notin T_j}\epsilon_e(t,j),$$

with:

$$\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)$$

$$< \left(\left(1 + \log_2 \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \right) \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) \hat{Q}_{\max} \right.$$

$$\left. \right) \frac{(2 + \lceil 3 + \log_2 k \rceil k_p) + \left(9\sqrt{k} + 10k_p\right) \sqrt{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\approx \tilde{O} \left(\frac{\left(\frac{\hat{Q}_{\max}}{\epsilon_s(1-\gamma)} + k_p \right) \mathcal{N}_{\mathcal{SAM}}(d_{known}) \hat{Q}_{\max}}{(1-\gamma)} \right)$$

and:

$$\sum_{t \notin T_j} \epsilon_e(t, j) \leqslant \left(Q_{\max} \sum_{k_a \in K_a} \sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a} \right),$$

where D_{i,j,k_a} is the delay of the k_a-th sample for $k_a \in K_a$ in the i-th approximation unit, with respect to MDP M_j. Note that the probability of success $1 - \delta$ holds for all timesteps in all MDPs simultaneously, and $\sum_{j=1}^{k_p} \sum_{t=0}^{\infty} \epsilon_e(t, j)$ is an undiscounted infinite sum. Unlike $\sum_{j=1}^{k_p} \sum_{t \in T_j} \epsilon_e(t, j)$ which is a sum over all MDPs, $\sum_{t \notin T_j} \epsilon_e(t, j)$ gives a separate bound on the total cost due to value function update delays for each MDP separately.

Proof. From Definition 5.2.13 we have that:

$$B^{\pi^*} Q_{\tilde{U}}(s, a, M) - \epsilon_c \leqslant \mathrm{E}\left[F^{\pi^*}(Q_{\tilde{U}}, u(s, a, M)) \right] - \frac{\epsilon_b}{\sqrt{k_a(s, a, M)}}$$

and

$$\mathrm{E}\left[F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s, a, M)) \right] \leqslant B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s, a, M) + \epsilon_c + \frac{\epsilon_b}{\sqrt{k_a(s, a, M)}},$$

where the expectations are over the next-states that samples in $u(s, a, M)$ used by F^{π^*} and $F^{\pi^{Q_{\tilde{U}}}}$ respectively land on.

From Lemma 7.1.1 we have:

$$P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s, a, M)) - B^{\pi^*} \tilde{Q}(s, a, M) \leqslant -\epsilon_c \right)$$

$$\leqslant P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M))\right] \leqslant -\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right)$$

$$\leqslant e^{\frac{-2\left(\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right)^2 k_a(s,a,M)}{\hat{Q}_{\max}^2}}$$

$$= e^{\frac{-2\left(\hat{Q}_{\max}\sqrt{\frac{\ln \frac{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{2k_a(s,a,M)}}\right)^2 k_a(s,a,M)}{\hat{Q}_{\max}^2}}$$

$$= \frac{\delta}{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}$$

and:

$$P\Big(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + 2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\Big)$$

$$\leqslant P\left(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - \mathrm{E}\left[F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M))\right] \geqslant \frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right)$$

$$\leqslant e^{\frac{-2\left(\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right)^2 k_a(s,a,M)}{\hat{Q}_{\max}^2}}$$

$$= e^{\frac{-2\left(\hat{Q}_{\max}\sqrt{\frac{\ln \frac{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{2k_a(s,a,M)}}\right)^2 k_a(s,a,M)}{\hat{Q}_{\max}^2}}$$

$$= \frac{\delta}{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}.$$

Substituting 0 for $\epsilon^-\left(\tilde{U}, u(s,a,M)\right)$ and $2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}$ for $\epsilon^+\left(\tilde{U}, u(s,a,M)\right)$ in Lemma 7.3.2 we have that since $|K_a| = \lceil 1+\log_2 k\rceil$, with probability at least $1-\frac{\delta}{2}$:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^*} Q_{\tilde{U}}(s,a,M) > -\epsilon_a - 2\epsilon_c, \tag{7.6}$$

and:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s,a,M)$$

$$< \epsilon_a + 2\epsilon_c + 2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} + 2d(s,a,M,N(s,a,M),d_{known}), \tag{7.7}$$

for all $(s, a, M, \tilde{U})$ during non-delay steps. We also have that:

$$Q_{\tilde{U}}(s, a, M) - B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s, a, M) \leqslant Q_{\max} \ \forall \ (s, a, M, \tilde{U}).$$

Since $k_a^- \geqslant \frac{k_a}{2} \ \forall k_a \in K_a - \{1\}$ we have that for any (s, a, M) within d_{known} distance of an approximation unit with $k_a^- > 0$ active samples, with probability $1 - \frac{\delta}{2}$:

$$Q_{\tilde{U}}(s, a, M) - B^{\pi^{Q_{\tilde{U}}}} Q_{\tilde{U}}(s, a, M) < \epsilon_a + 2\epsilon_c + 2\hat{Q}_{\max}\sqrt{\frac{2 \ln \frac{4[1+\log_2 k]\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}}.$$

Substituting $k \geqslant \frac{\hat{Q}_{\max}^2}{2\epsilon_s^2(1-\gamma)^2} \ln\left(\frac{4[1+\log_2 k]\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}\right)$ for k_a into equation 7.7, we have that with probability at least $1-\frac{\delta}{2}$ for any (s, a, M) within d_{known} of an approximation unit with k active samples[4]:

$$\tilde{Q}(s, a, M) - B^{\pi^{Q_{\tilde{U}}}} \tilde{Q}(s, a, M) < \epsilon_a + 2\epsilon_c + 2(1-\gamma)\epsilon_s.$$

Let T_H, H, $\tilde{U}(t, j)$, $\tau(t, j)$, and $p^e_{h,k_a}(s_{t,j})$ be defined as in Lemma 7.3.3. Even though $\tilde{\pi}$ is non-stationary, it is comprised of stationary segments. Starting from step t in MDP M_j, $\tilde{\pi}$ is equal to $\pi^{Q_{\tilde{U}(t,j)}}$ for at least $\tau(t, j)$ steps. Substituting the above into Lemma 3.2.7 we have that with probability at least $1 - \frac{\delta}{2}$, for all M_j and $t \in T_j$:

$$V^*(s_{t,j}, M_j) - V^{\tilde{\pi}}(s_{t,j}, M_j) \leqslant \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} + 3\epsilon_s + \epsilon_e(t, j),$$

where:

$$\epsilon_e(t, j) = \gamma^{\tau(t,j)} Q_{\max} + 2\sum_{h \in H}(hp^e_{h,1}(s_{t,j}))Q_{\max}$$

$$+ \sum_{k_a \in \{K_a - 1\}} 2\sum_{h \in H}(hp^e_{h,k_a}(s_{t,j}))2\hat{Q}_{\max}\sqrt{\frac{2 \ln \frac{4[1+\log_2 k]\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}}.$$

Define:

$$c_0 = \left(1 + \log_2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil\right)\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}).$$

[4] Notice that in this case $d(s, a, M, N(s, a, M), d_{known}) = 0$.

From the above it follows that:

$$
\begin{aligned}
&\sum_{j=1}^{k_p} \sum_{t \in T_j} \epsilon_e(t,j) \\
&= \sum_{j=1}^{k_p} \sum_{t \in T_j} \Bigg(\gamma^{\tau(t,j)} Q_{\max} \\
&+2 \sum_{h \in H} (hp^e_{h,1}(s_{t,j})) Q_{\max} \\
&+ \sum_{k_a \in \{K_a - 1\}} 2 \sum_{h \in H} (hp^e_{h,k_a}(s_{t,j})) 2\hat{Q}_{\max} \sqrt{\frac{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}} \Bigg) \\
&= \sum_{j=1}^{k_p} \sum_{t \in T_j} \gamma^{\tau(t,j)} Q_{\max} \\
&+2 \sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (hp^e_{h,1}(s_{t,j})) Q_{\max} \\
&+2 \sum_{k_a \in \{K_a - 1\}} \sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (hp^e_{h,k_a}(s_{t,j})) 2\hat{Q}_{\max} \sqrt{\frac{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}} \\
&< \frac{k_p \lceil 1 + \log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) Q_{\max}}{(1-\gamma)} \\
&+ \frac{2(1+k_p) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} \\
&+2 \sum_{k_a \in \{K_a - 1\}} \frac{(k_a - k_a^- + k_p) c_0 2\hat{Q}_{\max} \sqrt{\frac{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(k_a - k_a^- + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} \\
&\leqslant \frac{(2 + \lceil 3 + \log_2 k \rceil k_p)\, c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}
\end{aligned}
$$

$$+\frac{2\sum_{k_a\in\{K_a-1\}}\left(\frac{k_a}{2}+k_p\right)c_0 2\hat{Q}_{\max}\sqrt{\frac{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{k_a}}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{(2+\lceil 3+\log_2 k\rceil k_p)c_0 Q_{\max}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$+\frac{2\sum_{k_a\in\{K_a-1\}}\left(\sqrt{k_a}+\frac{2k_p}{\sqrt{k_a}}\right)c_0\hat{Q}_{\max}\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{(2+\lceil 3+\log_2 k\rceil k_p)c_0 Q_{\max}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} + \left(2\sqrt{k}\left(1+\sum_{i=0}^{\infty}\left(\frac{1}{2^i}+\frac{1}{2^i\sqrt{2}}\right)\right)\right.$$

$$\left.+4k_p\left(\sum_{i=1}^{\infty}\frac{1}{2^i}+\sum_{i=0}^{\infty}\frac{1}{2^i\sqrt{2}}\right)\right)\frac{c_0\hat{Q}_{\max}\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{(2+\lceil 3+\log_2 k\rceil k_p)c_0 Q_{\max}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$+\frac{\left(2\sqrt{k}\left(3+\frac{2}{\sqrt{2}}\right)+4k_p\left(1+\frac{2}{\sqrt{2}}\right)\right)c_0\hat{Q}_{\max}\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{c_0\hat{Q}_{\max}\left((2+\lceil 3+\log_2 k\rceil k_p)+\left(9\sqrt{k}+10k_p\right)\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}\right)}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \left(\left(1+\log_2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})\hat{Q}_{\max}\right.$$

$$\left.\frac{(2+\lceil 3+\log_2 k\rceil k_p)+\left(9\sqrt{k}+10k_p\right)\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}\right)$$

with probability $1-\delta$, where in step 3 we used the fact that there can be at most

$\lceil 1 + \log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})$ policy changes and Lemma 7.3.3, and in step 8 we used the fact that $\hat{Q}_{\max}\sqrt{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}} \geqslant Q_{\max}$. Since Lemma 7.3.2 (used to bound the Bellman error of each $(s, a, M, \tilde{U})$) and Lemma 7.3.3 (used to bound how many times each $(s, a, M, \tilde{U})$ is encountered) hold with probability at least $1 - \frac{\delta}{2}$ each, the bound above holds with probability at least $1 - \delta$.

Since for realistically large $\mathcal{N}_{\mathcal{SAM}}(d_{known})$:

$$1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}} \approx 1,$$

we have that:

$$\begin{aligned}
&\sum_{j=1}^{k_p} \sum_{t \in T_j} \epsilon_e(t, j) \\
&\quad < \left(\left(1 + \log_2 \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil\right) \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) \hat{Q}_{\max} \right. \\
&\qquad \left. \right) \frac{(2 + \lceil 3 + \log_2 k \rceil k_p) + \left(9\sqrt{k} + 10k_p\right) \sqrt{2 \ln \frac{4\lceil 1+\log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\lceil 1+\log_2 k \rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} \\
&\quad \approx \tilde{O}\left(\frac{\left(\frac{\hat{Q}_{\max}}{\epsilon_s(1-\gamma)} + k_p\right) \mathcal{N}_{\mathcal{SAM}}(d_{known}) \hat{Q}_{\max}}{(1-\gamma)} \right)
\end{aligned}$$

with probability $1 - \delta$, where we have substituted k with

$$c \frac{\hat{Q}_{\max}^2}{2\epsilon_s^2 (1-\gamma)^2} \ln \left(\frac{4\lceil 1 + \log_2 k \rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta} \right)$$

for some constant c.

From the definition of a delay step (Definition 6.3.5) we have that the maximum number of delay steps in MDP M_j is $\sum_{k_a \in K_a} \sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a}$. Allowing for the

algorithm to perform arbitrarily badly ($\epsilon_e(t,j) = Q_{\max}$) on every delay step we have that $\forall j$:

$$\sum_{t \notin T_j} \epsilon_e(t,j) \leqslant Q_{\max} \sum_{k_a \in K_a} \sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a}.$$

□

Some readers may wonder why K_a only contains k and powers of 2, rather than all integers in $[1,k]$. When setting K_a we have two balance two competing goals: 1) Ensuring that the Bellman error drops quickly as more samples are gathered, and 2) limiting the number of policy changes. While setting K_a to all integers in $[1,k]$ accomplishes the first goal, it does poorly on the second.

7.4.3 Using the median trick

Theorem 7.4.3 shows that when k_m is set properly, we can trade a factor of $\hat{Q}^2_{max}$ for a factor of σ^2, where σ^2 is an upper bound on the variance of the Bellman operator. While in the worst case σ^2 can be as large as $\frac{\hat{Q}^2_{\max}}{4}$, in many cases it will be significantly smaller.

Theorem 7.4.3. *Let $(s_{1,j}, s_{2,j}, s_{3,j}, \dots)$ for $j \in \{1, \dots, k_p\}$ be the random paths generated in MDPs $M_1, \dots, M_{k_p}$ respectively on some execution of Algorithm 1, and $\tilde{\boldsymbol{\pi}}$ be the (non-stationary) policy followed by algorithm 1. Let $\boldsymbol{\epsilon_b} = \sigma\sqrt{4k_m}$, $\boldsymbol{k_m} = \left\lceil 5.6 \ln \frac{4\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta} \right\rceil$, $\boldsymbol{k} \geqslant \left\lceil \frac{\sigma^2 4}{(1-\gamma)^2 \epsilon_s^2} \right\rceil k_m$, $\boldsymbol{K_a} = \{2^0 k_m, 2^1 k_m, 2^2 k_m, \dots, 2^i k_m\} \cup \{k\}$ where i is the largest positive integer such that $2^i k_m < k$, ϵ_c be defined as in Definition 5.2.13, ϵ_a be defined as in Algorithm 2, k_p be defined as in Definition 6.3.1, and $\boldsymbol{T_j}$ be the set of non-delay steps (see Definition 6.3.5) for MDP M_j. If $\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} < 1$ and $\mathcal{N}_{\mathcal{SAM}}(d_{known}) \geqslant 2$, with probability at least*

$1-\delta$, *for all* t, j*:*

$$V^{\tilde{\pi}}(s_{t,j}, M_j) \geqslant V^*(s_{t,j}, M_j) - \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} - 3\epsilon_s - \epsilon_e(t,j),$$

where:

$$\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j) = \sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j) + \sum_{j=1}^{k_p}\sum_{t\notin T_j}\epsilon_e(t,j),$$

with:

$$\begin{aligned}
&\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j) \\
&< \left(\left(1+\log_2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})\right. \\
&\left.\right)\frac{\left(2k_m + \left\lceil 3+\log_2\frac{k}{k_m}\right\rceil k_p\right)Q_{\max} + \left(9\sqrt{k}+10k_p\right)\sigma\sqrt{8k_m}}{1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\left\lceil 1+\log_2\frac{k}{k_m}\right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} \\
&\approx \tilde{O}\left(\frac{\left(\frac{\sigma^2}{\epsilon_s(1-\gamma)} + k_p Q_{\max}\right)\mathcal{N}_{\mathcal{SAM}}(d_{known})}{(1-\gamma)}\right)
\end{aligned}$$

and:

$$\sum_{t\notin T_j}\epsilon_e(t,j) \leqslant \left(Q_{\max}\sum_{k_a\in K_a}\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a}\right),$$

where D_{i,j,k_a} *is the delay of the* k_a*-th sample for* $k_a \in K_a$ *in the* i*-th approximation unit, with respect to MDP* M_j*. Note that the probability of success* $1-\delta$ *holds for all timesteps in all MDPs simultaneously, and* $\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j)$ *is an undiscounted infinite sum. Unlike* $\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)$ *which is a sum over all MDPs,* $\sum_{t\notin T_j}\epsilon_e(t,j)$ *gives a separate bound on the total cost due to value function update delays for each MDP separately.*

Proof. From Lemma 7.2.2 we have that:

$$P\left(F^{\pi^*}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) \leqslant -\epsilon_c\right) \leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2},$$

and

$$P\left(F^{\pi^{Q_{\tilde{U}}}}(Q_{\tilde{U}}, u(s,a,M)) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \geqslant \epsilon_c + 2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}\right) \leqslant \frac{\delta}{4\left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}.$$

Substituting 0 for $\epsilon^-\left(\tilde{U}, u(s,a,M)\right)$ and $2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}}$ for $\epsilon^+\left(\tilde{U}, u(s,a,M)\right)$ in Lemma 7.3.2 we have that since $|K_a| = \left\lceil 1+\log_2 \frac{k}{k_m}\right\rceil$, with probability at least $1-\frac{\delta}{2}$:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^*}Q_{\tilde{U}}(s,a,M) > -\epsilon_a - 2\epsilon_c \tag{7.8}$$

and:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) < \epsilon_a + 2\epsilon_c + 2\frac{\epsilon_b}{\sqrt{k_a(s,a,M)}} + 2d(s,a,M,N(s,a,M),d_{known}), \tag{7.9}$$

for all $(s,a,M,\tilde{U})$ during non-delay steps. We also have that:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) \leqslant Q_{\max} \;\forall\; (s,a,M,\tilde{U}).$$

Since $k_a^- \geqslant \frac{k_a}{2}\ \forall k_a \in K_a - \{k_m\}$, we have that for any (s,a,M) within d_{known} distance of an approximation unit with $k_a^- > 0$ active samples with probability $1-\frac{\delta}{2}$:

$$Q_{\tilde{U}}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}}Q_{\tilde{U}}(s,a,M) < \epsilon_a + 2\epsilon_c + 2\sigma\sqrt{\frac{8k_m}{k_a}}.$$

Substituting $k \geqslant \left\lceil \frac{\sigma^2 4}{(1-\gamma)^2\epsilon_s^2}\right\rceil k_m$ for k_a into equation 7.9, we have that with probability at least $1-\frac{\delta}{2}$ for any (s,a,M) within d_{known} of an approximation unit with k active

samples[5]:

$$\tilde{Q}(s,a,M) - B^{\pi^{Q_{\tilde{U}}}}\tilde{Q}(s,a,M) < \epsilon_a + 2\epsilon_c + 2(1-\gamma)\epsilon_s.$$

Let T_H, H, $\tilde{U}(t,j)$, $\tau(t,j)$, and $p^e_{h,k_a}(s_{t,j})$ be defined as in Lemma 7.3.3. Even though $\tilde{\pi}$ is non-stationary, it is comprised of stationary segments. Starting from step t in MDP M_j, $\tilde{\pi}$ is equal to $\pi^{Q_{\tilde{U}(t,j)}}$ for at least $\tau(t,j)$ steps. Substituting the above into Lemma 3.2.7 we have that with probability at least $1-\frac{\delta}{2}$, for all M_j and $t \in T_j$:

$$V^*(s_{t,j},M_j) - V^{\tilde{\pi}}(s_{t,j},M_j) \leqslant \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} + 3\epsilon_s + \epsilon_e(t,j),$$

where:

$$\epsilon_e(t,j) = \gamma^{\tau(t,j)}Q_{\max} + 2\sum_{h\in H}(hp^e_{h,1}(s_{t,j}))Q_{\max} + \sum_{k_a\in\{K_a-1\}} 2\sum_{h\in H}(hp^e_{h,k_a}(s_{t,j}))2\sigma\sqrt{\frac{8k_m}{k_a}}.$$

Define:

$$c_0 = \left(1+\log_2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known}).$$

From the above it follows that:

$$\begin{aligned}
&\sum_{j=1}^{k_p}\sum_{t\in T_j}\epsilon_e(t,j)\\
&= \sum_{j=1}^{k_p}\sum_{t\in T_j}\Bigg(\gamma^{\tau(t,j)}Q_{\max}\\
&+2\sum_{h\in H}(hp^e_{h,1}(s_{t,j}))Q_{\max}\\
&+\sum_{k_a\in\{K_a-k_m\}} 2\sum_{h\in H}(hp^e_{h,k_a}(s_{t,j}))2\sigma\sqrt{\frac{8k_m}{k_a}}\Bigg)
\end{aligned}$$

[5] Notice that in this case $d(s,a,M,N(s,a,M),d_{known}) = 0$.

$$= \sum_{j=1}^{k_p} \sum_{t \in T_j} \gamma^{\tau(t,j)} Q_{\max}$$

$$+2 \sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (hp^e_{h,1}(s_{t,j})) Q_{\max}$$

$$+2 \sum_{k_a \in \{K_a - k_m\}} \sum_{j=1}^{k_p} \sum_{t \in T_j} \sum_{h \in H} (hp^e_{h,k_a}(s_{t,j})) 2\sigma \sqrt{\frac{8k_m}{k_a}}$$

$$< \frac{k_p \left\lceil 1 + \log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) Q_{\max}}{(1-\gamma)}$$

$$+ \frac{2(k_m + k_p) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$+2 \sum_{k_a \in \{K_a - k_m\}} \frac{(k_a - k_a^- + k_p) c_0 2\sigma \sqrt{\frac{8k_m}{k_a}}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_a-k_a^-+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} + \frac{2\sum_{k_a \in \{K_a - k_m\}} \left(\frac{k_a}{2} + k_p\right) c_0 2\sigma \sqrt{\frac{8k_m}{k_a}}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} + \frac{2\sum_{k_a \in \{K_a - k_m\}} \left(\sqrt{k_a} + \frac{2k_p}{\sqrt{k_a}}\right) c_0 \sigma \sqrt{8k_m}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}} + \left(2\sqrt{k}\left(1 + \sum_{i=0}^{\infty}\left(\frac{1}{2^i} + \frac{1}{2^i\sqrt{2}}\right)\right)\right.$$

$$\left. +4k_p\left(\sum_{i=1}^{\infty}\frac{1}{2^i} + \sum_{i=0}^{\infty}\frac{1}{2^i\sqrt{2}}\right)\right) \frac{c_0 \sigma \sqrt{8k_m}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) c_0 Q_{\max}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$+ \frac{\left(2\sqrt{k}\left(3 + \frac{2}{\sqrt{2}}\right) + 4k_p\left(1 + \frac{2}{\sqrt{2}}\right)\right) c_0 \sigma \sqrt{8k_m}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \frac{c_0 \left(\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) Q_{\max} + \left(9\sqrt{k} + 10k_p\right) \sigma \sqrt{8k_m}\right)}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\leqslant \left(\left(1 + \log_2 \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil\right) \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) \right)$$

$$\frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p\right) Q_{\max} + \left(9\sqrt{k} + 10k_p\right) \sigma \sqrt{8k_m}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

with probability $1 - \delta$, where in step 3 we used the fact that there can be at most $\left\lceil 1 + \log_2 \frac{k}{k_m} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})$ policy changes. Since Lemma 7.3.2 (used to bound the Bellman error of each $(s, a, M, \tilde{U})$) and Lemma 7.3.3 (used to bound how many times each $(s, a, M, \tilde{U})$ is encountered) hold with probability of at least $1 - \frac{\delta}{2}$ each, the bound above holds with probability of at least $1 - \delta$.

Since for realistically large $\mathcal{N}_{\mathcal{SAM}}(d_{known})$:

$$1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m + k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}} \approx 1,$$

and $\sigma \leqslant \frac{Q_{\max}}{2}$, we have that:

$$\sum_{j=1}^{k_p} \sum_{t \in T_j} \epsilon_e(t, j)$$

$$< \left(\left(1 + \log_2 \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \right) \left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known}) \right.$$

$$\left. \right) \frac{\left(2k_m + \left\lceil 3 + \log_2 \frac{k}{k_m} \right\rceil k_p \right) Q_{\max} + \left(9\sqrt{k} + 10k_p \right) \sigma \sqrt{8k_m}}{1 - \sqrt{\frac{2\left\lceil \frac{1}{1-\gamma} \ln \frac{Q_{\max}}{\epsilon_s} \right\rceil \ln \frac{2\left\lceil 1+\log_2 \frac{k}{k_m} \right\rceil}{\delta}}{(k_m+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}}$$

$$\approx \tilde{O} \left(\frac{\left(\frac{\sigma^2}{\epsilon_s(1-\gamma)} + k_p Q_{\max} \right) \mathcal{N}_{\mathcal{SAM}}(d_{known})}{(1-\gamma)} \right)$$

with probability $1-\delta$, where in the last step we have substituted k_m with its definition (a constant times a logarithm that gets suppressed by $\tilde{O}$) and k with

$$c \frac{\sigma^2 4 k_m}{(1-\gamma)^2 \epsilon_s^2}$$

for some constant c.

From the definition of a delay step (Definition 6.3.5) we have that the maximum number of delay steps in MDP M_j is $\sum_{k_a \in K_a} \sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a}$. Allowing for the algorithm to perform arbitrarily badly ($\epsilon_e(t,j) = Q_{\max}$) on every delay step we have that $\forall j$:

$$\sum_{t \notin T_j} \epsilon_e(t,j) \leqslant Q_{\max} \sum_{k_a \in K_a} \sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})} D_{i,j,k_a}.$$

□

The next chapter will begin by discussing the significance of these bounds.

7.5 "Number of suboptimal steps" sample complexity bounds

Given the TCE bounds of the previous section we can now simply use Lemma 4.3.2 to convert them to "Number of suboptimal steps" sample complexity bounds. For example, by applying Lemma 4.3.2 to Theorem 7.4.2 we arrive at the following:

Corollary 7.5.1. *Let* $(s_{1,j}, s_{2,j}, s_{3,j}, \dots)$ *for* $j \in \{1, \dots, k_p\}$ *be the random paths generated in MDPs* $M_1, \dots, M_{k_p}$ *respectively on some execution of Algorithm 1, and* $\boldsymbol{\tilde{\pi}}$ *be the (non-stationary) policy followed by algorithm 1. Let* $\boldsymbol{k_m} = 1$, $\boldsymbol{\epsilon_b} = \hat{Q}_{\max}\sqrt{\frac{\ln \frac{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}{2}}$, $\boldsymbol{k} \geqslant \frac{\hat{Q}^2_{\max}}{2\epsilon_s^2(1-\gamma)^2}\ln\left(\frac{4\lceil 1+\log_2 k\rceil \mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}\right)$, $\boldsymbol{K_a} = \{2^0, 2^1, 2^2, \dots, 2^i\} \cup \{k\}$ *where* i *is the largest positive integer such that* $2^i < k$, ϵ_c *be defined as in Definition 5.2.13,* ϵ_a *be defined as in Algorithm 2,* k_p *be defined as in Definition 6.3.1, and* $\boldsymbol{T_j}$ *be the set of non-delay steps (see Definition 6.3.5) for MDP* M_j. *If* $\frac{2\left\lceil \frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil \ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})} < 1$ *and* $\mathcal{N}_{\mathcal{SAM}}(d_{known}) \geqslant 2$, *with probability at least* $1-\delta$, *there will be at most:*

$$\left(\left(1+\log_2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\right)\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})\hat{Q}_{\max}\right.$$

$$\left.\right)\frac{(2+\lceil 3+\log_2 k\rceil k_p)+\left(9\sqrt{k}+10k_p\right)\sqrt{2\ln\frac{4\lceil 1+\log_2 k\rceil\mathcal{N}_{\mathcal{SAM}}(d_{known})^2}{\delta}}}{\epsilon\left(1-\sqrt{\frac{2\left\lceil\frac{1}{1-\gamma}\ln\frac{Q_{\max}}{\epsilon_s}\right\rceil\ln\frac{2\lceil 1+\log_2 k\rceil}{\delta}}{(1+k_p)\mathcal{N}_{\mathcal{SAM}}(d_{known})}}\right)}$$

$$+\left(Q_{\max}\sum_{j=1}^{k_p}\sum_{k_a\in K_a}\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})}D_{i,j,k_a}\right)$$

$$\approx \tilde{O}\left(\frac{\left(\frac{\hat{Q}_{\max}}{\epsilon_s(1-\gamma)}+k_p\right)\mathcal{N}_{\mathcal{SAM}}(d_{known})\hat{Q}_{\max}}{\epsilon(1-\gamma)}\right)+\left(Q_{\max}\sum_{j=1}^{k_p}\sum_{k_a\in K_a}\sum_{i=1}^{\mathcal{N}_{\mathcal{SAM}}(d_{known})}D_{i,j,k_a}\right)$$

steps such that:

$$V^{\tilde{\pi}}(s_{t,j}, M_j) < V^*(s_{t,j}, M_j) - \frac{4\epsilon_c + 2\epsilon_a}{1-\gamma} - 3\epsilon_s - \epsilon,$$

where D_{i,j,k_a} *is the delay of the* k_a*-th sample for* $k_a \in K_a$ *in the* i*-th approximation unit, with respect to MDP* M_j. *Note that the probability of success* $1-\delta$ *holds for all timesteps in all MDPs simultaneously, and* $\sum_{j=1}^{k_p}\sum_{t=0}^{\infty}\epsilon_e(t,j)$ *is an undiscounted infinite sum.*

Even though "number of suboptimal steps" sample complexity in the discrete, single MDP, instantaneous update case is not the primary focus of this work, Corollary 7.5.1 is an improvement over the best available bounds for discrete MDPs (Szita and Szepesvári, 2010) by a factor of $\frac{Q_{\max}}{(1-\gamma)}$.

8

Discussion

Theorems' 7.4.2 and 7.4.3 dependence on $\frac{1}{\epsilon_s}$ is linear for the TCE metric, leading to bounds that are quadratic in $\frac{1}{\epsilon_s}$ with respect to the "number of suboptimal steps" metric. While both bounds are optimal with respect to what these metrics are measuring (see Section 4.3.1), the fact that we are able to achieve linear dependence on $\frac{1}{\epsilon_s}$ for TCE is very informative. Since TCE is arguably much better aligned with how practical applications will evaluate cost, it tells us that achieving low ϵ_s is much easier than the "number of suboptimal steps" metric would suggest.

8.1 Generalization over state-action-MDP triples

Definition 5.2.13 allows us to generalize across states, actions, and MDPs. One thing to notice is that there is no fundamental difference between generalizing across state-actions of the same MDP and state-actions of different MDPs. Generalizing across state-action-MDP triples is a natural extension of the idea of generalizing across state-actions rather than just across states.

When exploring in two identical MDPs, the distance between the same state-action in the two MDPs will be zero, while the covering number of the state-action-

MDP space will be identical to the covering number of each MDP. Conversely, when two MDPs have nothing in common, the distance between state-actions across the two MDPs will be greater than or equal to $Q_{\max}$, and the covering number of the combined state-action-MDP space will be the sum of the covering numbers of the individual MDPs. The power of our generalization scheme is that in addition to these two extreme cases, it can handle everything in between. MDPs in a state-action-MDP space can have sections in which they are identical, completely dissimilar, or exhibit varying degrees of similarity. Definition 5.2.13 allows us to take advantage of any degree of similarity to decrease the cost of exploration.

Contrary to previous work (Pazis and Parr, 2013), Definition 5.2.13 does not assume that $B^\pi Q_{\tilde{U}}(s, a, M)$ is Lipschitz continuous. The Lipschitz assumption is problematic for two reasons: Not only are Lipschitz continuity based algorithms required to know the Lipschitz constant in advance (which will typically not be the case in an exploration scenario), but even small discontinuities break their guarantees. By contrast, the effect that discontinuities and underestimations of the distance function have on Definition 5.2.13 is an increase of ϵ_c, which causes our bounds to degrade gracefully.

8.2 Discrete state-action spaces

Algorithms 1 and 2 are directly applicable to discrete MDPs. A distance function can be defined over a discrete space just as easily as it can be defined over a continuous one. Even if no distance information exists, by picking the distance function which returns $d(s, a, M, \bar{s}, \bar{a}, \bar{M}) = 0$ when $(s, a, M) = (\bar{s}, \bar{a}, \bar{M})$ and $d(s, a, M, \bar{s}, \bar{a}, \bar{M}) = \infty$ otherwise, the covering number becomes the cardinality of the state-action-MDP space $|S||A|k_p$ and our bounds still hold for discrete MDPs.

8.3 Continuous action spaces

In Definition 5.2.4 we assumed that there exists a finite upper bound $|\mathcal{A}|$ on the number of actions the agent can take in any state. To some extent this is trivially true as long as we are dealing with electronics. Any digital to analogue converter will have finite precision and accept a discrete set of values as input. While this is sufficient to satisfy efficient PAC-MDP requirements, enumerating the available range is impractical for many practical applications.

Dealing with action selection in continuous and/or multidimensional action spaces is an open research problem that we cannot hope to fully cover in this section. Instead we will present a simple method to limit $|\mathcal{A}|$. Similarly to how we defined a covering number over the state-action-MDP space, we can define a covering set $\mathcal{N}_a(d_{known})$ for a particular action a. We will define $|\mathcal{A}|$ such that the cardinality of the covering set for any action in the state-action-MDP space will be less than or equal to $|\mathcal{A}|$. From Definition 5.2.13, an approximate Bellman operator $\bar{B}$ that evaluates only the actions in the covering set for each next state will only differ by at most ϵ_c from the approximate Bellman operator $\tilde{B}$ that evaluates all actions (even in the cases where $\tilde{B}$ has an infinite number of available actions). This means that even if we were to discretize our action space to $|\mathcal{A}|$ intelligently selected actions, we would only have to pay a penalty of at most $\frac{\epsilon_c}{1-\gamma}$ in our bounds.

8.4 Incorporating prior knowledge

Apart from a distance function (which is not even required to be informative for discrete state-action-MDP spaces), our analysis assumed that we are starting from a blank slate without any knowledge of the MDPs we are exploring. It is very easy to incorporate prior knowledge from multiple sources, without adversely affecting our bounds.

8.4.1 Existing sample sets

Existing sample sets generated from the same state-action-MDP space (but not necessarily from the same MDPs we are going to explore) can be simply added to the inclusion candidate queue before (or even during) policy execution. Every sample added to the queue before policy execution begins, that was generated from the same MDPs we are going to be exploring in, reduces exploration's sample requirements by one.

8.4.2 Partial knowledge of the value function

Rather than the fixed bound of $Q_{\max}$ for every state-action-MDP triple, prior knowledge of a better upper bound for some state-action-MDP triples can be incorporated directly into the Bellman operator. $((1-\gamma)\epsilon_s + \epsilon_a)$-accurate prior knowledge of the value of the identifying state-action-MDP triple of an approximation unit effectively reduces the covering number by one.

8.4.3 Partial knowledge of the reward and transition models

One of the ways in which we can derive a good distance function is by partial knowledge of the reward and transition models. Upper bounds on the reward received from various state-action-MDP triples and knowledge about their next-state distributions can be used to derive very accurate distance functions (or refine a coarser distance function). A distance function that accurately reflects the behavior of the Bellman operator can help keep the covering number small.

While beyond the scope of this work, there are many other approaches that can be combined with our algorithms to incorporate prior knowledge without adversely affecting our bounds (Mann, 2012).

9

Related Work

9.1 Bellman error bounds

Error bounds based on the max-norm of the Bellman error have been known for over two decades (Williams and Baird, 1993). While those bounds are provably tight with respect to the max-norm of the error, they cannot take into account fine-grained information about the distribution of Bellman errors. For the same reason that the maximum expected discounted accumulated reward of an MDP can be much smaller than $\frac{\mathcal{R}_{\max}}{1-\gamma}$, bounds based on the max-norm of the Bellman error can be loose.

Bellman error MDPs are much more flexible and intuitive, allowing us to prove stronger bounds with less effort. Additionally, Bellman error MDP bounds imply max-norm bounds. As we have shown in Section 3.4, previously known bounds based on the max-norm of the Bellman error can be derived as a direct consequence of the fact that in a discounted MDP where the reward function is in $[R_{\min}, R_{\max}]$, the value function is bounded by $\left[\frac{R_{\min}}{1-\gamma}, \frac{R_{\max}}{1-\gamma}\right]$.

9.2 Cost of exploration metrics

9.2.1 Regret

Regret is a metric that shares the fine-grained nature of TCE. While it has gained significant traction in other exploration settings (Ortner and Ryabko, 2012), as explained in Section 4.1, it is unsuitable for our setting.

Because regret compares the performance of an optimal policy to the learner's performance *from the starting state*, it is very easy to design *discounted* MDPs for which regret is arbitrarily bad (see section 4.1 for more details). Fortunately TCE does not share the same drawback.

9.2.2 Sample complexity for exploration

In the discounted MDP exploration setting, or when we cannot assume a finite diameter, the most commonly used metric is the number of time-steps t such that $V^{\pi}(s_t) < V^{*}(s_t) - \epsilon$, commonly referred to as "sample complexity". One of the main drawbacks of sample complexity is that it is very coarse: It only counts the number of significantly suboptimal steps, and does not discriminate between the magnitude of different errors. TCE shares the broad applicability of sample complexity yet at the same time is fine grained. Additionally, as we have shown in Section 4.3, TCE bounds imply sample complexity bounds.

9.3 Simultaneous learning and policy execution

While it appears to be an obvious idea, the only other work we are aware of where learning and policy execution happen in parallel processes is the work of Hester et al. (Hester and Stone, 2013, 2012; Hester et al., 2012). The authors correctly argue that while there exist algorithms that learn with few samples, and there exist algorithms that are sufficiently computationally efficient so as to be able to take

actions in realtime, no previous work addresses both problems simultaneously. They then proceed to demonstrate experimentally that their algorithm is in fact able to learn very quickly while acting in realtime.

While the experimental results of Hester et al. are very encouraging, their algorithm does not come with any performance guarantees, nor does it handle concurrent exploration in multiple MDPs (though extending it to the multiple MDP case appears straightforward).

9.4 Concurrent learning in multiple MDPs

A recent paper has demonstrated that it is possible to explore in multiple MDPs simultaneously (Guo and Brunskill, 2015). Our work significantly improves on that result, by:

- extending concurrent exploration to continuous state-action spaces.
- allowing for non-instantaneous value function updates.
- reducing the sample complexity of exploration on all key quantities.
- allowing smooth generalization between concurrent MDPs, rather than clustering MDPs and treating each cluster as identical.

9.5 Using samples as soon as they become available

An interesting algorithm for discrete MDPs is Model-Based Interval Estimation (Strehl and Littman, 2005). Unlike many algorithms before it, Model-Based Interval Estimation integrates new samples as soon as they are available, instead of waiting until enough samples have been gathered for a particular state-action to get a good estimate. When k_a is set as in Theorems 7.4.2 or 7.4.3, Algortihm 2 is somewhere in between Model-Based Interval Estimation and other PAC algorithms in that sense.

It does not use every single sample as soon as it becomes available, but it does not wait until an approximation unit has k samples before using it either.

Lattimore and Hutter (2012) present an algorithm that integrates new samples every time the available number of samples for a particular state-action doubles. Their algorithm also has a significantly reduced dependence on the discount factor. Unfortunately their bounds only hold for the restricted case of discrete MDPs where every state-action leads to at most two next states. While they do give a way to transform any MDP to one where each state-action leads to at most two next states, the transformation leads to quadratic dependence on the number of state-actions, and requires scaling γ to $\gamma^{\frac{1}{\log_2(|S||A|}}$ (which ultimately makes their bounds far weaker than the state of the art).

9.6 PAC optimal exploration in continuous state-action spaces

Exploration in Metric State Spaces (Kakade et al., 2003) is the first example in the PAC MDP literature which tries to address exploration in continuous state spaces. While definitely a step in the right direction, the paper did not offer a concrete algorithm. Instead, an efficient black box approximate planner and local approximate modeling algorithm are assumed to exist. Unfortunately some of the assumptions regarding the abilities of these black box algorithms are overly optimistic (e.g. good model approximation everywhere, not just with some probability $1 - \delta$).

C-PACE (Pazis and Parr, 2013), the author's first publication in PAC optimal exploration, is to the best of our knowledge the first concrete PAC-optimal exploration algorithm for online, discounted, continuous state-action MDPs. The work presented in this thesis represents a drastic, multifaceted improvement over C-PACE.

Grande et al. (2014) have recently shown that it is possible to perform sample efficient exploration with Gaussian processes. Their work was motivated by the fact

that many PAC-optimal algorithms (including C-PACE), perform a fixed point computation after every step, which makes them unsuitable for realtime applications. They present DGPQ, the first Q-learning inspired algorithm for PAC-optimal exploration in continuous spaces. Experimental results with DGPQ are encouraging, as the authors demonstrate that DGPQ requires significantly less computation per step than C-PACE. While it is encouraging to see that the community is finally developing an interest for PAC-optimal exploration algorithms that can be used in realistic applications, DGPQ does have some drawbacks of its own:

- DGPQ assumes that it will always be able to perform an update between steps. Even if performing an update is fast when using a sparse GP approximation, there may be domains for which performing one update per step is impossible.

- Like all Q-learning inspired algorithms, DGPQ requires significantly more samples than fixed-point based methods.

As this thesis has demonstrated, we can have the best of both worlds: The sample efficiency of fixed-point based methods can be combined with realtime execution if we allow for policy execution and learning to run on parallel processes.

Nouri and Littman (2008) and Jong and Stone (2007) present interesting algorithms for exploration in continuous state spaces, but stop short of providing PAC-optimal bounds.

For cases where the user can come up with a good set of features but not an exploration strategy, Strehl and Littman (2008) provide an algorithm that can explore in polynomial time in environments whose dynamics can be accurately modeled by linear regression.

Similarly, Brunskill et al. (2009) also use parametric models, but allow for the state-action space to be partitioned into multiple models, rather than a monolithic one.

9.7 PAC optimal exploration in discrete state-action spaces

For better or worse, after more than two decades of research, there is no shortage of theoretical papers focused on the discrete PAC-MDP case. Unfortunately, the huge constants associated with discrete PAC-MDP bounds preclude the use of the associated algorithms in non-trivially sized discrete MDPs, which coupled with their inability to handle continuous spaces makes them inapplicable in any kind of realistic domains. While the discrete MDP case is an obvious place to start, this not very practical setting has received a disproportionate amount of attention.

Rather than provide an exhaustive list, we will summarize the contributions of a representative set of publications focusing on discrete MDPs.

PAC bounds were introduced to the reinforcement learning community by Fiechter (1994). Inspired by Valiant's PAC learning framework (Valiant, 1984), Fiechter was the first to prove that PAC optimal exploration in MDPs is possible, and provided the first PAC optimal exploration algorithm.

An assumption made by Fiechter's algorithm was that the agent has access to a reset action that can take it back to the starting state(s). Kearns and Singh (2002) presented E^3, an algorithm that did not require a reset action and handled the exploration-exploitation trade-off in an explicit manner. The guarantee provided by E^3 was that after a number of steps polynomial in all relevant quantities, it would halt in some state and output a policy that was PAC optimal with respect to that (one) state.

While Fiechter introduced PAC bounds to RL, the definition of sample complexity has evolved over time. For PAC-optimal exploration algorithms in discounted MDPs, the most commonly used one was introduced by Kakade (2003).

Delayed Q-learning (Strehl et al., 2006), a model-free exploration algorithm, was the first PAC optimal exploration algorithm with log-linear sample complexity on

the number of state-actions. Other aspects of its bounds have since been improved upon by a model based algorithm (Szita and Szepesvári, 2010).

9.8 Other forms of exploration

While PAC-optimal exploration is arguably one of the most theoretically interesting forms of exploration, other forms of exploration have been proposed and used over the years.

One of the simplest and most commonly used approaches to exploration is the so called ϵ-greedy family of algorithms. Such algorithms select a random action with probability ϵ and act greedily according to their current belief otherwise. As long as $\epsilon > 0$ they are guaranteed to explore the entire state-action space eventually, although even in the case where $\epsilon = 1$ the time required may be exponential in the size of the state-action space. In practice ϵ is adjusted throughout the execution of the algorithm according to some "cooling" schedule, usually starting at or very close to 1 and eventually approaching 0.

Another approach to exploration is that of Bayesian or PAC-Bayesian exploration (Kolter and Ng, 2009). Bayesian exploration tries to optimize for a very different goal than typical PAC-optimal methods. Its assumption is that all that matters is the cumulative discounted reward from the current state, and as such it chooses to explore unknown state-actions only when those state-actions are expected to perform better that the known state-actions. This leads to a significantly more myopic algorithm that explores far less than other PAC-optimal methods. Such an approach would be appropriate in situations where its assumptions are true, which for example may include the testing phase in a learning scenario with both learning and testing phases.

Finally, the simplest form of exploration is not an algorithm, but rather a set of (not particularly realistic) assumptions which allow us to ignore the issue. The

first assumption is that all state-actions have non-zero probability of reaching any state, while the second is that the agent's policy will be stochastic, selecting all actions with non-zero probability. Together these assumptions imply that all state-actions will eventually be explored. Of course the first assumption is not true for many real-world domains, and the second implies that the cost of exploration will be infinite.

10

Experimental Evaluation

This chapter presents an experimental evaluation of exploration on the noisy pendulum swing-up problem (Wang et al., 1996). In this problem we start with a pendulum of unknown length and mass that is hanging downwards, and the goal is to bring it and balance it to the upright position by applying forces to the cart to which it is attached. The 2-dimensional continuous state space includes the vertical angle θ and angular velocity $\dot{\theta}$ of the pendulum. The action space of the process is the set of forces in $\{-50N, 0N, 50N\}$, with uniform noise in $[-25N, 25N]$ added to each action (significantly more than typical for this domain). The MDP space is the mass of the pendulum, which was allowed to vary in $[1kg, 10kg]$.

A reward of $1 - (\frac{\theta}{\pi})^2$ was given as long as $|\theta| \leqslant \pi/2$, and a reward of 0 as long as $|\theta| > \pi/2$. The discount factor of the process was set to 0.98, the control interval to 100ms, and each episode was 200 steps long. The distance function was set to the two norm of the difference between state-action-MDPs with the action space rescaled to $[-1, 1]$. K_a was set to $\{1, 2, 4\}$, ϵ_b was set to 0.1, k_m was set to 1, ϵ_a was set to 0.001, and d_{known} was set to 0.1. All graphs are averages over 100 repetitions of each experiment.

Since both the model and a vast amount of accumulated knowledge exists for this domain, many algorithms exist that achieve good performance when taking advantage of this information. The purpose of these experiments is not to claim that policies produced by exploration outperform policies produced by such algorithms. The goal of these experiments is to demonstrate that we can tackle this problem even under the weakest of assumptions, with an algorithm that provides strong theoretical guarantees, providing some indication that exploration would be able to perform well on domains where no such knowledge exists.

10.1 Performance under resource constraints

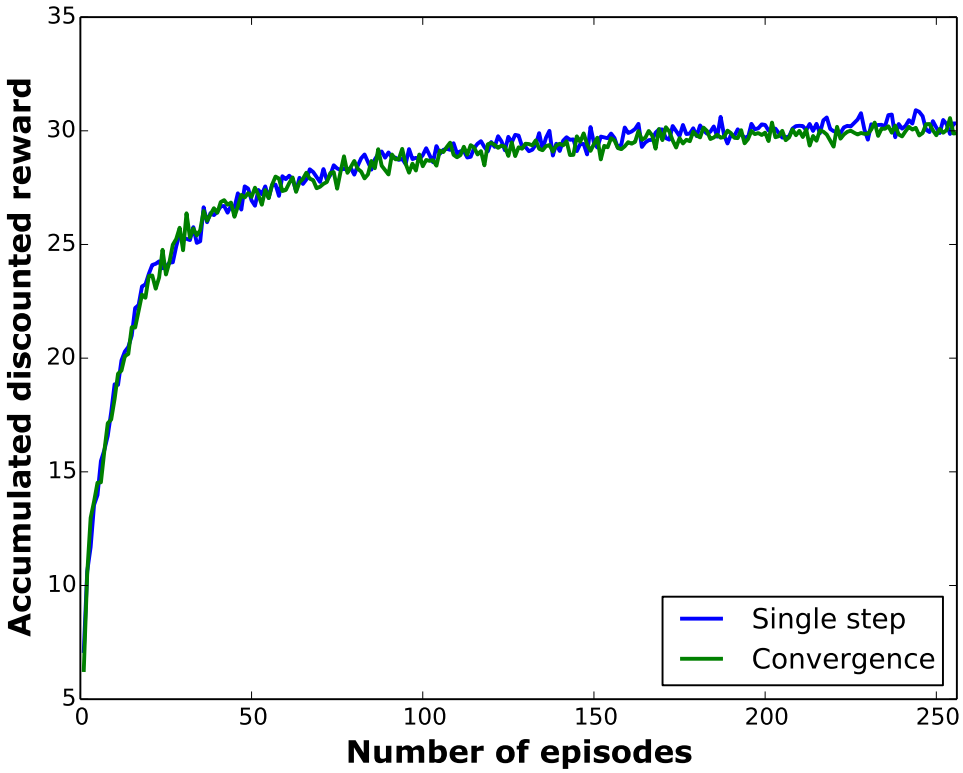

FIGURE 10.1: Accumulated discounted reward per episode, when performing a single Bellman backup between steps (blue line), and performing as many Bellman backups as it takes for the value function to converge between steps (green line).

One of the major advantages of our algorithm is that it can maintain PAC guarantees even when computational constraints prevent the learner from reaching a fixed point between steps. The goal of the first experiment is to investigate the difference in performance between the idealized, instantaneous update case, and the case where we can only perform a limited amount of computation between steps. For this experiment we used a single pendulum with a mass of 2 kg.

Figure 10.1 compares the accumulated discounted reward per episode when performing a single Bellman backup between steps (blue line), versus performing as many Bellman backups as it takes for the value function to converge between steps (green line). The blue line represents what would be typical in a realtime application, while the green line represents the ideal (instantaneous update) case. The two graphs are practically indistinguishable, indicating that our algorithm can perform well even under real-world computational constraints.

10.2 Generalization across different MDPs

Another major advantage of our algorithm is that it allows us to generalize across non-identical MDPs. The goal of our second experiment is to illustrate the role of the distance function in inter-MDP generalization. In this experiment our algorithm was concurrently exploring on a pendulum of mass 1 kg and a pendulum of mass 10 kg. Figure 10.2 compares exploration performance for three different settings of the distance function. The MDP component of the distance function (the difference in mass between the pendulums in question) was multiplied with $\frac{1}{0.0001}$ (no generalization, blue line), $\frac{1}{+\infty}$ (aliasing, green line), and $\frac{1}{20}$ (generalization, red line).

When the MDP component of the distance function is multiplied with $\frac{1}{0.0001}$, the distance between samples of different pendulums is greater than $Q_{\max}$. Thus, samples originating from the mass 1 kg pendulum do not contribute to the value

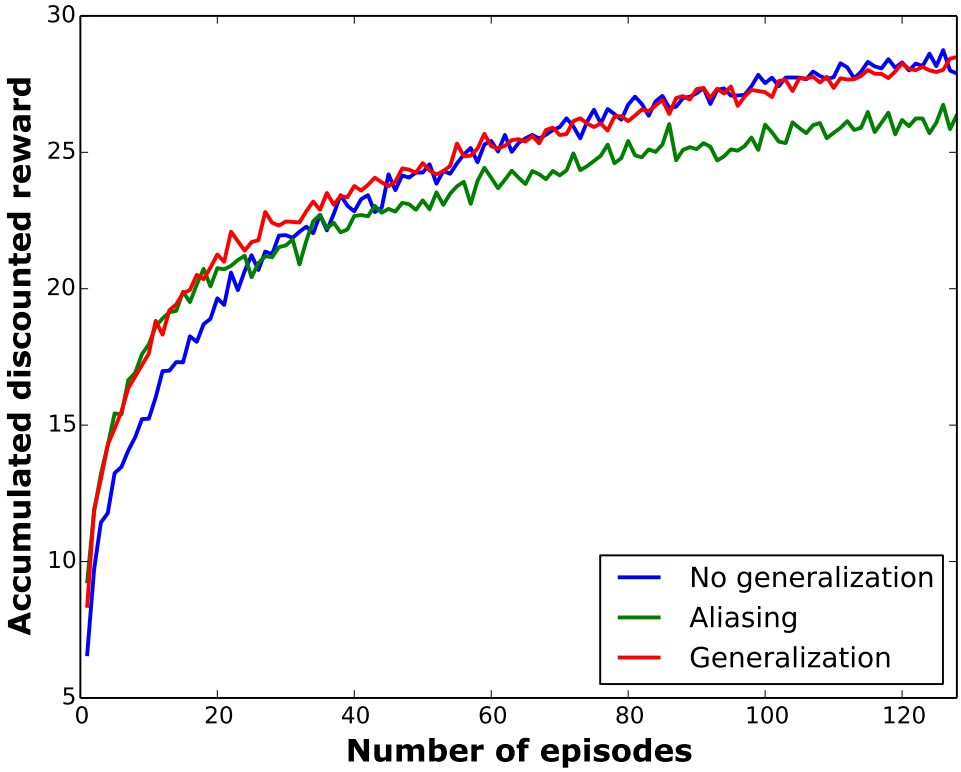

FIGURE 10.2: Accumulated discounted reward per episode when concurrently exploring with a pendulum of mass 1 kg and a pendulum of mass 10 kg for three different settings of the distance function.

function of the mass 10 kg pendulum and vise versa. In effect, we are exploring in two MDPs independently. The result is a policy that improves slowly, but achieves good performance eventually.

When the MDP component of the distance function is multiplied with $\frac{1}{+\infty}$, the distance between samples of different pendulums that have the same state-action is 0. In this case, all samples are treated as if they were generated by both pendulums. The rate with which samples that can be used in each MDP are collected is twice the rate of the independent exploration case. This results in a policy whose performance improves quickly. On the other hand, the fact that we are not distinguishing between the two pendulums limits performance even when many samples have been gathered.

Finally, when the MDP component of the distance function is multiplied with $\frac{1}{20}$ we get the best of both worlds. Samples originating from the mass 1 kg pendulum provide useful info about the mass 10 kg pendulum and vise versa. This helps performance improve quickly. On the other hand, since we are still able to distinguish between samples collected from each MDP, our performance when many samples are available is as good as in the case where we are exploring in each MDP independently.

10.3 Concurrent exploration

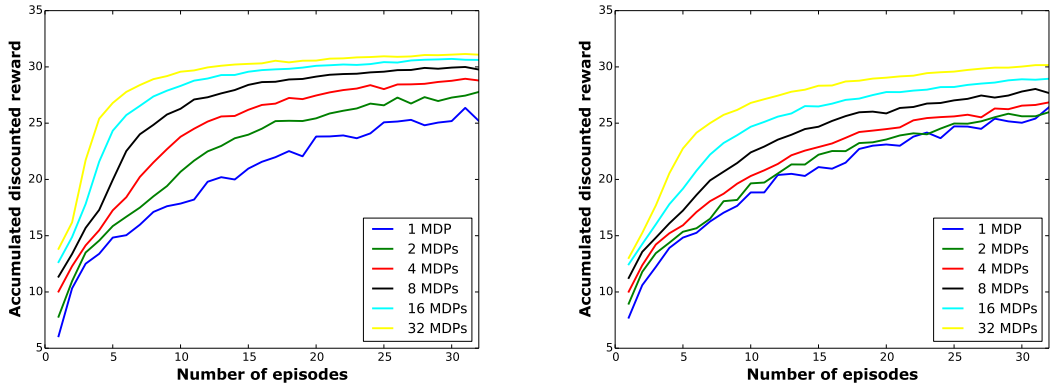

FIGURE 10.3: Accumulated discounted reward per episode when exploring in a set of identical pendulums all of which have a mass of 2 kg (left), or when exploring on a set of pendulums where the mass of each pendulum was drawn uniformly at randomly from $[1kg, 10kg]$ (right). The x axis is the number of episodes executed on *each* concurrent MDP.

The purpose of our third experiment was to evaluate concurrent exploration performance both when the MDPs were identical, and when they were chosen from a distribution. For the identical case all pendulums had a mass of 2 kg. For the distribution case, at the start of each of the 100 repetitions of each experiment a set of pendulums was chosen each with a mass uniformly drawn from $[1kg, 10kg]$. The set of pendulums was then kept constant between episodes. The MDP component of the distance function was multiplied with $\frac{1}{20}$.

Figure 10.3 shows the results for both the identical MDP case (left) and the MDP distribution case (right). In both cases, the more concurrent MDPs we explore in, the faster performance improves. In other words, exploring in multiple MDPs concurrently reduces the cost of exploration per MDP.

10.4 Sequential versus concurrent exploration

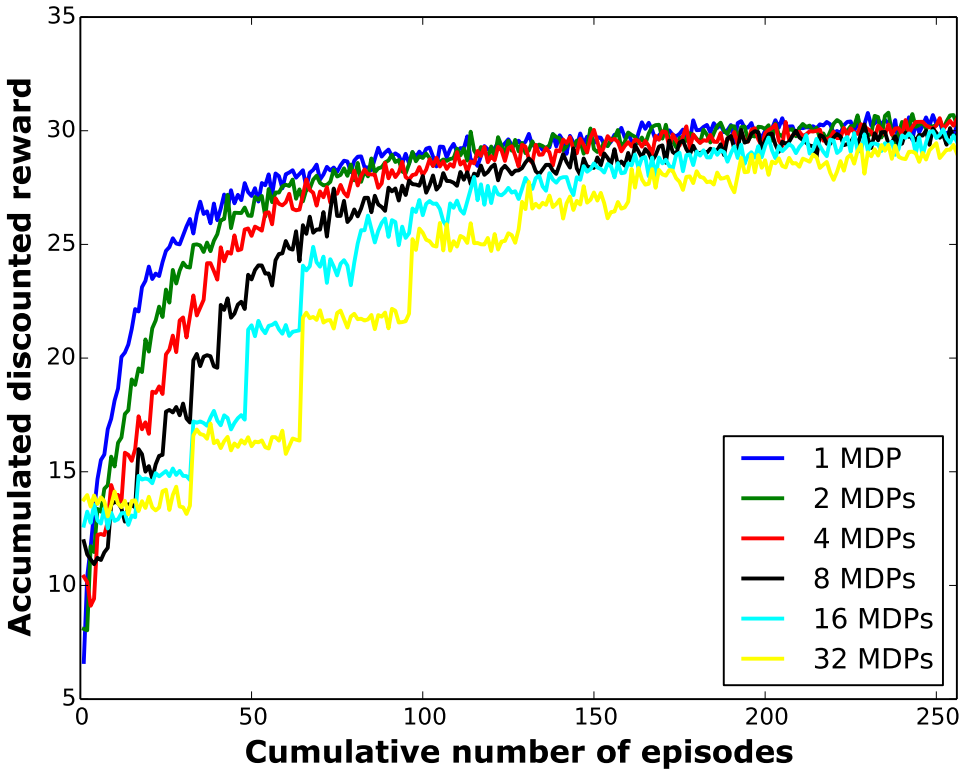

FIGURE 10.4: Accumulated discounted reward per episode when exploring in a set of identical MDPs. The x axis is the *sum* of episodes executed on all concurrent MDPs.

For our final experiment we wanted to evaluate the performance of exploration when we have a choice of executing a finite number of episodes concurrently or sequentially. Figure 10.4 shows the accumulated discounted reward per episode when exploring in a set of identical MDPs. Unlike figure 10.3, this time the x axis is

the *sum* of episodes executed on all concurrent MDPs. As expected, performance improves faster when each episode is executed sequentially rather than concurrently. In other words, exploring in multiple MDPs concurrently increases the total cost of exploration. This is agreement with what is predicted from our bounds (increasing k_p increases TCE).

When taken together, the last two experiments lead us to the following conclusion: If concurrent exploration is necessary, a single learner is preferable to exploring in each MDP separately. On the other hand, when a choice between sequential and concurrent exploration exists, sequential exploration leads to lower TCE.

11

Conclusion

11.1 Contribution

This work has made a number of novel contributions to the field of PAC-optimal exploration in particular, and to the field of reinforcement learning in general:

- Our first contribution is the theory of Bellman error MDPs, a new analysis methodology for reinforcement learning algorithms. The use of Bellman error MDPs simplifies analysis and allows us to derive fine-grained bounds. This thesis focused on using Bellman error MDPs to analyze the PAC-optimal exploration algorithm presented in Chapter 6, but Bellman error MDPs can just as easily be used to analyze algorithms in different settings, such as batch mode learning methods.

- We have introduced TCE, a new metric on the total cost of exploration for the online, discounted reinforcement learning setting. TCE is more fine-grained and useful than typical "number of significantly suboptimal steps" PAC bounds. As with Bellman error MDPs we concentrated on analyzing our own algorithm in terms of TCE, but the same analysis can be performed on many existing

algorithms. In addition, it would make sense for future work to concentrate on the TCE metric, since as we have shown TCE PAC-bounds imply PAC-bounds on the number of significantly suboptimal steps.

- Arguably the most important contribution of this thesis is the introduction of a concrete PAC-optimal exploration algorithm that can deal with continuous and/or large state-action spaces. We view this as the first step needed to render PAC-optimal exploration algorithms useful in practical applications, rather than purely theoretical concepts.

- While lowering the sample complexity of exploration with respect to the size of the state-action space has been the first priority for most publications in PAC optimal exploration, the dependence on the discount factor is also important. With the settings presented in Theorem 7.4.2, Algorithm 2 improves over the state of the art in both continuous and discrete exploration algorithms by a factor of $\frac{Q_{\max}}{1-\gamma}$.

- Our use of the median trick is to the best of our knowledge not only the first in PAC-optimal exploration, but also the first for any reinforcement learning algorithm. It allows us to improve our bounds for MDPs in which the variance of the Bellman operator is less than $\frac{\hat{Q}_{\max}}{2}$.

- This work has also improved on recent results on concurrent exploration by extending it to continuous MDPs, and drastically improving on previous bounds even for the discrete case.

- Finally, another step this thesis takes in making PAC-optimal exploration algorithms useful in practical applications is the first known analysis of stochastic and non-uniform delayed updates. Our analysis shows that we can maintain

PAC-optimal guarantees even under realistic computational constraints and/or network delays.

11.2 Future work

This work focused on the fully observable case. Unfortunately, in real life full observability is a luxury we seldomly have. In most domains our state information is both incomplete and noisy. Robust RL methods that take partial observability and measurement noise into account, and try to correct for them, are an important next step.

One aspect of the real world which our exploration algorithms did not take into account is that in many real life systems catastrophic failures are unacceptable. For example, even if our budget allows us to crash a number of autonomous helicopters while learning to fly, that number will be prohibitively small compared to most PAC bounds. Consequently methods for safe exploration are of great real world interest (Moldovan and Abbeel, 2012).

We designed and analyzed our algorithm in a setting where we only care about the total cost of exploration, not when this cost is incurred. While this setting is popular with the theoretical community, in practical applications *when* costs are incurred is far from irrelevant. Consider for example the case where PAC-optimal exploration is employed by a company designing a product. Exploration costs incurred during development are far less important than exploration costs incurred when the product has reached the end user. Designing and analyzing algorithms that have explicit exploration and exploitation phases based on the theory developed in this work is a promising next step.

A central assumption made by our algorithm in the context of continuous state-action spaces and multiple concurrent MDPs, is that there exists some distance function in which the effects of applying the Bellman operator are approximately

smooth[1]. While it is reasonable to expect that such a distance function exists, many obvious distance functions that a user might try may not satisfy this requirement, or may have a large smoothness constant. Automatic discovery of suitable distance functions is an important next step.

[1] Our bounds allow for discontinuous functions, and small local discontinuities will not significantly affect performance. Unfortunately, large discontinuities on a global scale can make ϵ_c unacceptably large.

Bibliography

Alon, N., Matias, Y., and Szegedy, M. (1999), "The space complexity of approximating the frequency moments," *Journal of Computer and System Sciences - JCSS (special issue of selected papers from STOC'96)*, 58, 137–147.

Brunskill, E., Leffler, B., Li, L., Littman, M., and Roy, N. (2009), "Provably efficient learning with typed parametric models," *The Journal of Machine Learning Research*, 10, 1955–1988.

Fiechter, C.-N. (1994), "Efficient Reinforcement Learning," in *In Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pp. 88–97, ACM Press.

Grande, R. C., Walsh, T. J., and How, J. P. (2014), "Sample Efficient Reinforcement Learning with Gaussian Processes," in *International Conference on Machine Learning*, pp. 1332–1340.

Guo, Z. and Brunskill, E. (2015), "Concurrent PAC RL." in *AAAI Conference on Artificial Intelligence*, pp. 2624–2630.

Hester, T. and Stone, P. (2012), "Intrinsically Motivated Model Learning for a Developing Curious Agent," in *The Eleventh International Conference on Development and Learning*.

Hester, T. and Stone, P. (2013), "TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots," *Machine Learning*, 90, 385–429.

Hester, T., Quinlan, M., and Stone, P. (2012), "RTMBA: A Real-Time Model-Based Reinforcement Learning Architecture for Robot Control," in *IEEE International Conference on Robotics and Automation*, pp. 85–90.

Jaksch, T., Ortner, R., and Auer, P. (2010), "Near-optimal Regret Bounds for Reinforcement Learning," *Journal of Machine Learning Research*, 11, 1563–1600.

Jong, N. and Stone, P. (2007), "Model-based exploration in continuous state spaces," *Abstraction, Reformulation, and Approximation*, 4612, 258–272.

Kaelbling, L. P., Littman, M., and Moore, A. (1996), "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, 4, 237–285.

Kakade, S., Kearns, M. J., and Langford, J. (2003), "Exploration in Metric State Spaces," in *International Conference on Machine Learning*, pp. 306–312.

Kakade, S. M. (2003), "On the sample complexity of reinforcement learning," Ph.D. thesis, Gatsby Computational Neuroscience Unit, University College London.

Kearns, M. J. and Singh, S. P. (2002), "Near-Optimal Reinforcement Learning in Polynomial Time," *Machine Learning*, 49, 209–232.

Kolter, J. Z. and Ng, A. Y. (2009), "Near-Bayesian exploration in polynomial time," in *International Conference on Machine Learning*, pp. 513–520.

Lattimore, T. and Hutter, M. (2012), "PAC Bounds for Discounted MDPs," in *Proceedings of the 23th International Conference on Algorithmic Learning Theory*, vol. 7568 of *Lecture Notes in Computer Science*, pp. 320–334, Springer Berlin / Heidelberg.

Li, L. (2009), "A unifying framework for computational reinforcement learning theory," Ph.D. thesis.

Mann, T. A. (2012), "Scaling Up Reinforcement Learning Without Sacrificing Optimality by Constraining Exploration," Ph.D. thesis.

McDiarmid, C. (1989), "On the method of bounded differences," in *Surveys in Combinatorics*, no. 141 in London Mathematical Society Lecture Note Series, pp. 148–188, Cambridge University Press.

Moldovan, T. M. and Abbeel, P. (2012), "Safe Exploration in Markov Decision Processes," in *Proceedings of the 29th International Conference on Machine Learning*, pp. 1711–1718.

Nouri, A. and Littman, M. (2008), "Multi-resolution exploration in continuous spaces," *Advances in Neural Information Processing Systems*, 21, 1209–1216.

Ortner, R. and Ryabko, D. (2012), "Online Regret Bounds for Undiscounted Continuous Reinforcement Learning," in *Advances in Neural Information Processing Systems 25*, pp. 1772–1780.

Pazis, J. and Parr, R. (2013), "PAC Optimal Exploration in Continuous Space Markov Decision Processes," in *AAAI Conference on Artificial Intelligence*, pp. 774–781.

Puterman, M. L. (1994), *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley-Interscience.

Strehl, A. and Littman, M. (2008), "Online linear regression and its application to model-based reinforcement learning," *Advances in Neural Information Processing Systems*, 20, 1417–1424.

Strehl, A. L. and Littman, M. L. (2005), "A theoretical analysis of Model-Based Interval Estimation," in *International Conference on Machine Learning*, pp. 856–863, New York, NY, USA, ACM.

Strehl, A. L., Li, L., Wiewiora, E., Langford, J., and Littman, M. L. (2006), "PAC model-free reinforcement learning," in *International Conference on Machine Learning*, pp. 881–888.

Strehl, A. L., Li, L., and Littman, M. L. (2009), "Reinforcement Learning in Finite MDPs: PAC Analysis," *Journal of Machine Learning Research*, 10, 2413–2444.

Sutton, R. and Barto, A. (1998), *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, Massachusetts.

Szita, I. and Szepesvári, C. (2010), "Model-based reinforcement learning with nearly tight exploration complexity bounds," in *International Conference on Machine Learning*, pp. 1031–1038.

Valiant, L. G. (1984), "A Theory of the Learnable," *Communications ACM*, 27, 1134–1142.

Wang, H., Tanaka, K., and Griffin, M. (1996), "An Approach to Fuzzy Control of Nonlinear Systems: Stability and Design Issues," *IEEE Transactions on Fuzzy Systems*, 4, 14–23.

Williams, R. and Baird, L. (1993), "Tight Performance Bounds on Greedy Policies Based on Imperfect Value Functions," Tech. rep., Northeastern University, College of Computer Science.

Biography

Jason Pazis

Born in Toronto Canada on the 3rd of July 1984

Ph.D. in Computer Science, Duke University

Committee: Ronald Parr (advisor), Vincent Conitzer, George Konidaris, Mauro Maggioni, Peter Stone

M.Sc. in Computer Science, Duke University

Thesis: Non-parametric Approximate Linear Programming for MDPs

Committee: Ronald Parr (advisor), Vincent Conitzer, Mauro Maggioni, Silvia Ferrari

M.Sc. in Electronic and Computer Engineering, Technical University of Crete, Greece.

Thesis: Efficient Reinforcement Learning in High-Dimensional Continuous State and Action Spaces

Committee: Michail G. Lagoudakis (advisor), Michalis Zervakis, Nikos Vlassis

B.Sc. in Electronic and Computer Engineering, Technical University of Crete, Greece.

Thesis: Learning Continuous-Action Control Policies

Committee: Michail G. Lagoudakis (advisor), Michalis Zervakis, Nikos Vlassis