Noisy Hierarchical Reinforcement Learning

Viet Nguyen, School of Computer Science

McGill University, Montreal

April 15, 2023

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Computer Science

Abstract

Recently, reinforcement learning (RL) has emerged to become the framework of choice for trial-and-error learning in highly complex, dynamic environments such as game-playing, autonomous driving, and molecular biology [13]. Hierarchical reinforcement learning (HRL) decomposes a reinforcement learning problem into subtasks via temporal or state abstractions, where higher-level control invokes lower-level subtasks as if they were primitive actions [34]. HRL benefits especially in situations where the subproblems are small and highly repetitive, as a solution to one subproblem can be recycled to all other subproblems equivalent to it.

In this thesis, we study the efficiency of a posterior-sampling based hierarchical reinforcement learning algorithm, in particular, guarantees on the regret bounds of the algorithm in terms of the repeated structures within the environment. We further propose novel regret bounds in the case of noisy substructures, i.e., when the substructure similarities are imperfect, in terms of controls on the levels of the noise. We propose a framework for abstracting substructure similarities using known techniques from studies of state abstraction in reinforcmement learning and discuss previous results in this new context. Finally, we detail further expansions to our current work in various reinforcement learning settings.

Abrégé

Récemment, l'apprentissage par renforcement (RL) est devenu le cadre de choix pour l'apprentissage par essais et erreurs dans des environnements dynamiques très complexes tels que le jeu, la conduite autonome et la biologie moléculaire [13]. L'apprentissage par renforcement hiérarchique (HRL) décompose un problème d'apprentissage par renforcement en sous-tâches en utilisant des abstractions temporelles ou d'état, dans lequel le contrôle de niveau supérieur appelle des sous-tâches de niveau inférieur comme s'il s'agissait d'actions primitives [34]. La HRL bénéficie en particulier dans les situations où les sous-problèmes sont petits et très répétitifs, car une solution à un sous-problème peut être recyclée à tous les autres sous-problèmes qui lui sont équivalents.

Dans cette thèse, nous étudions l'efficacité d'un algorithme d'apprentissage par renforcement hiérarchique basé sur l'échantillonnage postérieur, en particulier les garanties sur les limites de regret de l'algorithme en termes des structures répétées dans l'environnement. De plus, nous proposons de nouvelles limites de regret dans le cas des sous-structures bruyantes, c'est-à-dire lorsque les similitudes des sous-structures sont imparfaites, en termes de contrôles sur le niveau de bruit. Nous proposons un cadre pour l'abstraction des similarités de sous-structure en utilisant des techniques connues dans l'étude de l'abstraction des états dans le domaine de l'apprentissage par renforcement et nous discutons des résultats préexistants dans ce nouveau cadre. Enfin, nous détaillons d'autres extensions de notre recherche actuelle dans divers contextes d'apprentissage par renforcement.

Acknowledgements

I am grateful to everyone whose hard work allowed me to receive the education I have had.

I would like to thank my supervisor Doina Precup for her meticulous guidance on research all throughout my master's degree and thesis. I would also like to thank Haque Ishfaq whom attentively introduced me to academic research in reinforcement learning and mentored me throughout the process.

Table of Contents

Abstract i

Abrégé ii

Acknowledgements iii

1 Introduction 2

2 Elementary theory 4

2.1 Modeling learning by trial and error 4

2.2 Discounted Markov Decision Processes 5

2.3 Bellman equations and Bellman operators 8

2.4 The planning problem: solving MDPs 10

2.5 The control problem and Q-learning 11

3 Exploration in Reinforcement Learning 13

3.1 Episodic MDPs 13

3.2 Regret 14

3.3 ϵ-greedy revisited 15

3.4 Optimism in the face of uncertainty (OFU) 16

3.5 Model-based optimistic exploration 17

3.6 Model-based posterior sampling for exploration 19

3.7 Optimistic sampling 19

3.8 Model-free exploration 20

3.9 Exploration in larger state spaces 21

4 Hierarchical Reinforcement Learning: planning and learning 23

4.1 Cleaner robots and the breakdown of multi-level tasks 23

4.2 The HRL problem formulation 25

4.3 HRL with posterior sampling-driven exploration 27

4.4 Noisy PSHRL: motivation 30

4.5 Characterizing noise 31

4.6 Online-offline data generation and the noisy PSHRL algorithm 34

4.7 Efficiency of noisy PSHRL 36

4.8 Efficient planning with hierarchical structures 37

4.9 Exit profiles: quality and suboptimality 39

5 Regret analysis of Algorithm 2 42

5.1 Regret decomposition 42

5.2 Construction of confidence sets 45

5.3 Controlling the per-timestep Bellman errors 48

5.4 Self-normalization bounds 50

5.5 Regret 53

6 Generalizing noise 54

6.1 Structural misspecification 54

6.2 MDP homomorphisms 55

6.3 Aggregated state representation 57

7 Discussion and Future Works 58

Notation

Let S be a set. When it is clear from context, we avoid cluttering of notation and use $S \coloneqq |S|$ in regret bounds and within complexity operators $(\mathcal{O}, O)$.

Chapter 1

Introduction

Coupled with powerful function approximation tools from the deep learning literature, reinforcement learning agents achieved superhuman-level performances in a wide variety of games, and show promising potential in various other fields of industry [23, 24]. In continual learning, reinforcement learning agents have been developed to adapt to a changing environment while continually parse and learn new information as they come [10,22,47]. Meanwhile, theoretical results provide guarantees on the performance of these algorithms, pushing the boundaries of our fundamental understanding of the discipline and the extent of what is possible [19, 44]. Very often, theoretical analyses are conditional on one or many assumptions about the structure and/or the dynamics of the environment in which their algorithms operate on. These algorithms leverage aspects of these peculiarities of the environments and use them to their advantage.

In this thesis, we study hierarchical reinforcement learning, reinforcement learning that leverages the knowledge of highly repeatable and hierarchical substructures innate to the environment to achieve better sample efficiency. We will describe an efficient algorithm to solve environments when the repeated substructures are essentially identical [46]. We propose a variant of this algorithm and explore how it performs when the repeated substructures slightly differ from one another. We will show that our proposed algorithm could be efficient as well, and explore further ways in which the substructures could differ from one another in terms of their structures and dynamics. We will see how ideas from hierarchical reinforcement learning could be applied in these situations as well.

Chapter 2 provides a brief mathematical introduction to the theory of reinforcement learning to help the reader familiarize themselves with the field. **Chapter 3** dives deeper into exploration, the open problem for reinforcement learning agents to try out various exploratory behaviors in a newly encountered environment in order to acquire more knowledge about it before committing to more exploitative strategies. **Chapter 4** presents hierarchical reinforcement learning in hierarchical environments as well as in near-hierarchical or "noisy" environments. We present here our provable efficiency guarantees for the noisy case, and explore how one can accelerate the algorithm's runtime by once again leveraging the hierarchical structure of the environment. The efficiency guarantees are formally proven in **Chapter 5**. **Chapter 6** presents extensions to the hierarchical model by using well-studied ideas such as MDP homomorphisms and bisimulation to describe more complex ways in which substructures could differ in a controllable way. Finally, we discuss the results obtained as well as propose several future directions in which we can proceed in **Chapter 7**.

Chapter 2

Elementary theory

2.1 Modeling learning by trial and error

At the most fundamental level, reinforcement learning is the mathematical abstraction of an intelligent agent operating in an environment. At every timestep, the agent observes the current state it is in within the environment, then chooses an action that it performs. The agent is then transitioned to another state in the environment as a result of their action taken. As an example, consider a very methodical colleague navigating a university campus. Every 10 meters, they have to choose a cardinal direction towards which they will walk the next 10 meters. Upon making a choice, they proceed with their 10 meter walk, and are then confronted again with the same question while standing still in this new state.

The agent operating in an environment is a good model for interaction. To model learning, we introduce an immediate reward signal every time the agent takes action, while making the agent behave in such a way that expected future rewards are maximized. In the case of our colleague, assume they are navigating towards building A, but does not know in which direction it is. To model their process of learning, after every decision they make, the environment can give an immediate reward signal that is inversely proportional to the remaining distance to building A to incentivize the agent (our colleague) to perform actions that will result in a lowering of the distance remaining.

Formally put, let $\mathcal{S}$ be a state space, and let $\mathcal{A}$ be an action space. At timestep t, the agent in state $s_t \in \mathcal{S}$ takes action $a_t \in \mathcal{A}$. They immediately receive a reward r_t and are then transitioned to state s_{t+1}. The figure below illustrates this loop.

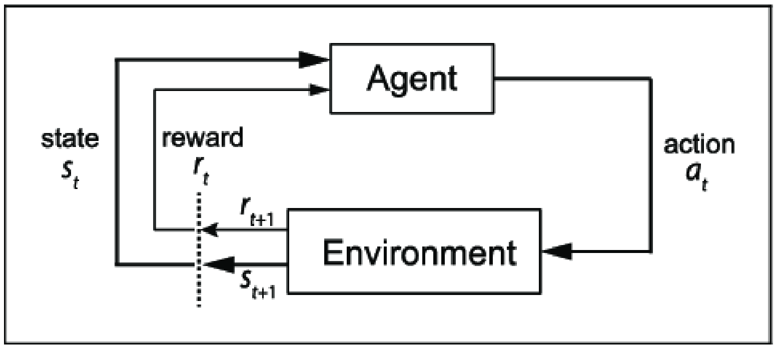

Figure 2.1: The reinforcement learning cycle.

The environment in which the agent takes actions is modeled by a Markov Decision Process (MDP).

2.2 Discounted Markov Decision Processes

A Markov Decision Process (MDP) is a mathematical formalization of the agent-environment interaction described above. We largely follow the same constructions as in [1].

Definition 2.2.1. A (infinite-horizon) discounted Markov Decision Process is a 5-tuple $M = (\mathcal{S}, \mathcal{A}, P^M, r^M, \gamma)$ specified by:

- A state space $\mathcal{S}$ which may be finite or infinite. For the remainder of this chapter, we assume $\mathcal{S}$ is finite.
- An action space $\mathcal{A}$ which for the purposes of this thesis, we will assume to be finite.

- A transition function $P^M : \mathcal{S} \times \mathcal{A} \longrightarrow \Delta(\mathcal{S})$ where $\Delta(S)$ is the probability simplex over $\mathcal{S}$. We write $P^M(s'|s,a)$ to be the probability of transitioning from state s to state s' when taking action a in the MDP M.
- A reward function $r^M : \mathcal{S} \times \mathcal{A} \longrightarrow [0,1]$, the immediate reward associated with taking action a in state s in the MDP M. When the environment is modeled by stochastic rewards $R^M(s,a)$ instead, we take $r^M(s,a) = \mathbb{E}\left[R^M(s,a)\right]$.
- A discount factor $\gamma \in [0,1)$ which defines the horizon for the problem.

Remark 2.2.2. When it is clear from context, we will omit the specification of the MDP in the transition function and in the reward function, that is, we write $r = r^M$ and $P = P^M$ when there is no confusion.

Often times, an initial state distribution $\mu \in \Delta(\mathcal{S})$ might be given, in which case, we sample our starting state $s_0 \sim \mu$. Otherwise, μ is supported on a single state s_0. At each timestep $t = 1, 2, \ldots$, the agent observes its current state s_t, takes action a_t, receives an immediate reward $r_t = r(s_t, a_t)$, and observes the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$. A sequence of t timesteps:

$$\mathcal{H}_t = (s_0, a_0, r_0, s_1, \ldots, s_t, a_t, r_t)$$

is a history. Let $\mathcal{H}$ be the space of all histories.

The agent's decision making strategy is modeled by a policy.

Definition 2.2.3. A policy is a map $\pi : \mathcal{H} \longrightarrow \Delta(\mathcal{A})$ from a history to a distribution over actions $\mathcal{A}$. At timestep t, the agent samples an action a_t from its policy $\pi(\cdot|\mathcal{H}_t)$. A stationary policy $\pi : \mathcal{S} \longrightarrow \Delta(\mathcal{A})$ specifies a decision-making strategy where the agent samples its action based only on its current state, and a deterministic stationary policy $\pi : \mathcal{S} \longrightarrow \mathcal{A}$ samples from a single action.

We now build the tools to formalize the agent's goal of maximizing future rewards.

Definition 2.2.4. For an MDP M and a policy π, we define the value function $V^{M,\pi} : \mathcal{S} \to \mathbf{R}$ as the discounted sum of future rewards:

$$V^{M,\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | \pi, s_0 = s\right]$$

where the expectation is with respect to the randomness in state transitions, immediate rewards, and the stochasticity of π.

The action-value function (or quality function) $Q^{M,\pi} : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ is defined similarly:

$$Q^{M,\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | \pi, s_0 = s, a_0 = a\right]$$

As $r \leq 1$, we have that both the value and the state-action value functions are bounded above by $1/(1-\gamma)$. We will write $V^{M,\pi} = V^{\pi}$ and $Q^{M,\pi} = Q^{\pi}$ when the MDP is clear from context.

The goal of a reinforcement learning agent is to find a policy which maximizes the value function:

$$\arg\max_{\pi \in \Pi} V^{M,\pi}(s)$$

where Π denotes the space of policies. In fact, there always exists a stationary deterministic policy that simultaneously maximizes $V^{M,\pi}(s)$ for all s. We state the theorem without proof.

Theorem 2.2.5. Let Π be the space of non-stationary and randomized policies. Define the optimal value and action-value functions:

$$V^*(s) = \sup_{\pi \in \Pi} V^{\pi}(s)$$

$$Q^*(s, a) = \sup_{\pi \in \Pi} Q^{\pi}(s, a)$$

There exists a stationary, deterministic policy π^* such that $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$V^{\pi^*}(s) = V^*(s)$$
$$Q^{\pi^*}(s,a) = Q^*(s,a)$$

We refer to such a π^* as an optimal policy.

Remark 2.2.6. We note that it is possible to have multiple optimal policies for a given MDP M, i.e. different policies that yield the same value function.

The significance of the above theorem lies in that we can restrict our policy search space to just policies that are stationary and deterministic without any loss in performance. However, many algorithms in the literature naturally output an optimal stochastic policy.

On the other hand, when given the optimal action-value function $Q^*(s,a)$, we can recover a deterministic stationary optimal policy as follows:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s,a)$$

The optimal action-value function Q^* tells us the maximum discounted expected returns one can achieve by selecting each action $a \in \mathcal{A}$. It suffices to act greedily with respect to Q^* to guarantee having maximal discounted returns at every timestep.

2.3 Bellman equations and Bellman operators

Bellman equations [7] relate the value functions and the action-value functions. They are powerful tools that are commonly used in analyzing algorithms. We state them here without proof. For a more in-depth derivation of these equations, we refer the reader to [39] and [41].

Lemma 2.3.1. (Bellman consistency equations and operators) Let $\pi \in \Pi$ be a stationary policy. Then, $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$\begin{aligned} V^\pi(s) &= Q^\pi(s, \pi(s)) \\ Q^\pi(s,a) &= r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[V^\pi(s')\right] \end{aligned}$$

Define the Bellman operator for value functions $\mathcal{T}^{M,\pi} : (\mathcal{S} \to \mathbf{R}) \to (\mathcal{S} \to \mathbf{R})$ by its action on a function $f : \mathcal{S} \to \mathbf{R}$:

$$\mathcal{T}^{M,\pi}[f](s) = r^M(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim P^M(\cdot|s,a)}\left[f(s')\right]$$

We can also define the Bellman operator for action-value functions in a similar fashion. Again, when M is understood from context, we omit the superscript.

On top of characterizing any value function, we precisely characterize the optimal action-value function with the Bellman optimality equation.

Lemma 2.3.2. Let $Q : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$ be a function. We say that Q satisfies the Bellman optimality equation if $\forall (s,a) \in \mathcal{S} \times \mathcal{A}$:

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in \mathcal{A}} Q(s', a')\right]$$

For any function Q of the correct type, $Q = Q^*$ if and only if Q satisfies the Bellman optimality equation.

Define the Bellman optimality operator $\mathcal{T}^* : (\mathcal{S} \times \mathcal{A} \to \mathbf{R}) \to (\mathcal{S} \times \mathcal{A} \to \mathbf{R})$ by its action on a function $f : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$:

$$\mathcal{T}^*[f](s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a' \in \mathcal{A}} f(s', a')\right]$$

we write $Q = Q^* \iff Q = \mathcal{T}^* Q$, in other words, a function Q is an optimal action-value function if and only if it is a fixed point of the Bellman optimality operator.

2.4 The planning problem: solving MDPs

Given full knowledge of our environment (MDP), we wish to output an optimal policy. We can work with action-value functions directly, that is, we search for the fixed-point of the Bellman optimality operator; this is the principal idea behind value iteration. As seen previously, given the optimal action-value function, we easily recover an optimal policy by acting greedily with respect to the action taken. On the other hand, we can work directly with policies, and sequentially output policies that yield increasingly higher value functions; this is the principal idea behind policy iteration.

As $\gamma < 1$, it can be shown that $\mathcal{T}^*$ is a contraction mapping on the space of functions of type $\mathcal{S} \times \mathcal{A} \to \mathbf{R}$ bounded above by $1/(1-\gamma)$ endowed with the sup norm $\|\cdot\|_\infty$. This normed space is indeed a Banach space. Therefore, $\mathcal{T}^*$ has a unique fixed point by the Banach fixed-point theorem. A full proof is presented in Appendix A of [41].

Value iteration starts off with any arbitrary function $Q_0 : \mathcal{S} \times \mathcal{A} \to \mathbf{R}$, and repeatedly applies the Bellman optimality operator to yield a sequence $\{Q_0, Q_1, Q_2, \dots\}$ converging to the optimal value function Q^*.

Concretely, we simultaneously update for all pairs $(s, a) \in \mathcal{S} \times \mathcal{A}$ until convergence:

$$Q(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) \max_{a' \in \mathcal{A}} Q(s', a')$$

Policy iteration, on the other hand, repeats the following two steps until convergence:

1. Policy evaluation: with policy π^t at timestep t, compute Q^{π^t}.

2. Policy improvement: we compute a new improved policy for the next timestep: $\forall s \in \mathcal{S}, \pi^{t+1}(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi^t}(s, a)$.

It can be shown that the sequences of value functions Q^{π^t} is non-decreasing and converge in sup norm to Q^*. An in-depth analysis of value iteration and policy iteration is given in [1].

2.5 The control problem and Q-learning

Suppose now that the agent does not have access to any information about the environment. The control problem consists of simultaneously discover new knowledge about the environment as well as using that new knowledge to find better policies. This setting appropriately models the "learning by trial-and-error" behavior that reinforcement learning seeks to capture. One would like for the agent to start off with exploratory policies, i.e. policies that prioritize observing more states and trying newer actions, and slowly opt for more exploitative policies. Based on current known information about the environment, exploitative policies maximize expected returns.

Holistic treatments of this subject can be found in [39]. We will present the Q-learning algorithm, which segue us into the next chapter.

Q-learning consists of repeatedly updating an estimate for the action-value function, while acting ϵ-greedily with respect to it, that is, at every timestep t, with probability $1 - \epsilon_t$, the action $a_t = \arg\max_{a \in \mathcal{A}} Q(s_t, a)$ is chosen while a random action is taken with probability ϵ_t.

Recall that at timestep t, the optimal action-value function satisfies $Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s' \sim P(\cdot|s_t, a_t)}[\max_{a' \in \mathcal{A}} Q^*(s', a')]$. Denote by Q the estimate for this optimal action-value function that our algorithm maintains. Upon observing an interaction (s_t, a_t, r_t, s_{t+1}), since s_{t+1} is a sample from $P(\cdot|s_t, a_t)$, we can use $r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$ as the (temporal-difference learning) target towards which we update our Q estimate in order to "push" Q

towards Q^*. Given a learning rate schedule α_t, we have an update rule for Q:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_{\alpha' \in \mathcal{A}} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

With every update of our Q estimate, our temporal-difference (TD) target changes. To make learning more stable, one can consider setting $Q^- = Q$ and update towards our stable TD target $r_t + \gamma \max_{a' \in \mathcal{A}} Q^-(s_{t+1}, a')$ for τ amount of timesteps before resetting Q^-.

An easy way to introduce exploration into our control scheme is to start off with $\epsilon_0 = 1$ (that is, behave completely randomly) and decay ϵ_t over time with respect to t, stopping at $\epsilon_t = 0.1$ for all $t \geq T$ for some T large. However, as we will see in the next section, this method of exploration leaves much to be desired.

Chapter 3

Exploration in Reinforcement Learning

The treatment of exploration in reinforcement learning is a difficult and lengthy one. As such, we will primarily be introducing ideas that are relevant for the analysis of noisy hierarchical reinforcement learning. We borrow the exposition of the subject done in [15]. We also present the theory in the episodic MDP case.

3.1 Episodic MDPs

An episodic MDP M is a MDP specified by $M = (\mathcal{S}, \mathcal{A}, \{P_h^M\}_{h\in[H]}, \{r_h^M\}_{h\in[H]}, H, \mu)$ where H is the horizon of the MDP, that is, every episode terminates in H timesteps. We have transition functions P_h^M and reward functions r_h^M for each timestep, thus making our problem non-stationary by nature. However, in what follows, we will be studying the setting of episodic stationary MDPs, that is, there is only one transition function and one reward function, independent of the timestep.

Consider now an agent in state s at timestep $h \leq H$, following policy π. Its value function is given by:

$$V^{M,\pi}(s) = \mathbb{E}_\pi \left[\sum_{i=h}^{H} r^M(s_i, a_i) | s_h = s, M \right]$$

similarly, its action-value function is given by:

$$Q^{M,\pi}(s, a) = \mathbb{E}_\pi \left[\sum_{i=h}^{H} r^M(s_i, a_i) | s_h = s, a_h = a, M \right]$$

We remark that since the episode necessarily terminates in H steps, there is no need to discount the rewards. Instead, we take an expectation over future rewards until termination while proceeding with π. Furthermore, we can convert non-stationary episodic MDPs into stationary episodic MDPs by mapping its non-stationarities into a state space of size $\mathcal{S} \times H$ where states are duplicated across timesteps.

3.2 Regret

A useful quantity to measure the optimality of an episodic MDP algorithm is the regret. After K episodes, the regret of an algorithm is:

$$\text{Regret}(K) = \sum_{k=1}^{K} \left(V^*(s_0) - V^{\pi^k}(s_0) \right)$$

that is, the cumulative difference in value achieved by an improving policy π^k with respect to the optimal policy π^*. Let H denote the horizon of the episodic MDP, and $T = HK$ the number of timesteps taken up until episode K. An algorithm is generally considered to be efficient when the regret is sublinear in T.

3.3 ϵ-greedy revisited

There are a few immediate drawbacks with an ϵ-greedy exploration strategy. Firstly, it is highly sample inefficient as samples are used exactly once. Furthermore, ϵ-greedy is not effective in covering the state space. Consider the Riverswim environment [28] in **Figure 3.1**.

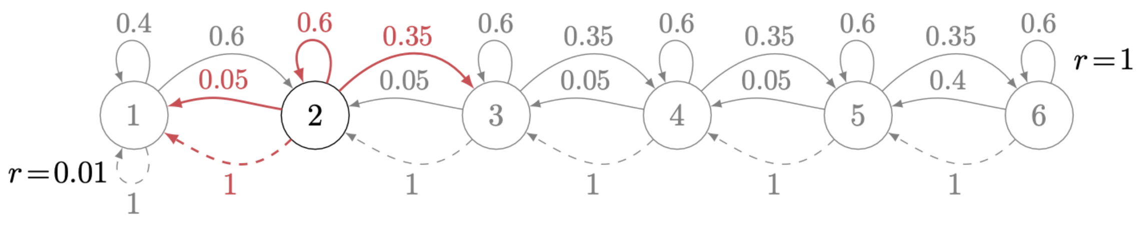

Figure 3.1: The Riverswim environment is a good stress-tester for exploration algorithms. Undirected exploration schemes usually fail to find the optimal policy due to the exponentially decreasing probability of reaching the final state.

Ideally, the agent would like to reach state 6, where it can keep selecting the option to transition to the right and earn a reward of 1 for every timestep. We observe that at any intermediate states 2-5, selecting the right action would only transition the agent to the right with probability 0.35, while selecting the left action would surely transition the agent left. The probability of the agent playing a random policy reaching the final state 6 is proportional to $1/2^x$, decreasing exponentially as the river gets longer (as we introduce more intermediate states). A random policy will most likely get stuck on the first few states before ϵ_t decays enough for the agent to exploit the sub-optimal policy of just transitioning left for a small reward of $r = 0.01$.

After a lot of testing, it was found that scheduling ϵ_t as:

$$\epsilon_t = \begin{cases} 1.0 & t < 6000 \\ \frac{\epsilon_0}{N(s_t)^{1/2}} & \text{otherwise} \end{cases}$$

would result in the optimal policy being found. However, these values are highly problem-dependent, difficult to find efficiently, and often times require accurate simulations of the environment which is not possible for many real-world reinforcement learning problems (e.g. online learning in medicine, autonomous driving). Directed exploration can be achieved by baking the epistemic uncertainties about the environment into the decision making process.

3.4 Optimism in the face of uncertainty (OFU)

The principle of optimism in the face of uncertainty boils down to the philosophy that when there are unknowns about the world, we can imagine the best possible world (reward-wise) given the current knowledge and behave accordingly. If we were right about our model, then there are no regrets as we are already behaving optimally. On the other hand, if the world is indeed worse than we imagined (reward-wise), then that knowledge is learned and used into constructing the next optimistic model of the world. In either case, we are not losing since we are either optimal or acquiring useful knowledge about the environment.

Translating this philosophy into the context of reinforcement learning, assume we have partial knowledge of our environment after having played a few episodes in our environment. After computing the value function based on empirical data, we increase it with respect to the uncertainties in the environment that have yet to be covered by our empirical data. We take actions greedily with respect to this optimistic Q estimate instead. Let $N(s, a)$ count the number of times we've been in state s and taken action a, Q estimates for pairs (s, a)'s are inflated proportionally to $1/N(s, a)$.

3.5 Model-based optimistic exploration

The following treatment of this subject assumes the model runs for K episodes, within each episode runs H timesteps. At episode $k \in [K]$ let $T = kH$, we concatenate data from individual episodes in a sequential manner and denote the entirety of observed data by $\mathcal{D}_T = \{(s_t, a_t, r_t)\}_{t=0}^{T}$. Consider the empirical MDP $\hat{M}_T = (\mathcal{S}, \mathcal{A}, \hat{P}_T, \hat{r}_T, H, \mu)$ with

$$\begin{aligned}\hat{P}_T(s'|s,a) &= \frac{\sum_{t=1}^{T} \mathbb{1}\{s_t = s, a_t = a, s_{t+1} = s'\}}{N(s,a)} \\ \hat{r}_T(s,a) &= \frac{\sum_{t=1}^{T} r_t \mathbb{1}\{s_t = s, a_t = a\}}{N(s,a)}\end{aligned}$$

We follow the construction of the Bounded Parameter MDP proposed in [38]. Denote by $\mathcal{M}$ the space of all MDPs, let

$$\mathcal{M}_T = \left\{(\mathcal{S}, \mathcal{A}, P, r, H) \in \mathcal{M} : r(s,a) \in B_T^r(s,a), P(\cdot|s,a) \in B_T^P(s,a)\right\}$$

with

$$B_T^r(s,a) = [\hat{r}_T(s,a) \pm \beta_T^r(s,a)]$$

$$B_T^P(s,a) = \left\{p(\cdot|s,a) \in \Delta(\mathcal{S}) : \left\|P(\cdot|s,a) - \hat{P}_T(\cdot|s,a)\right\|_1 \leq \beta_T^P(s,a)\right\}$$

be our confidence region in the space of MDPs, where our control parameters β_T^r and β_T^P guarantee that with probability at least $1-\delta$, our true MDP lies within $\mathcal{M}_T$. From Hoeffding's inequality [16], we compute β_T^r, and from Weissman's inequality for the L1 deviation of the empirical distribution [45], we compute β_T^P:

$$\beta_T^r(s,a) \propto \sqrt{\frac{\log(N(s,a)/\delta)}{N(s,a)}}, \quad \beta_T^P(s,a) \propto \sqrt{\frac{\mathcal{S}\log(N(s,a)/\delta)}{N(s,a)}}$$

We perform value iteration to compute the Q estimate of the empirical MDP with added bonuses β_T^r and β_T^P as per the principle of Optimism in the Face of Uncertainty. Our updates become:

$$\begin{aligned} Q(s,a) &\leftarrow \max_{r(s,a)\in B_T^r(s,a)} r(s,a) + \max_{P\in B_T^P(s,a)} \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right] \\ &= \hat{r}_T(s,a) + \beta_T^r(s,a) + \max_{P\in B_T^P(s,a)} \mathbb{E}_{s'\sim P(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} Q(s',a')\right] \end{aligned}$$

In a sense, we are not updating according to the empirical nor the true MDP, but according to the MDP within $\mathcal{M}_T$ that maximizes the value function, thus resulting in inflated values and inciting exploration. Methods that inject a deterministic bonus signal into the updates of the Q estimates are commonly referred to as Upper Confidence Bounds methods (UCB).

In 2010, Jaksch et. al. [4] proved in their seminal work that for the case of episodic MDPs with stationary transitions and reward functions depend on the level $h \in [H]$, their model-based algorithm UCRL2 described above with high probability suffers a regret of $\tilde{\mathcal{O}}(H\mathcal{S}\sqrt{\mathcal{A}T} + H^2\mathcal{S}\mathcal{A})$ where $\tilde{\mathcal{O}}$ hides logarithmic terms. They also prove a theoretical lower bound for the regret of any reinforcement learning algorithm to be $\Omega(\sqrt{H\mathcal{S}\mathcal{A}T})$.

Further refinements to this regret bound came in 2017 when Azar et. al. [5] proved a regret bound of $\tilde{\mathcal{O}}(H\sqrt{\mathcal{S}\mathcal{A}T} + H^2\mathcal{S}^2\mathcal{A})$ of their algorithm UCBVI by isolating all the additional bonuses to the Q estimate update into one single exploration bonus term $b_T(s,a)$, essentially performing value iteration on the empirical MDP with inflated rewards $\hat{r}_T(s,a) + b_T(s,a)$.

Adding an exploration bonus term loosely dependent on some notion of state-visitation has since become common practice in applied reinforcement learning [21], achieving very strong results in a variety of fields such as game-playing, robotics, optimal control, and autonomous driving [20] [3].

Another very popular method for directed exploration is posterior sampling.

3.6 Model-based posterior sampling for exploration

Posterior sampling methods are the application of Thompson's sampling method [43] proposed in 1933, but has quickly emerged in the field of reinforcement learning as a principal tool to drive exploration. At its core, a distribution over the space of MDPs is maintained at every episode and updated as the agent further interacts with the environment and collects more data. After each episode, the posterior is then updated, an MDP is sampled from the updated posterior, an optimal policy is then computed for the sampled MDP, and then finally used to play the next episode in the environment.

The intuition behind why posterior sampling works is that at the beginning of the learning process when few samples are available to the agent, the posterior over MDPs will be quite uniform and thus, there would be a lot of uncertainty in the MDPs sampled. It is precisely these uncertainties that will drive the exploration in the early episodes. As more samples are observed, the posterior distribution concentrates on the true MDP, and thus the agent naturally converges towards a more exploitative strategy.

Formally, let $\mathcal{P}_0$ be the prior distribution over $\mathcal{M}$. At episode $k \in [K]$, we sample an MDP $M_k \sim \mathcal{P}_k(\cdot|\mathcal{D}_k)$, and compute an optimal policy $\pi^k \in \arg\max_{\pi\in\Pi} V^{M^k,\pi^k}$. Finally, $\mathcal{D}_{k+1}$ is observed by executing policy π^k. As shown in [15], a Dirichlet prior can be used for modeling transition probabilities, while a Beta or Normal-Gamma prior is the go-to for modeling rewards. The authors of [29] proved a regret of $\tilde{\mathcal{O}}(H\mathcal{S}\sqrt{\mathcal{A}T})$ for the posterior sampling reinforcement learning algorithm using a Dirichlet prior, and argue that posterior sampling reinforcement learning not only match the theoretical regret bounds that UCB-based methods, but are also more computationally efficient.

3.7 Optimistic sampling

Authors of [29] reveal a key insight into the workings of posterior sampling. The value functions of the sampled MDPs are stochastically optimistic with respect to the true MDP conditioned on any possible history. Essentially, this regresses posterior sampling into being a random version of UCB-based algorithms, except not as “optimistic” (since UCB-based methods directly use the upper bounds to construct exploration bonus terms, this is as optimistic as can be). In 2019, authors of [33] proved that by randomly perturbing the observed rewards with Gaussian noise (with appropriate variance tuning), they are able to approximate sampling from a posterior distribution, all the while being optimistic with respect to the true MDP’s value function with high probability.

As perturbing collected reward data by Gaussians is a lot easier and efficient to perform than maintaining posterior distributions over the space of MDPs, optimistic sampling is emerging to be a strong contender to the well-established UCB paradigm [29]. In addition, the proof techniques are largely the same as UCB-type proofs in that there is an additional step to show that the perturbed value function estimates are indeed stochastically optimistic with respect to the true value function, while most of the concentration arguments are recycled [17] [44].

3.8 Model-free exploration

Model-based methods that employ some variant of value iteration [4, 5, 33] usually suffer a time complexity of $\mathcal{O}(H\mathcal{S}^2\mathcal{A})$ due to the planning steps. Solutions such as Optimistic Real-time dynamic programming (Opt-RTDP) simultaneously play the environment while value iterating, save the time complexity by a factor of $\mathcal{S}$, and promise adaptability compared to other algorithms such as UCBVI. However, the space complexity incurred by model-based methods of $\mathcal{O}(H\mathcal{S}^2\mathcal{A})$ is harder to reduce as one needs to store $\mathcal{S}^2\mathcal{A}$ transitions and $\mathcal{S}\mathcal{A}$ rewards.

Optimistic Q-learning [18] performs Q-learning updates directly without the need to rely on the empirical model. To incite exploration, a bonus term is injected directly into the TD target for Q-learning, that is, our Q estimates update towards

$$r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a') + b_t$$

where $b_t \propto \text{const} \cdot \sqrt{\frac{H^3 \log(\mathcal{SAT}/\delta)}{N(s_t,a_t)}}$, and the step sizes $\alpha_) t$ are of order $\mathcal{O}(1/N(s,a))$ or $\mathcal{O}(1/\sqrt{N(s,a)})$. One observes that once again, our bonuses inflate the TD targets, making our algorithm update towards an optimistic estimate of the true optimal action-value function. The authors of [18] showed that Opt-Q-learning with high probability suffers a regret of $\tilde{\mathcal{O}}(H^2\sqrt{\mathcal{SAT}} + H^2\mathcal{SA})$.

3.9 Exploration in larger state spaces

So far in our exposition of reinforcement learning, we have always referred to values as a function over the state space. When our state space is finite, one represents the value function as entries to a $\mathcal{S}$ or $\mathcal{S} \times \mathcal{A}$ sized table, and successive updates to the value function essentially refer to modifying tabular values. However, many real life learning tasks require modeling the state space as a countably infinite or even a continuous space. Such cases would require modeling the value function as a parametrized function of the states instead. A typical case would be representing the value function as a neural network, that is, approximate the true value function with a neural network $f(\cdot, \theta) : \mathcal{S} \rightarrow \mathbf{R}$ where θ are parameters to the neural network f. Instead of replacing table values as we do in the finite state space setting, we now "pull" $f(\cdot, \theta)$ towards the true value function V by running gradient descent on the mean-squared error (MSE) loss between f's current value and its TD-target bootstrap $r + f(\cdot, \theta)$ to update the parameters θ.

Empirically, the above process might perform reasonably well in a variety of tasks such as Atari game playing [27]. However, we are only able to provably guarantee efficiency when strong assumptions are made, for example when the true value function lives in linear space (the linear MDP setting) [19], or in spaces we can control (RKHS, bounded eluder-dimension) [17,44,48]. Essentially, these assumptions are imposed so that value function and value function updates are tractable, and thus performance can be bounded and controlled. Whether or not battle-tested large state space environments in the applied reinforcement literature (Atari, MuJoCo, etc...) fit these assumptions is another topic of discussion altogether, and it is a remarkable feat that very successful results have been obtained by large state space algorithms that as of current do not have theoretical guarantees [6, 11].

The theory for exploration in large state spaces will not be covered further in this thesis as the setting for noisy hierarchical reinforcement learning is that of finite state spaces. However, we will be proposing possible extensions to the theory when applied in continuous hierarchical environments in a later chapter.

Chapter 4

Hierarchical Reinforcement Learning: planning and learning

In this chapter, we see how HRL arises naturally from compositions of reinforcement learning problems as well as some standard techniques for agents to learn hierarchical environments. We present our results in the misspecified setting, i.e. characterizing the expected worst-case performance of our HRL algorithm when the equivalence of subMDPs assumption is loosened. Finally, we study the conditions that enable efficient planning in hierarchical MDPs.

4.1 Cleaner robots and the breakdown of multi-level tasks

Consider a robot agent tasked to clean an apartment building. There are a number of floors with different layouts to clean, and within each floor there are rooms with different layouts that the robot must learn to navigate in. Assume now that there are only several possible layouts for floors, as well as only several possible layouts for rooms. A naive approach would be to employ a reinforcement learning agent that goes from floor to floor and learns how to clean each floor individually. A more principled approach would be to break down the control sequence of the agent into a higher level control which controls the robot's navigation between rooms within a floor, and a lower level control which guides the robot in cleaning within particular rooms. Thus, the higher level control would dictate the sequence of rooms to visit such that the robot would be the most efficient, at every room invoking a lower level control that executes the cleaning process within rooms. Solving the overall cleaning problem by breaking it down into substeps and a multi-layer control scheme is the heart of hierarchical reinforcement learning.

Assume now that the room layout labels are given to the robot, i.e. the robot knows which rooms have the exact same layout, but does not know what the actual layout is (it has to learn this). One can imagine that if there are few possible room layouts, the higher level control need only invoke one of few low-level cleaning policies, optimal for each room layout that it's learned, thus making the learning a lot more efficient. Furthermore, consider two rooms A and B that are known to have the exact same layout. Experience accumulated in room A can be treated as experience learned in room B and vice versa, thus making a very strong case for data efficiency. One would expect the robot to approach an optimal hierarchical policy very quickly depending on how many repeated structures there are and how easy the low level control policies are to learn.

Given an MDP that can be broken down into substructures, a powerful and efficient framework for planning is the options model. When the only knowledge about our environment is that it has a hierarchical structure, we naturally arrive at a model-based algorithm: after every episode, we construct a model MDP based on our experiences, and run our options planning algorithm on it to extract a policy to be used for the next episode.

We establish the framework for our analysis of hierarchical reinforcement learning in the next few sections. Our HRL setup and preliminary results borrow from [40, 46, 46].

4.2 The HRL problem formulation

We now formalize the intuitions provided in the introductory section of this chapter. This section demonstrates that precisely, small and repeated substructures within the environment **are** the conditions for which HRL methods can be a lot more efficient than traditional RL methods.

Let $M = (\mathcal{S}, \mathcal{A}, P, r, s_e, s_0)$ be a finite-time horizon MDP where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, P is the transition function, and r is the reward model. We consider random rewards, that is, upon taking action a in state s, the agent receives a reward of $r(\cdot|s,a) \in [0,1]$. State s_e is an exit state, while state s_0 is a starting state, and we let $\bar{r}$ denote the expectation of r. The agent starts at state s_0 and plays the environment until state s_e is achieved, attempts to maximize its expected cumulative reward over T episodes:

$$\max_{\pi} \sum_{t=1}^{T} \mathbb{E}\left[\sum_{h=1}^{\tau_t} r_{th} | \pi, s_{t0} = s_0\right]$$

where τ_t is the random variable to denote the time at which the agent enters s_e during episode t. We assume that under any policy $\pi \in \Pi$, $\tau \leq \tau_{\max}$ with probability 1 and $\mathbb{E}[\tau] \leq H$ where h indexes the timesteps within an episode, and t indexes the episodes themselves. We also denote $r_{th} \sim r(\cdot|s_{th}, a_{th})$ the immediate reward obtained during episode t at level h. We now formalize subproblems as subMDPs.

Definition 4.2.1. Consider a partition of the non-terminal states $\mathcal{S} \setminus \{s_e\}$ into L disjoint subsets $\mathcal{L} = \{\mathcal{S}_i\}_{i=1}^{L}$. We define an induced subMDP $M_i = (\mathcal{S}_i \cup \mathcal{E}_i, \mathcal{A}, P_i, r_i, \mathcal{E}_i)$ as follows:

- $\mathcal{S}_i$ is the internal state set of the subMDP.
- $\mathcal{A}$ is the same action space.
- The exit state set $\mathcal{E}_i = \{e \in \mathcal{S} \setminus \mathcal{S}_i : \exists (s,a) \in \mathcal{S} \times \mathcal{A} \ s.t. \ P(e|s,a) > 0\}$.
- The state space of M_i is $\mathcal{S}_i \cup \mathcal{E}_i$.
- P_i and r_i are respectively the restriction of P and r to domains $\mathcal{S}_i \times \mathcal{A}$.

- The subMDP M_i terminates once it reaches a state in $\mathcal{E}_i$, that is, an exit state.

Given a partition $\mathcal{L} = \{\mathcal{S}_i\}_{i=1}^{L}$ of states in the MDP M, consider the set of induced subMDPs, $\{M_i\}_{i=1}^{L}$, define ρ the maximum size of any subMDP and $\mathcal{E}$ the set of all exit states as follows:

$$\rho = \max_i |\mathcal{S}_i \cup \mathcal{E}_i| \,, \text{ and } \mathcal{E} = \bigcup_{i=1}^{L} \mathcal{E}_i,$$

Each of the subMDPs M_i can be viewed as a subproblem of the original MDP in this way. We now concretely define an equivalence on subMDPs.

Definition 4.2.2. Let M_i and M_j be two subMDPs. We say that M_i and M_j are equivalent if and only if there exists a bijection $f : \mathcal{S}_i \cup \mathcal{E}_i \longrightarrow \mathcal{S}_j \cup \mathcal{E}_j$ such that $f(\mathcal{S}_i) = \mathcal{S}_j$, $f(\mathcal{E}_i) = \mathcal{E}_j$, and through f, the subMDPs have the same transition probabilities and rewards at internal states.

When subMDPs in $\mathcal{L}$ can be grouped via this subMDP equivalence relation, solutions to one subMDP can be re-used to solve every other subMDP equivalent to it. Let $K \leq L$ be the number of equivalence classes of subMDPs induced by a particular partition $\mathcal{L}$ of M. When there are no equivalences among the subMDPs, $K = L$. When there are repeated structures, $K < L$.

Traditional analyses of reinforcement learning algorithms are heavily dependent on $|\mathcal{S}|$, the size of the state space, or a proxy of it. In HRL, regret bounds and sample efficiency will now depend on M, the maximum size of a subMDP, K, the number of equivalence classes, and $\mathcal{E}$, the number of exit states. We expect that HRL algorithms would offer strong improvements over standard RL algorithms precisely when the original MDP can be decomposed into 1. many small subproblems and 2. few equivalence classes. This translates to $\rho K \ll |\mathcal{S}|$, where instead of learning policies or value functions over the entire state space, we learn at most K options over a maximum of ρ subMDP states per option, and recycle our options to equivalent MDPs.

Another important determining factor for improved performances of HRL is the number of exit states $|\mathcal{E}|$. These are the states that connect subMDPs to each other, and having a small number of exit states means having a small number of "bottleneck" states in M. Contributions [25, 35–37] demonstrate in part and/or in full that fewer bottleneck states enable more computationally efficient planning.

4.3 HRL with posterior sampling-driven exploration

The purpose of HRL is to make use of known hierarchical structures in the problem to improve the algorithmic efficiency of reinforcement learning agents. The control problem that we study is thus the following. The agent is to find a value-maximizing policy while operating in an unknown environment M. The agent is given a partition $\mathcal{L} = \{M_i\}_{i=1}^{L}$ of the MDP into subMDPs M_i without knowledge of the sub-transitions P_i's and sub-rewards r_i's. Finally, the agent has knowledge that the environment obeys a hierarchical structure, i.e. the agent is given the groupings of subMDPs (or sub-states) based on their equivalence classes (implicitly, the agent knows K and M).

Our algorithm applies posterior sampling to hierarchical reinforcement learning (PSHRL) to guide exploration, as the epistemic uncertainties about the environment is represented by a probability distribution over MDPs (we refer the reader to Chapter 3 for an exposition of posterior sampling based methods). As we will see, incorporating the knowledge of hierarchical structure within the environment into the prior distribution over the space of MDPs is a lot more natural in posterior sampling than having to compute very precise upper-confidence bounds that take into account the hierarchical structures. Furthermore, one can think of PSHRL as working with a posterior that is only supported on hierarchical MDPs that follow the given structure, thus reducing the model search component of our model-based algorithm to a smaller one within the space of MDPs. The following algorithm presents the most general form of the posterior sampling reinforcement learning algorithm (PSRL) (Algorithm 1 in [46]).

Algorithm 1: PSRL with a Planner, Sampler, and Inferer

```
Prior knowledge $\mathcal{P}^0$, planning algorithm plan, sampling algorithm sample, inference
  algorithm infer;
for episode $t$ = 1,2, ..., $T$ do
    $M^t \sim \text{sample}(\mathcal{P}^t)$;
    $\pi^t = \text{plan}(M^t)$;
    execute $\pi^t$ over episode $t$, observe $\mathcal{D}_t$;
    $\mathcal{P}^{t+1} = \text{infer}(\mathcal{P}^{\text{t}}, \mathcal{D}_{\text{t}})$
end
```

In the above, $\mathcal{P}^t$ encodes a posterior distribution over the space of MDPs, the sampling algorithm samples M^t from $\mathcal{P}^t$, the planner computes an optimal policy for M^t (e.g. via value iteration or policy iteration), and the inference algorithm applies Bayes' rule to update the posterior according to data from the most recent episode executed.

Posterior sampling hierarchical reinforcement learning essentially encodes the hierarchical structure directly into the posterior above. Ideally, a planner customized to handle environments with said hierarchical structure would be used instead of the traditional value iteration (VI) or policy iteration (PI).

Define the Bayesian regret of an algorithm after the execution of T episodes as:

Definition 4.3.1. For any learning algorithm alg, the Bayesian regret over the first T episodes is:

$$\text{BayesRegret}(\text{alg}, T) = \sum_{t=1}^{T} \mathbb{E}\left[V^*(s_0) - V^{\pi^t}(s_0)\right]$$

where V^* is the optimal value function for M, V^{π^t} is the value function for M under policy π^t, and π^t is the policy played by the agent at episode t.

Remark 4.3.2. Minimizing the Bayesian regret is equivalent to maximizing $\sum_{t=1}^{T} \mathbb{E}\left[V^{\pi^t}\right] = \sum_{t=1}^{T} \mathbb{E}\left[\sum_{h=1}^{\tau_t} r_{th}\right]$.

As opposed to the frequentist regret seen previously in Chapter 2, this regret averages out the stochasticity in the choice of policies for every episode t. This notion of regret is appropriate to the analysis of PSRL and PSHRL since much randomness arises from the episodic policies being derived from randomly sampled hierarchical MDPs every episode.

In Wen et. al. [46], the authors proved Bayesian regret bounds for PSHRL in terms of the hierarchical parameters ρ and K:

Theorem 4.3.3. (PSHRL Bayesian regret bound, Theorem 1 of [46]) If $\mathcal{P}^0$ exhibits hierarchical structure with a maximum of ρ states per subMDP and K subMDP equivalence classes, sample draws from the posterior distribution, and infer applies Bayes' rule, then:

$$\text{BayesRegret}(\text{PSHRL}, T) \leq T \cdot \mathbb{E}\left[V^*(s_0) - V^{\tilde{\pi}}(s_0)\right] + \mathcal{O}\left(H^{3/2}\rho\sqrt{K}\sqrt{\mathcal{A}T\log(\mathcal{A}KH\tau_{\max}T)}\right)$$

where $\tilde{\pi} = \text{plan}(M)$. Hiding logarithmic factors, the bound becomes $\mathbb{E}\left[V^*(s_0) - V^{\tilde{\pi}}(s_0)\right] + \tilde{\mathcal{O}}\left(H^{3/2}\rho\sqrt{K}\sqrt{\mathcal{A}T}\right)$.

Remark 4.3.4. Recall that the lower bound for reinforcement learning algorithms in the finite state and action space setting is $\Omega(\sqrt{H\mathcal{S}\mathcal{A}T})$. UCB-based methods and various posterior and optimistic sampling methods have been shown to enjoy regret bounds as close as $\tilde{\mathcal{O}}(H\mathcal{S}\sqrt{\mathcal{A}T})$ (more obscure simultaneous training schemes can gain a factor of $\sqrt{\mathcal{S}}$). In particular, the bound of PSRL presented in Osband et. al. (2013) [28] is precisely $\tilde{\mathcal{O}}(H^{3/2}\mathcal{S}\sqrt{\mathcal{A}T})$.

If we treat M as an entire MDP instead of considering the given partition $\mathcal{L}$, we would have that $K = L = 1$, and $\rho = \mathcal{S}$, from 4.3.3 we recover the bound proven in Osband et. al. (2013). It is clear that when $\rho\sqrt{K} << \mathcal{S}$, the PSHRL bound conveys strong improvements over the standard PSRL regret bound. As highlighted in [46], this improvement has two components:

1. A replacement of $\tilde{\mathcal{O}}(\sqrt{\mathcal{S}}) \to \tilde{\mathcal{O}}(\sqrt{\rho K})$, as the algorithm need only learn about a smaller number of distinct states in the hierarchical MDP.

2. A replacement of $\tilde{\mathcal{O}}(\sqrt{\mathcal{S}}) \to \tilde{\mathcal{O}}(\sqrt{\rho})$ because at each state-action pair in the hierarchical MDP, there is at most M states the agent could transition to (as ρ is the size of the largest subMDP equivalence class).

We also remark that the bound in 4.3.3 allows for sub-optimality in the planning operator. If $\tilde{\pi} = \pi^*$, then there is no sub-optimality in the planning algorithm. Otherwise, the sub-optimality contributes to a linear growth in the regret of the algorithm.

4.4 Noisy PSHRL: motivation

We consider once again our robot cleaner friend from the beginning of the chapter, except this time we loosen the strict assumption of "equivalent" rooms into "similar" rooms. This slight difference between the rooms could be the result of slightly misplaced furnitures. One would like to know to what extent can one still use a hierarchically-driven solution for non-hierarchical problems (assume the rooms are equivalent anyway and recycle learned options) over . Specifically, we would like to control the regret of the algorithm based on certain variables that either evoke a notion of similarity among subMDPs or control the amount of difference there are between similar subMDPs.

This situation naturally arises in the real world where often times, reward signals are captured by a sensor that may be naturally subjected to noise or to adversarial attacks. A reinforcement learning agent operating in a continuous environment might even make mistakes in estimating the transition function due to noisy observations and thus confusion arises on which state it was really transitioned to, inducing noise in the transition estimates. Furthermore, machine learning engineers might even compromise a bit of optimality for a lot of performance gains in the form of sample efficiency when applying a HRL algorithm to a near HRL problem. This next part of the thesis presents a novel characterization of noise in the context of hierarchical reinforcement learning and shows regret bounds of PSHRL when applying to near-hierarchical environments.

4.5 Characterizing noise

This section outlines the novel foundation for the analysis of noisy HRL. Assume that we are given an MDP M^* with a partition $\mathcal{L} = \{M_i\}_{i=1}^{L}$, as well as the information about K groupings of mutually similar subMDPs. Precisely, we are given $\mathcal{G} : \mathcal{M} \to [K]$ such that for any two similar subMDPs $M_i, M_j \in \mathcal{L}, \mathcal{G}(M_i) = \mathcal{G}(M_j)$. We begin with a structural assumption:

Assumption 4.5.1. Two subMDPs M_i, M_j are similar if there exists a bijection $f : \mathcal{S}_i \cup \mathcal{E}_i \to \mathcal{S}_j \cup \mathcal{E}_j$ such that $f(\mathcal{S}_i) = \mathcal{S}_j, f(\mathcal{E}_i) = \mathcal{E}_j$. Furthermore, $\forall (s, a) \in \mathcal{S}_i \times \mathcal{A}, \text{supp}(P_i(\cdot|s, a)) = \text{supp}(P_j(\cdot|f(s), a))$, that is, the transition functions share support but are not necessarily the same. The reward **distributions** r_i and r_j need not be the same.

This assumption loosens the strong equivalence relation into a notion of similarity that is structure preserving. This assumption formalizes the intuition that two MDPs are similar if the MDP graphs have the same vertices and edges when drawn out, but the non-zero transition probabilities as well as the rewards need not be the same. We now characterize these similarities precisely. We call the grouping of all subMDPs similar to each other a "similarity class".

Let $\kappa_{\mathcal{G}} : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ be such that $\kappa_{\mathcal{G}}(s)$ returns the set of all states similar to s given the grouping $\mathcal{G}$ and the implicit bijections from the previous assumption. We now define controls on the reward and transition function differences.

Definition 4.5.2. Let $k \in [K]$ denote similarity classes. For subMDPs M_i and M_j with $\mathcal{G}(M_i) = \mathcal{G}(M_j) = k$, let $f_{i,j}$ denote the bijection between M_i and M_j, we define:

$$\zeta_r^{(k)}(i, j) = \max_{(s,a)\in\mathcal{S}_i\times\mathcal{A}} |\hat{r}_i(s, a) - \hat{r}_j(f_{i,j}(s), a)|$$

$$\zeta_P^{(k)}(i, j) = \max_{(s,a)\in\mathcal{S}_i\times\mathcal{A}} \|P_i(\cdot|s, a) - P_j(\cdot|f_{i,j}(s), a)\|_1$$

as the upper bound to the difference in rewards and L1 norm of the transition probabilities for the same state-action pair. Let

$$\zeta_r^{(k)} = \max_{\{i,j:\mathcal{G}(M_i)=\mathcal{G}(M_j)=k\}} \zeta_r^{(k)}(i,j)$$
$$\zeta_P^{(k)} = \max_{\{i,j:\mathcal{G}(M_i)=\mathcal{G}(M_j)=k\}} \zeta_P^{(k)}(i,j)$$

be the upper bound to the difference in rewards and difference in L1 norm of the transition probabilities for the similarity class k. Finally, define:

$$\zeta_r = \max_{k\in[K]} \zeta_r^{(k)}, \ \ \zeta_P = \max_{k\in[K]} \zeta_P^{(k)}$$

the upper control for the variability of reward signals and transition probabilities over **all** similarity classes.

Intuitively, the values of ζ_r and ζ_P control how "un-hierarchical" the MDP M^* is given a partition $\mathcal{L}$ of the MDP as well as a similarity grouping function $\mathcal{G}$ with respect to its rewards and transitions respectively.

As seen previously, PSHRL can be understood to be sampling candidate MDPs on the smaller space of MDPs that possess the required hierarchical structure. As $t \to \infty$, the posterior distribution concentrates on a MDP with said hierarchical structure. Since in our noisy case M^* is not hierarchical, PSHRL will never be able to concentrate on it. Our algorithm instead concentrates on a surrogate hierarchical MDP M of M^*. In a sense, we use this surrogate MDP as a proxy to approximate our performance on the real near-hierarchical MDP (we use the term near-hierarchical MDP to denote MDPs whose ζ_r and ζ_P are small). We now formally define the hierarchical surrogate with respect to M^*.

Definition 4.5.3. (Hierarchical surrogate MDP) Let $M^* = (\mathcal{S}, \mathcal{A}, P^*, r^*, s_e, s_0)$ be an episodic near-hierarchical MDP with a subMDP partitioning $\mathcal{L} = \{M_i\}_{i=1}^{L}$ and a grouping of sub-

MDPs $\mathcal{G} : \mathcal{M} \to [K]$ into K similarity classes. Let $d \in \Delta(\mathcal{S})$ be any state distribution. We define the (d-)hierarchical surrogate MDP $M = (\mathcal{S}, \mathcal{A}, P, r, s_e, s_0)$ of M^* as follows:

- They share the same state and action space $\mathcal{S}$ and $\mathcal{A}$.
- The transition function for any pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ is a weighted average of transition functions for all pairs (v, a), with $v \in \kappa_{\mathcal{G}}(s)$ with respect to d, that is,

$$P(s'|s,a) = \sum_{v \in \kappa_{\mathcal{G}}(s)} P^*(s'|v,a) \frac{d(v)}{\sum_{w \in \kappa_{\mathcal{G}}(s)} d(w)}, \ \forall s, s' \in \mathcal{S}, a \in \mathcal{A}$$

- Similarly, the reward distribution for any pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ is the weighted average of reward distributions for all pairs (v, a) with $v \in \kappa_{\mathcal{G}}(s)$ with respect to d, that is:

$$r(s,a) = \sum_{v \in \kappa_{\mathcal{G}}(s)} r^*(v,a) \frac{d(v)}{\sum_{w \in \kappa_{\mathcal{G}}(s)} d(w)}, \ \forall s \in \mathcal{S}, a \in \mathcal{A}$$

- s_e and s_0 are the exit and starting states respectively.

Intuitively, we are forcing a unified structure for all similar subMDPs weighted by d via a d-averaging of the dynamics of the similar subMDPs. By construction, the hierarchical surrogate is necessarily hierarchical. If M^* is the original near-hierarchical MDP, we write M to denote the hierarchical surrogate. Analogous to the hierarchical case, let ρ denote the size of the largest similarity class.

We define a state visitation distribution as follows:

Definition 4.5.4. Let π be a policy. Then, the π-state visitation distribution is denoted by d^π and is given by:

$$d^\pi(s) = \sum_{i=0}^{\infty} \gamma^i \mathbb{P}\left(s_i = s | \pi\right)$$

that is, conditioned on the dynamics of π, $d^\pi(s)$ is the probability that the agent is in s at any given time.

Often times, reinforcement learning proofs rely on the Markov property of the MDP and that empirical samples from trajectories are i.i.d. of the true MDP. In the misspecified case, however, we no longer have this property. Consider two equivalent subMDPs M_i and M_j of some hierarchical MDP M. Then, an observed reward $r_i(s,a)$ in M_i is treated as an empirical observation from $r_j(f(s),a)$ and vice versa as they are both i.i.d. from the same reward distribution. This is what often is referred to as having better sample efficiency since one is learning the same empirical knowledge being in either s or $f(s)$. Consider now two similar subMDPs M_i and M_j. Let $s^i \in M_i$ and $s^j \in M_j$ be similar states as per $\kappa_{\mathcal{G}}$. During an episodic learning process, observations from either s^i or s^j are collected by the execution of per-episode policies π^t. Since our algorithm is hierarchical, the visitation probabilities to either s^i or s^j are complex, and the observations in those states are averaged according to a highly complicated and random distribution that depends on the sampling of the candidate per-episode MDPs M^t from which the algorithm runs the planning algorithm to obtain π^t. To circumvent this challenge, we make the following assumption.

Assumption 4.5.5. Assume that we have access to a d^{π^*}-sampling oracle that can sample a state $s \sim d^{\pi^*}$, a reward $r^*(s,a)$ from s and some action a, and sample from the transition $P^*(\cdot|s,a)$ to get the next state s'.

In this way, conditioned on our historical data being generated by the oracle, our observations for similar states s^i and s^j are i.i.d. in the d^{π^*}-hierarchical surrogate.

4.6 Online-offline data generation and the noisy PSHRL algorithm

The proposed model-based noisy PSHRL algorithm generates both online and offline data every episode. At the start of episode t, the agent uses offline data from all previous episodes to update the posterior distribution $\mathcal{P}^{t-1} \to \mathcal{P}^t$. With this posterior, the agent samples a hierarchical MDP $M^t \sim \text{sample}(\mathcal{P}^t)$, and produces $\pi^t = \text{plan}(\mathcal{M}^t)$. With this new policy, the agent runs π^t online on the true MDP M^* until termination at $s_e \in \mathcal{S}$ for τ_t timesteps, producing the sequence $\mathcal{O}^t = \{x_{t0}, a_{t0}, r_{t0}, x_{t1}, a_{t1}, r_{t1}, \ldots, x_{t\tau_t}, a_{t\tau_t}, r_{t\tau_t}, s_e\}$. It is with respect to this online trial done at every episode that the Bayes regret bound will be given in the next section. Note that the observations made during this online execution cannot be used to update the posterior distribution since the distribution of states are not i.i.d. with the d^{π^*} sampling oracle, and therefore the empirical hierarchical MDP models used to construct confidence sets for the proof do not necessarily concentrate on the hierarchical surrogate M of M^*. After the online run, the offline data for episode t is produced. At timestep $h \leq \tau_t$, the agent samples $s_{th} \sim d^{\pi^*}|_{\kappa_{\mathcal{G}}(x_{th})}$, that is, the distribution d^{π^*} restricted to the set of states equivalent to x_{th}, the state seen at timestep h during the online trial $\mathcal{O}^t$. In state s_{th}, the agent executes the same action $a_{th} = \pi^t(x_{th})$ chosen during the online trial, and records the observed action and next state into r_{th} and s'_{th}. Let $\mathcal{D}_t = \{(s_{th}, a_{th}, r_{th}, s'_{th})\}_{h=0}^{\tau_t}$ denote the offline data collected during episode t. Define the union of all offline data as $\mathbb{D}_t = \bigcup_{i=1}^{t} \mathcal{D}_t$, with which the algorithm maintains and updates the posterior distribution over hierarchical MDPs every episode.

Remark 4.6.1. By this way of firstly sampling $\mathcal{O}^t$, then sampling the offline data according to states observed in $\mathcal{O}^t$, we are ensuring that the offline observations with which the sampling mechanism samples a new MDP M^t every episode remains consistent with the target hierarchical surrogate. In this way, M^t and M are conditionally i.i.d. with respect to $\mathbb{D}_t$.

Algorithm 2: Noisy Offline-Online PSHRL with a Planner, Sampler, and Inferer

```
Input: Prior knowledge $\mathcal{P}^0$, functions plan, sample, infer;
Initialize $\mathcal{D}_0, \mathcal{D}'_0 = \emptyset, \mathbb{D}_0 = \emptyset$.
for episode $t = 1, 2, \ldots, T$ do
    $\mathcal{P}^t = \text{infer}(\mathcal{P}^{t\text{-}1}, \mathbb{D}_{t\text{-}1})$;
    $M^t \sim \text{sample}(\mathcal{P}^t)$;
    $\pi^t = \text{plan}(M^t)$;
    Run online trial with $\pi^t$, collect in $\mathcal{O}^t = \{x_{t0}, a_{t0}, r_{t0}, x_{t1}, a_{t1}, r_{t1}, \ldots, x_{t\tau_t}, a_{t\tau_t}, r_{t\tau_t}, s_e\}$;
    Generate:
     $\mathcal{D}_t = \big\{\big(s_{th} \sim d^{\pi^*}|_{\kappa_{\mathcal{G}}(x_{th})}, a_{th} \sim \pi^t(s_{th}), r_{th} \sim r^*(s_{th}, a_{th}), s'_{th} \sim P^*(\cdot|s_{th}, a_{th})\big)\big\}_{h=0}^{\tau_t}$;
    Collect $\mathbb{D}_t = \mathbb{D}_{t-1} \cup \mathcal{D}_t$;
end
```

4.7 Efficiency of noisy PSHRL

Define the optimality difference between MDPs Ψ as follows:

$$\Psi(M, M') = \mathbb{E}\left[V^{M,\pi} - V^{M',\pi'}\right]$$

where $\pi = \text{plan}(M)$ and $\pi' = \text{plan}(M')$. In the misspecified or PSHRL setting with the additional Assumptions 4.5.1 and 4.5.5, the offline-online PSHRL algorithm (**Algorithm 2**) enjoys a Bayes regret bound of:

$$\begin{aligned}\text{BayesRegret}(T) &= \mathcal{O}\left(T(\Psi(M^*, M) + \zeta_r + H\zeta_P)\right) + \mathcal{O}\left(H^{3/2}\rho\sqrt{K\mathcal{A}T\log(\mathcal{A}KH\tau_{\max}T)}\right) \\ &= \tilde{\mathcal{O}}\left(T(\Psi(M^*, M) + \zeta_r + H\zeta_P) + H^{3/2}\rho\sqrt{K\mathcal{A}T}\right)\end{aligned}$$

As with most misspecification cases, we recover a T-linear dependency of our regret bound on hierarchical parameters ζ_r and ζ_P which capture the degree of misspecification. We refer the reader to Chapter 5 for a complete proof of this bound.

We can control $\Psi(M^*, M)$ by using the L1 deviation between the uniform distribution over $\kappa_{\mathcal{G}}(s)$ and the distribution d^{π^*} restricted on $\kappa_{\mathcal{G}}(s)$ to further specify the order of $\Psi(M^*, M)$ from the construction of the hierarchical surrogate with respect to d^{π^*}. However small this quantity may be, it cannot be sub-linear in T as Ψ is temporally invariant. As long as the gap exists, it will necessarily incur a T-linear regret term in the Bayes regret bound.

We conjecture a correlation between ζ_r, ζ_P and $\Psi(M^*, M)$. The smaller the parameters controlling how un-hierarchical the near-hierarchical MDP M^* are, the more likely its hierarchical surrogate M will "resemble". Conditioned on the conjecture being true, a small-enough gap makes the noisy PSHRL algorithm a reasonable choice for near-hierarchical environment learning and solving, as all T-linear terms would be effectively small. Considering the sample efficiency of the algorithm, it could be that the optimal policy for the hierarchical surrogate is attained long before the scaling in T of $\Psi(M^*, M)$ and ζ_r, ζ_P takes a noticeable effect on the performance of the algorithm. As the bounds are given for Bayes regret, this could translate to the algorithm finding a π such that it performs optimally for most of the time on M^* while completely failing at some edge cases with very low probability of reaching.

4.8 Efficient planning with hierarchical structures

Our proof works for any planning algorithm plan to plan on the sampled M^t every episode. Our bound is general even in the case where there are no repeated hierarchical similarities, that is, $K = L = 1$, where it is of note that our algorithm vastly increases in efficiency when there are hierarchical similarities present, evident from the bound and from the informal condition $\rho K << |\mathcal{S}|$. As stated previously, a standard planning algorithm such as value iteration (VI) works as part of our PSHRL training loop, however we will look into a specific planning method specifically designed to leverage the hierarchical structures in the MDP for improved efficiency. This problem has been explored in [40] in the framework of options by using option models. If options are thought to be solutions to subproblems within the MDP coupled with subgoals, additional rewards associated with particular states in the MDP, then hierarchical planning can be thought of as constructing options corresponding to subMDPs. In this way, one needs to consider the possible combinations of values associated with the exit states of subMDPs. Taking the cleaning robot as an example, one can make each possible door in a room a subgoal by giving them a high reward. An option can now be trained inside the room (within the subMDP) given the additional bonuses attributed to exit states. This intuition is formalized through the notion of exit profiles [40, 46].

Definition 4.8.1. An exit profile J for subMDP M_i is a vector of values $J(e), \forall e \in \mathcal{E}_i$.

The structure outside the subMDP is summarized through J for the agent. Any exit profile J is associated an optimal policy $\pi_{i,J}$ for the subMDP M_i which we will think of as an option. Furthermore, the same exit profile J induces the same option for all equivalent subMDPs.

Assume now that we have given to each equivalent subMDP class k a **set** of exit profiles $\tilde{\mathcal{J}}_k$ and corresponding options for each $J \in \tilde{\mathcal{J}}_k$ (this can be computed via traditional VI). We define an induced high-level MDP $M_G = \left(\mathcal{S}_G, \mathcal{A}_G, P^G, r^G\right)$ such that

- $\mathcal{S}_G = \mathcal{E} \cup \{s_0\}$, the union of all exit states with the starting state.
- For each $s \in \mathcal{S}_G$, if s is a state in subMDP M_i, then its action space $\mathcal{A}_G(s)$ is the set of options computed for M_i.
- $r^G(s, \pi_{i,J})$ is the expected reward obtained from $s \in \mathcal{S}_i$ when $\pi_{i,J}$ is run until the option reaches $e \in \mathcal{E}_i$.
- $P^G(e|s, \pi_{i,J})$ gives the probability of transitioning to $e \in \mathcal{E}_i$.

The quantities $r^G(s, \pi_{i,J})$ and $P^G(e|s, \pi_{i,J})$ form the option model for $\pi_{i,J}$, defined as in [40]. We remark here that the exit state of a subMDP is in fact a starting state for another subMDP from the way subMDP partitioning was defined. P^G essentially encodes the probability of hopping from one particular subMDP to another. Solving the higher-level MDP by computing options and option models corresponding to sets of exit profiles is called Planning with Exit Profiles (PEP) [46]:

Algorithm 3: Planning with Exit Profiles (PEP)

```
MDP $M$, $k$ sets of exit profiles $\tilde{\mathcal{J}}_k$, one for each equivalent subMDP class $k$;
**Step 1: Option generation**
  **for** $k$ = 1, 2, ..., $K$ **do**
    For each exit profile $J \in \tilde{\mathcal{J}}_k$, compute $\pi_{k,J}$ for subMDPs in equiv. class $k$, and its
      associated model
**end**
**Step 2: plan with options**
  Compute a policy for the induced high-level $M^G$, which induces a policy $\tilde{\pi}$ for $M$.;
**Return**: $\tilde{\pi}$.
```

We use VI to both generate options for each exit profile as well as plan on the high-level MDP M^G. For this process to terminate in a finite number of steps, the following assumption is made:

Assumption 4.8.2. For M, all its induced subMDPs with exit profiles, and the induced M^G, the transition probability graph corresponding to an optimal policy is acyclic.

VI outputs a value function in n iterations under a proper initialization where n is the cardinality of the state space.

Now, if in Algorithm 1 we compute $\pi^t = \text{plan}(M^t)$ with VI, we would have that the computational complexity of VI is $O(\mathcal{S}^2\mathcal{A}\rho)$. On the other hand, denote $X = \max_k |\tilde{\mathcal{J}}_k|$, the computational complexity of PEP is:

$$\mathcal{O}(KX\rho^2\mathcal{A}\rho) + \mathcal{O}(\mathcal{E}^2X\rho) \leq \mathcal{O}\left(X\left[K\rho^2\mathcal{A} + \mathcal{E}^2\right]\rho\right)$$

Where $\mathcal{O}(KX\rho^2\mathcal{A}\rho)$ is the complexity of the option generation process, and $\mathcal{O}(\mathcal{E}^2X\rho)$ is the complexity of planning on the high-level MDP M^G. PEP will be efficient if $X\rho^2K < \mathcal{O}(\mathcal{S}^2)$ and $\mathcal{E}^2X < \mathcal{O}(\mathcal{S}^2A)$, that is, when subMDPs are small and a small number of exit profiles is used to find options for each equivalent subMDP class, respectively. Furthermore, we desire the total number of exit states $\mathcal{E}$ to be small as well similar to what was mentioned in previous sections, where exit states are bottlenecks to efficient learning.

4.9 Exit profiles: quality and suboptimality

From the perspective of an agent navigating a subMDP, the outer MDP structure is summarized through exit profiles. If these exit profiles are not representative of value maximizing behavior beyond the subMDP, then the options generated in Step 1 of Algorithm 3 are not optimal for the control problem of the entirety of the environment. The performance of a PSHRL algorithm using PEP will depend on how good the given exit profiles are, which will be characterized by a notion of "quality" for exit profiles. Instead of explicitly manufacturing exit profiles from organically compiling the outside environment's information into vectors (this requires full knowledge of the environment which in a control setting, the agent does not have access to), PEP instead selects an ϵ-cover of exit profiles such that options generated by these are near optimal for any possible exit profile.

Let $\mathcal{J}_i \subset [0, H]^{|\mathcal{E}_i|}$ be the space of exit profiles for subMDP M_i, $V^{\pi}_{i,J}$ the value of policy π for exit profile J in subMDP M_i, and $V^{*}_{i,J}$ the value of the optimal policy of M_i w.r.t. exit profile J.

Definition 4.9.1. The suboptimality of a set of exit profiles $\tilde{\mathcal{J}}$ for M_i is

$$\Delta_i(\tilde{\mathcal{J}}) = \max_{s \in \mathcal{S}_i^0, J \in \mathcal{J}_i} \left(V^{*}_{i,J}(s) - \max_{\tilde{J} \in \tilde{\mathcal{J}}} V^{\pi_{i,\tilde{J}}}_{i,J}(s) \right)$$

where $\mathcal{S}_i^0$ is the possible start states of M_i.

Essentially, $\Delta_i(\tilde{\mathcal{J}})$ captures the ability of optimal policies for exit profiles within $\tilde{\mathcal{J}}$ to approach optimality in any exit profile. For any exit profile J, there exists some $\tilde{J} \in \tilde{\mathcal{J}}$ that induces a $\Delta_i(\tilde{\mathcal{J}})$-optimal policy under J (that is, under the subMDP M_i with exit profile J). Let $\Delta = \max_i \Delta_i(\tilde{\mathcal{J}}_{k_i})$ where k_i is the equivalence class M_i is in, and $\tilde{\mathcal{J}}_k$'s are sets of exit profiles given to each equivalence class in Step 1 of Algorithm 3. Δ is the worst possible gap in subMDP value among all equivalence classes. Authors in [46] proved the following:

Proposition 4.9.2. If in Step 2 of PEP (Algorithm 3), the agent uses VI with initialization $V = 0$ to compute a high-level policy $\tilde{\pi}$ for M^G, then under Assumption 4.8.2, we have that $V^*(s_0) - V^{\tilde{\pi}}(s_0) \leq \Delta|\mathcal{E}|$.

That is, when both the exit profiles given to PEP are of high quality (Δ small) and there is a small number of possible exit states ($\mathcal{E}$ small), then the value function of the high-level policy $\tilde{\pi}$ that is computed by PEP is close to the optimal value function. We control Δ by selecting aforementioned ϵ-covers of exit profiles.

Definition 4.9.3. A finite set $\tilde{\mathcal{J}}$ is a (sup norm) ϵ-cover for $\mathcal{J}$ if for any $J \in \mathcal{J}$, there exists a $\tilde{J} \in \tilde{\mathcal{J}}$ such that $\left\| J - \tilde{J} \right\|_\infty < \epsilon$.

We have the following result per [46]:

Proposition 4.9.4. For subMDP M_i in equivalence class k, if $\tilde{\mathcal{J}}_k$ is an ϵ-cover of $\mathcal{J}_i \subset [0, H]^{\mathcal{E}_i}$, then $\Delta_i(\tilde{\mathcal{J}}_k) \leq 2\epsilon$.

As $\mathcal{J}_i \subset [0, H]^{\mathcal{E}_i}$ is bounded, there always exists a finite ϵ-cover $\tilde{\mathcal{J}}_k$ of $\mathcal{J}_i$ such that $\left|\tilde{\mathcal{J}}_k\right| \leq \lfloor H/\epsilon \rfloor^{|\mathcal{E}_i|}$ exponential in $\mathcal{E}_i$, only computationally efficient when $\max_i \mathcal{E}_i$ is very small (when for every subMDP, there are only a very small number of possible exit states).

In terms of regret, PSHRL with a PEP planner suffers a regret of $\Delta \mathcal{E} T + \tilde{\mathcal{O}}(H^{3/2}\rho\sqrt{K}\sqrt{\mathcal{A}T})$ which is an additional linear temporal dependency on $\Delta\mathcal{E}$. Controlling ϵ in selecting an ϵ-cover essentially allows for the Δ_i's to negligible amounts in the value function estimates. They are, however, non-negligible, as they induce an at most constant suboptimal difference every timestep. All in all, for the price of a somewhat controllable suboptimality term linear in T, when $X\rho^2 K < \mathcal{O}(\mathcal{S}^2)$ and $\mathcal{E}^2 X < \mathcal{O}(\mathcal{S}^2\mathcal{A})$, one observes significant gains in computational efficiency.

Chapter 5

Regret analysis of Algorithm 2

We begin with a regret decomposition into manageable parts, of which we will control individually.

5.1 Regret decomposition

Recall that M^*, π^* are the true MDP and its optimal policy. Given the state distribution d^{π^*} and the similarities among subMDPs, we construct the hierarchical surrogate M, and label $\pi = \text{plan}(M)$. ζ_r and ζ_P are controls over the width of the reward functions and transition functions in similar subMDPs. We have that:

$$\begin{aligned}
\text{BayesRegret}(T) &= \sum_{t=1}^{T} \mathbb{E}\left[V^{M^*,\pi^*} - V^{M^*,\pi^t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[V^{M^*,\pi^*} - V^{M,\pi}\right] + \sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi} - V^{M,\pi^t}\right] + \sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi^t} - V^{M^*,\pi^t}\right] \\
&= T\Psi(M^*, M) + \sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi} - V^{M,\pi^t}\right] + \sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi^t} - V^{M^*,\pi^t}\right]
\end{aligned}$$

We can loosely control the first summation by $T\Psi(M^*, M) \leq TH$. Consider $V^{M,\pi^t} - V^{M^*,\pi^t}$:

$$\begin{aligned} V^{M,\pi^t} - V^{M^*,\pi^t} &\leq |\hat{r}^M(s,\pi^t(s)) - \hat{r}^{M^*}(s,\pi^t(s))| \\ &\quad + \sum_{s' \in \mathcal{S}} \left(P^M(s'|s,\pi^t(s)) V^{M,\pi^t}(s') - P^{M^*}(s'|s,\pi^t(s)) V^{M^*,\pi^t}(s') \right) \\ &\leq \zeta_r + H\zeta_P \end{aligned}$$

Therefore,

$$\sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi^t} - V^{M^*,\pi^t} \right] \leq T\zeta_r + TH\zeta_p$$

Consider finally $V^{M,\pi} - V^{M,\pi^t}$. Recall that $\pi^t = \text{plan}(M^t)$ with M^t sampled from the posterior distribution. Here, we make a crucial observation that is central to our analysis. We use our sampling oracle to sample rewards and transitions from states according to the distribution d^{π^*} to build our dataset, which is used in turn to sample an MDP M^t for episode t. As the hierarchical surrogate M is constructed by weighing similar transitions and rewards exactly according to d^{π^*}, our empirical estimates of the rewards and transitions from our dataset are exactly i.i.d. estimates for the transitions and rewards in M. At the start of teach episode t, conditioned on the dataset $\mathbb{D}_t$, M^t and M are independent and identically distributed. As plan : $\mathcal{M} \to \Pi$ is a deterministic mapping, it follows that the tuples (M, π) and (M^t, π^t) are i.i.d., thus $\mathbb{E}\left[V^{M,\pi} | \mathbb{D}_t\right] = \mathbb{E}\left[V^{M^t,\pi^t} | \mathbb{D}_t\right]$ and by the tower property, $\mathbb{E}\left[V^{M,\pi}\right] = \mathbb{E}\left[V^{M^t,\pi^t}\right]$. Thus, we have:

$$\sum_{t=1}^{T} \mathbb{E}\left[V^{M,\pi} - V^{M,\pi^t} \right] = \sum_{t=1}^{T} \mathbb{E}\left[V^{M^t,\pi^t} - V^{M,\pi^t} \right]$$

For any policy π and any MDP M, denote the Bellman consistency operator by $\mathcal{T}^{M,\pi}$, and we have $V^{M,\pi} = \mathcal{T}^{M,\pi} V^{M,\pi}$. We decompose the per-episode regret into per timestep Bellman errors using the Bellman consistency operator. Similarly to [28], we have:

$$\begin{aligned}
\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)(s_{t0}) &= (\mathcal{T}^{M^t,\pi^t}V^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}V^{M,\pi^t})(s_{t0}) \\
&= (\mathcal{T}^{M^t,\pi^t}V^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}V^{M,\pi^t})(s_{t0}) \pm \mathcal{T}^{M,\pi^t}V^{M^t,\pi^t}(s_{t0}) \\
&= \left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{t0}) + \sum_{s'\in\mathcal{S}} P^M(s'|s_{t0},\pi^t(s_{t0}))\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)(s') \\
&\quad \pm \left(V^{M^t,\pi^t} - V^{M,\pi^t}\right) \\
&= \left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{t0}) + \left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)(s_{t1}) + d_{t0} \\
&= \dots \\
&= \sum_{h=0}^{\tau_t-1}\left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{th}) + \sum_{h=0}^{\tau_t-1} d_{th}
\end{aligned}$$

where:

$$d_{th} = \sum_{s'\in\mathcal{S}} P^M(s'|s_{th},\pi^t(s_{th}))\left(V^{M^t,\pi^t}(s') - V^{M,\pi^t}(s')\right) - \left(V^{M^t,\pi^t}(s_{t,h+1}) - V^{M,\pi^t}(s_{t,h+1})\right)$$

The first factor is the one-step Bellman error under the sampled MDP M^t. The second term represents the randomness in the transitions of the hierarchical surrogate M. In state s_{th} under the policy π^t, the expected value of $\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)(s_{t,h+1})$ is exactly $\sum_{s'\in\mathcal{S}} P^M(s'|s_{th},\pi^t(s_{th}))\left(V^{M^t,\pi^t}(s') - V^{M,\pi^t}(s')\right)$. Similarly to [28], conditioned on the hierarchical surrogate M and the sampled MDP M^t, the term $\sum_{h=0}^{\tau_t-1} d_{th}$ has expectation zero. We thus have that:

$$\mathbb{E}\left[V^{M^t,\pi^t} - V^{M,\pi^t}|M^t, M\right] = \sum_{h=0}^{\tau_t-1}\mathbb{E}\left[\left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{th})|M^t, M\right]$$

where we decomposed the per-episode expected regret (conditioned on M^t, M) into per-timestep Bellman errors (conditioned on M^t, M). We now show that with high probability, M^t exists in a confidence set concentrating on the hierarchical surrogate M. The construction of these confidence sets will give us the controls for the per-timestep Bellman errors. From this point onward, our analysis largely borrows from [46].

5.2 Construction of confidence sets

For each episode t, let $\mathcal{N}^t(s,a)$ denote the number of times action a was taken while either being in s or in states similar to s, that is, any state in $\kappa_{\mathcal{G}}(s)$, in the first t offline episodes. In other words, $\mathcal{N}^t(s,a)$ count the occurrences of $\kappa_{\mathcal{G}}(s)$ and a in $\mathbb{D}_{t-1}$, and gets updated at the end of the episode when we update $\mathbb{D}_{t-1} \longrightarrow \mathbb{D}_t$. Let $\hat{P}^t(\cdot|s,a)$ and $\hat{r}^t(s,a)$ be respectively the empirical transition function and the empirical reward model based on offline observations in the first t episodes of choosing action a at state s **or states similar to** s. If $\mathcal{N}^t(s,a) = 0$, we choose $\hat{r}^t(s,a)$ arbitrarily in $[0,1]$ and $\hat{P}^t(\cdot|s,a)$ as an arbitrary distribution subject to the constraint that $\hat{P}^t(s'|s,a) > 0$ only if s' and s are in the same subMDP.

For episode t, the confidence set $\mathbb{M}_t$ is given by:

$$\mathbb{M}_t = \Bigg\{ \tilde{M} : \left\| \hat{P}^t(\cdot|s,a) - P^{\tilde{M}}(\cdot|s,a) \right\|_1 \leq \beta_1\left(\mathcal{N}^t(s,a), t\right) \ \forall s,a,$$
$$\left| \hat{r}^t(s,a) - \hat{r}^{\tilde{M}}(s,a) \right| \leq \beta_2\left(\mathcal{N}^t(s,a), t\right) \ \forall s,a,$$
$$\text{and } \tilde{M} \text{ satisfies the equivalent subMDP restriction} \Bigg\}$$

Selecting β_1 and β_2 such that M and $M^t \in \mathbb{M}_t$ with high probability requires the following concentration lemmas, stated without proof.

Lemma 5.2.1. (Weissman et. al., 2003) [45] Assume $p(\cdot)$ is a distribution over m distinct events and $\hat{p}(\cdot)$ is an empirical distribution for p from n i.i.d. samples. For any $\epsilon > 0$, we have

$$\mathbb{P}\left(\|p(\cdot) - \hat{p}(\cdot)\|_1 \geq \epsilon\right) \leq (2^m - 2)\exp\left(-\frac{n\epsilon^2}{2}\right)$$

Lemma 5.2.2. (Hoeffding, 1963) [16] For the deviation between the true mean $\bar{r}$ and the empirical mean $\hat{r}$ from n i.i.d. samples with support in $[0,1]$, for any $\epsilon > 0$, we have

$$\mathbb{P}\left(|\bar{r} - \hat{r}| \geq \epsilon\right) \leq 2\exp\left(-2n\epsilon^2\right)$$

Similar to [4], the lemma below gives precise expressions for β_1 and β_2. Recall that $\tau_{\max}$ is such that $\forall t, \mathbb{P}(\tau_t > \tau_{\max}) = 0$.

Lemma 5.2.3. For any $\delta \in (0,1)$, let $\beta_1(n,t) = \left(\frac{14\rho\log\left(\frac{2\mathcal{A}K\tau_{\max}t}{\delta}\right)}{\max(1,n)}\right)^{1/2}$, and $\beta_2(n,t) = \left(\frac{7\log\left(\frac{2\rho\mathcal{A}K\tau_{\max}t}{\delta}\right)}{2\max(1,n)}\right)^{1/2}$, we have that

$$\mathbb{P}(M \notin \mathbb{M}_t) = \mathbb{P}(M^t \notin \mathbb{M}_t) = \frac{\delta}{15t^6}$$

Proof. (of **Lemma 5.2.3**) Recall that ρ is the maximum size of the subMDPs, therefore upper bounds the size of the support of any transition distribution. Thus, we can use **Lemma 5.2.1** with $m = \rho$. Setting

$$\epsilon = \left(\frac{2\log\left(\frac{2^\rho 20\rho\mathcal{A}K\tau_{\max}t^7}{\delta}\right)}{\max(1,n)}\right)^{1/2} \leq \left(\frac{14\rho\log\left(\frac{2\mathcal{A}K\tau_{\max}t}{\delta}\right)}{\max(1,n)}\right)^{1/2} = \beta_1(n,t)$$

we must have that by Weissman's inequality,

$$\begin{aligned}
\mathbb{P}\left(\left\|\hat{P}^t(\cdot|s,a) - P^M(\cdot|s,a)\right\|_1 \geq \beta_1(n,t)\middle| M, n \text{ i.i.d. samples}\right) &\leq (2^\rho - 2)\exp\left(-\frac{n\beta_1(n,t)^2}{2}\right) \\
&\leq 2^\rho \exp\left(-\frac{n\epsilon^2}{2}\right) \\
&= 2^\rho \exp\left(-\frac{n}{2}\frac{2}{\max(1,n)}\log\left(\frac{2^\rho 20\rho\mathcal{A}K\tau_{\max}t^7}{\delta}\right)\right) \\
&\leq 2^\rho \frac{\delta}{2^\rho 20\rho\mathcal{A}K\tau_{\max}t^7} \\
&= \frac{\delta}{20t^7\rho\mathcal{A}K\tau_{\max}}
\end{aligned}$$

Similarly, set

$$\epsilon = \left(\frac{\log\left(\frac{120\rho\mathcal{A}K\tau_{\max}t^7}{\delta}\right)}{2\max(1,n)} \right)^{1/2} \leq \left(\frac{7\log\left(\frac{2\rho\mathcal{A}K\tau_{\max}t}{\delta}\right)}{2\max(1,n)} \right)^{1/2} = \beta_2(n,t)$$

we have that by Hoeffding's inequality,

$$\begin{aligned}
\mathbb{P}\Big(\left|\hat{r}^t(s,a) - \bar{r}^M(s,a)\right| \geq \beta_2(n,t) \Big| M, n \text{ i.i.d. samples} \Big) &\leq 2\exp\left(-2n\beta_2(n,t)^2\right) \\
&\leq 2\exp\left(-2n\epsilon^2\right) \\
&= 2\exp\left(-2n \cdot \frac{\log\left(120\rho\mathcal{A}K\tau_{\max}t^7/\delta\right)}{2\max(1,n)}\right) \\
&\leq 2\exp\left(-\log(120\rho\mathcal{A}K\tau_{\max}t^7)/\delta)\right) \\
&= \frac{\delta}{60^7\rho\mathcal{A}K\tau_{\max}}
\end{aligned}$$

For episode t, $\mathcal{N}^t(s,a) \in \{0,1,\ldots,t\cdot(\tau_{\max}-1)\}$ since there has been t episodes and s (and its similar states in $\kappa_{\mathcal{G}}(s)$) could've been attained at most $(\tau_{\max}-1)$ times since the episode always terminates on s_e, and each episode generates $\mathcal{D}_t$ offline data of max length $\tau_{\max}$. When $n=0$ (we do not have any i.i.d. samples observed), the confidence intervals trivially hold with probability 1. Union bounding over the possible values for $\mathcal{N}^t(s,a)$ gives:

$$\mathbb{P}\Big(\left\|\hat{P}^t(\cdot|s,a) - P^M(\cdot|s,a)\right\|_1 \geq \beta_1(\mathcal{N}^t(s,a),t) \Big| M \Big) \leq \sum_{n=1}^{t\cdot\tau_{\max}} \frac{\delta}{20t^7\rho\mathcal{A}\tau_{\max}K} < \frac{\delta}{20t^6\rho\mathcal{A}K}$$

and

$$\mathbb{P}\left(\left|\hat{r}^t(s,a) - \bar{r}^M(s,a)\right| \geq \beta_2(\mathcal{N}^t(s,a),t) \Big| M \right) \leq \sum_{n=1}^{t\cdot\tau_{\max}} \frac{\delta}{60^7\rho\mathcal{A}\tau_{\max}K} < \frac{\delta}{60t^6\rho\mathcal{A}K}$$

Since there are K unique subMDP similiarity classes and for each of these classes, a maximum of ρ states, there are at most ρK similar state classes. We can union bound over the actions

$\mathcal{A}$ and the equivalent state classes as follows:

$$\begin{aligned}\mathbb{P}\left(M \notin \mathbb{M}_t|M\right) &< \sum_{1}^{\rho K}\sum_{1}^{\mathcal{A}}\left[\frac{\delta}{60t^6\rho\mathcal{A}K} + \frac{\delta}{20t^6\rho\mathcal{A}K}\right] \\ &= \rho\mathcal{A}K\left[\frac{\delta}{60t^6\rho\mathcal{A}K} + \frac{\delta}{20t^6\rho\mathcal{A}K}\right] \\ &= \frac{\delta}{15t^6}\end{aligned}$$

Since the above hold for any M, we have

$$\mathbb{P}(M \notin \mathbb{M}_t) = \int_M \mathbb{P}(M)\mathbb{P}(M \notin \mathbb{M}_t|M) < \frac{\delta}{15t^6}$$

Finally, given that M and M^t are conditionally i.i.d. on the offline data $\mathbb{D}_t$, we have

$$\begin{aligned}\mathbb{P}(M^t \notin \mathbb{M}_t) &= \int_{\mathbb{D}_t} \mathbb{P}(\mathbb{D}_t)\mathbb{P}(M^t \notin \mathbb{M}_t|\mathbb{D}_t) \\ &= \int_{\mathbb{D}_t} \mathbb{P}(\mathbb{D}_t)\mathbb{P}(M \notin \mathbb{M}_t|\mathbb{D}_t) \\ &= \mathbb{P}(M \notin \mathbb{M}_t)\end{aligned}$$

■

5.3 Controlling the per-timestep Bellman errors

Recall that we seek to upper bound the expression $\sum_{t=1}^{T}\mathbb{E}\left[V^{M^t,\pi^t} - V^{M,\pi^t}\right]$. Consider:

$$\sum_{t=1}^{T}\mathbb{E}\left[V^{M^t,\pi^t} - V^{M,\pi^t}\right] < \sum_{t=1}^{T}\mathbb{E}\left[\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)\mathbb{1}\left[M, M^t \in \mathbb{M}_t\right]\right] + 2H\sum_{t=1}^{T}\mathbb{P}(M \notin \mathbb{M}_t)$$

where the desirable event is that both M and M_t are within our confidence region $\mathbb{M}_t$. The magnitude of the undesirable events are kept as H as $\mathbb{E}\left[V^{M,\pi}\right] \leq H$ for any M, π, since $\mathbb{E}\left[\tau\right] \leq H$ and r is supported on $[0,1]$. We take into account both bad events $M \notin \mathbb{M}_t$ and $M^t \notin \mathbb{M}_t$ as these have equal probabilities from **Lemma 5.2.3**, double counting the intersection event $M \wedge M^t \notin \mathbb{M}_t$. Choose $\delta = 1/H$, we have:

$$2H\sum_{t=1}^{T}\mathbb{P}(M \notin \mathbb{M}_t) < 2H\sum_{t=1}^{T}\frac{1}{15Ht^6} \leq \frac{2}{15}\sum_{t=1}^{\infty}\frac{1}{t^2} < \frac{1}{3}$$

On the other hand,

$$\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}\left[\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)\mathbb{1}\left[M, M^t \in \mathbb{M}_t\right]\right] \\
=&\sum_{t=1}^{T}\left\{\mathbb{E}\left[\sum_{h=0}^{\tau_t-1}\left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{th})|M^t, M\right]\mathbb{1}\left[M, M^t \in \mathbb{M}_t\right]\right\}
\end{aligned}$$

When $M, M^t \in \mathbb{M}_t$, we have that

$$\begin{aligned}
\left(\mathcal{T}^{M^t,\pi^t} - \mathcal{T}^{M,\pi^t}\right)V^{M^t,\pi^t}(s_{th}) \leq& \left|\bar{r}^{M^t}(s_{th}, \pi^t(s_{th})) - \bar{r}^{M}(s_{th}, \pi^t(s_{th}))\right| \\
&+ \left\|P^{M^t}(\cdot|s_{th}, \pi^t(s_{th})) - P^{M}(\cdot|s_{th}, \pi^t(s_{th}))\right\|_1\left\|V^{M^t,\pi^t}\right\|_\infty \\
&\leq 2\beta_2(\mathcal{N}^t(s_{th}, a_{th}), t) + 2H\beta_1(\mathcal{N}^t(s_{th}, a_{th}), t)
\end{aligned}$$

Thus,

$$\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}\left[\left(V^{M^t,\pi^t} - V^{M,\pi^t}\right)\mathbb{1}\left[M, M^t \in \mathbb{M}_t\right]\right] \\
&\leq 2\sum_{t=1}^{T}\mathbb{E}\left[\sum_{h=0}^{\tau_t-1}\left[\beta_2(\mathcal{N}^t(s_{th}, a_{th}), t) + H\beta_1(\mathcal{N}^t(s_{th}, a_{th}), t)\right]\right]
\end{aligned}$$

Upper bounding t by T, and substituting $\delta = 1/H$, we have that:

$$\begin{aligned}
\beta_2(\mathcal{N}^t(s_{th}, a_{th}), t) + H\beta_1(\mathcal{N}^t(s_{th}, a_{th}), t) \leq& \sqrt{\frac{7\log(2\rho\mathcal{A}KH\tau_{\max}T)}{2\max(1, \mathcal{N}^t(s_{th}, a_{th}))}} + H\sqrt{\frac{14\rho\log(2\mathcal{A}K\tau_{\max}T)}{\max(1, \mathcal{N}^t(s_{th}, a_{th}))}} \\
&\leq \mathcal{O}\left(H\sqrt{\frac{\rho\log(\mathcal{A}KH\tau_{\max}T)}{\max(1, \mathcal{N}^t(s_{th}, a_{th}))}}\right)
\end{aligned}$$

5.4 Self-normalization bounds

For this section, states denoted by s and its variants are understood to be representing all states in $\kappa_{\mathcal{G}}(s)$. We adopt the nomenclature of "similarity state-action pair" to specify when s in (s, a) represents itself and all states similar to it.

We seek to control the following expression:

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{1}{\max(1, \mathcal{N}^t(s_{th}, a_{th}))}}\right]$$

We define a special identity function that identifies similar state classes. Let $\mathrm{Id} : (\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A}) \longrightarrow \{0, 1\}$ be such that for any arbitrary $(s, a) \in \mathcal{S} \times \mathcal{A}$, $\mathrm{Id}((s', a'), (s, a))$ returns 1 only if $s' \in \kappa_{\mathcal{G}}(s)$ and $a' = a$. Trivially, $s \in \kappa_{\mathcal{G}}(s)$. We have that

$$\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{1}{\max(1, \mathcal{N}^t(s_{th}, a_{th}))}} = \sum_{(s,a)}\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}((s_{th}, a_{th}), (s, a))}{\max(1, \mathcal{N}^t(s, a))}}$$

where the outer sum over (s, a) is understood to be going over each similarity state-action pair once instead of going over each individual state-action pair. For any (s, a), we have the following decomposition:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th}, a_{th}), (s, a))}{\max(1, \mathcal{N}^t(s, a))}} &= \sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th}, a_{th}), (s, a))}{\max(1, \mathcal{N}^t(s, a))}}\mathbb{1}\left[\mathcal{N}^t(s, a) \leq \tau_{\max}\right] \\ &+ \sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th}, a_{th}), (s, a))}{\max(1, \mathcal{N}^t(s, a))}}\mathbb{1}\left[\mathcal{N}^t(s, a) > \tau_{\max}\right]\end{aligned}$$

To maximize the first summand, we observe that since each episode lasts a maximum of $\tau_{\max}$ timesteps, the agent would need to loop over the same similarity state-action pair (s, a) for the offline data $\mathcal{D}_t$, thereby maximally accumulating the counting function for two successive episodes and fail the $\mathbb{1}\left[\mathcal{N}^t(s, a) \leq \tau_{\max}\right]$ check starting from the third successive episode. Let t be the first episode where we encounter the similarity state-action pair (s_{th}, a_{th}), assume that for the next 2 episodes, the only data collected are $(\kappa_{\mathcal{G}}(s_{th}), a_{th})$. If (s, a) is such that

$\text{Id}((s_{th}, a_{th}), (s,a)) = 1$, then $\max(1, \mathcal{N}^t(s,a)) = 1$ and $\max(1, \mathcal{N}^{t+1}(s,a)) = \tau_{\max}$ as the counting function $\mathcal{N}^t$ only gets updated at the end of each episode when $\mathbb{D}_t$ is updated. Thus, we have that:

$$\begin{aligned}
&\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\text{Id}(s_{th}, a_{th}), (s,a))}{\max(1, \mathcal{N}^t(s,a))}}\mathbb{1}\left[\mathcal{N}^t(s,a) \le \tau_{\max}\right] \\
&\le \sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\text{Id}\left((s_{th}, a_{th}), (s,a)\right)\mathbb{1}\left[\mathcal{N}^t(s,a) \le \tau_{\max}\right] < 2\tau_{\max}
\end{aligned}$$

as we upper-bound $\frac{1}{\tau_{\max}} < 1$ during the next-episode accumulation.

Consider now the second summand. We begin with an auxillary inequality:

Lemma 5.4.1. For $n > \tau_{\max} > \tau_t$ and $j < \tau_t$, we have that $\frac{1}{n} \le \frac{2}{n+j}$.

Assume now that $\mathcal{N}^t(s,a) > \tau_{\max}$, and (s,a) has been observed $j_t \le \tau_t$ times in $\mathcal{D}_t$. We have

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\text{Id}((s_{th}, a_{th}), (s,a))}{\max(1, \mathcal{N}^t(s,a))}}\mathbb{1}\left[\mathcal{N}^t(s,a) > \tau_{\max}\right] &\le \sum_{t=1}^{T}\sum_{j=1}^{j_t}\sqrt{\frac{1}{\mathcal{N}^t(s,a)}}\mathbb{1}\left[\mathcal{N}^t(s,a) > \tau_{\max}\right] \\
&\le \sum_{t=1}^{T}\sum_{j=1}^{j_t}\sqrt{\frac{2}{\mathcal{N}^t(s,a)+j}}\mathbb{1}\left[\mathcal{N}^t(s,a) > \tau_{\max}\right] \\
&\le \sum_{n=1}^{\mathcal{N}^{T+1}(s,a)}\sqrt{\frac{2}{n}}
\end{aligned}$$

where we applied **Lemma 5.4.1.**, and that $\sum_t^T \sum_j^{j_t} 1 = \mathcal{N}^{T+1}(s,a)$. Thus, we have that for any similarity state-action pair (s,a):

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th},a_{th}),(s,a))}{\max(1,\mathcal{N}^t(s,a))}} &= \sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th},a_{th}),(s,a))}{\max(1,\mathcal{N}^t(s,a))}}\mathbb{1}\left[\mathcal{N}^t(s,a)\le\tau_{\max}\right] \\
&+\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th},a_{th}),(s,a))}{\max(1,\mathcal{N}^t(s,a))}}\mathbb{1}\left[\mathcal{N}^t(s,a)>\tau_{\max}\right] \\
&\le 4\tau_{\max}+\sum_{n=1}^{\mathcal{N}^{T+1}(s,a)}\sqrt{\frac{2}{n}} \\
&\le 4\tau_{\max}+\int_0^{\mathcal{N}^{T+1}(s,a)}\sqrt{2}n^{-1/2}\,dn \\
&= 4\tau_{\max}+2\sqrt{2\mathcal{N}^{T+1}(s,a)}
\end{aligned}$$

Recall that K is the number of similarity classes. Let M_i^* be any subMDP from similarity class i from the true MDP M^*, write $\mathcal{S}_i^*$ for the state space of M_i^*. Define $\bar{S}^* = \sum_{i=1}^K \mathcal{S}_i^*$, the sum of "distinct" internal states where the distinction is with respect to the similarity relation. Thus, summing over all similarity state-action pairs (s,a) once, we have

$$\begin{aligned}
\sum_{(s,a)}\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}(s_{th},a_{th}),(s,a))}{\max(1,\mathcal{N}^t(s,a))}} &\le 4\tau_{\max}\bar{S}\mathcal{A}+2\sqrt{2}\sum_{(s,a)}1\cdot\sqrt{\mathcal{N}^{T+1}(s,a)} \\
&\le 4\tau_{\max}\bar{S}\mathcal{A}+2\sqrt{2}\sqrt{\sum_{(s,a)}1}\sqrt{\sum_{(s,a)}\mathcal{N}^{T+1}(s,a)} \\
& 4\tau_{\max}\bar{S}\mathcal{A}+2\sqrt{2}\sqrt{\bar{S}\mathcal{A}}\sqrt{\sum_{(s,a)}\mathcal{N}^{T+1}(s,a)}
\end{aligned}$$

where we applied the Cauchy-Schwarz inequality. Thus,

$$\begin{aligned}
\mathbb{E}\left[\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{1}{\max(1,\mathcal{N}^t(s_{th},a_{th}))}}\right] &= \mathbb{E}\left[\sum_{(s,a)}\sum_{t=1}^{T}\sum_{h=0}^{\tau_t-1}\sqrt{\frac{\mathrm{Id}((s_{th},a_{th}),(s,a))}{\max(1,\mathcal{N}^t(s,a))}}\right] \\
&\leq 4\tau_{\max}\bar{\mathcal{S}}\mathcal{A} + 2\sqrt{2}\sqrt{\bar{\mathcal{S}}\mathcal{A}}\cdot\mathbb{E}\left[\sqrt{\sum_{(s,a)}\mathcal{N}^{T+1}(s,a)}\right] \\
&\leq 4\tau_{\max}\bar{\mathcal{S}}\mathcal{A} + 2\sqrt{2}\sqrt{\bar{\mathcal{S}}\mathcal{A}}\cdot\sqrt{\mathbb{E}\left[\sum_{(s,a)}\mathcal{N}^{T+1}(s,a)\right]} \\
&\leq 4\tau_{\max}\bar{\mathcal{S}}\mathcal{A} + 2\sqrt{2}\sqrt{\bar{\mathcal{S}}\mathcal{A}}\cdot\sqrt{\sum_{t=1}^{T}\mathbb{E}\left[\tau_t\right]} \\
&= 4\tau_{\max}\bar{\mathcal{S}}\mathcal{A} + 2\sqrt{2}\sqrt{\bar{\mathcal{S}}\mathcal{A}TH}
\end{aligned}$$

5.5 Regret

Putting it all together, we have

$$\begin{aligned}
&\mathrm{BayesRegret}(T) = \sum_{t=1}^{T}\mathbb{E}\left[V^{M^*,\pi^*} - V^{M^*,\pi^t}\right] \\
&= \sum_{t=1}^{T}\mathbb{E}\left[V^{M^*,\pi^*} - V^{M,\pi}\right] + \sum_{t=1}^{T}\mathbb{E}\left[V^{M,\pi} - V^{M,\pi^t}\right] + \sum_{t=1}^{T}\mathbb{E}\left[V^{M,\pi^t} - V^{M^*,\pi^t}\right] \\
&= \mathcal{O}\left(T\Psi(M^*,M)\right) + \mathcal{O}\left(T(\zeta_r + H\zeta_P)\right) + \mathcal{O}\left(H\sqrt{\rho\log\left(\mathcal{A}KH\tau_{\max}T\right)}\left[\tau_{\max}\bar{\mathcal{S}}\mathcal{A} + \sqrt{\bar{\mathcal{S}}\mathcal{A}HT}\right]\right) \\
&= \mathcal{O}\left(T(\Psi(M^*,M) + \zeta_r + H\zeta_P)\right) + \mathcal{O}\left(H^{3/2}\sqrt{\rho\bar{\mathcal{S}}\mathcal{A}T\log(\mathcal{A}KH\tau_{\max}T)}\right)
\end{aligned}$$

Given that $\bar{\mathcal{S}} \leq \rho K$ by construction, we have that

$$\begin{aligned}
\mathrm{BayesRegret}(T) &= \mathcal{O}\left(T(\Psi(M^*,M) + \zeta_r + H\zeta_P)\right) + \mathcal{O}\left(H^{3/2}\rho\sqrt{K\mathcal{A}T\log(\mathcal{A}KH\tau_{\max}T)}\right) \\
&= \tilde{\mathcal{O}}\left(T(\Psi(M^*,M) + \zeta_r + H\zeta_P) + H^{3/2}\rho\sqrt{K\mathcal{A}T}\right)
\end{aligned}$$

where $\tilde{\mathcal{O}}$ hides logarithmic factors. The proof is complete. ∎

Chapter 6

Generalizing noise

We explore several ways in which we can generalize the similarity proposed in **Chapter 4**. This chapter is mostly qualitative and serves more as a conversation starter about the possible ways in which we could generalize noise and compare these generalizations.

6.1 Structural misspecification

In **Chapter 4**, we characterized noise as slight fluctuations in the reward and transition distributions for state-action pairs in the same similarity class with misspecification parameters ζ_r and ζ_P for reward and transition respectively. More specifically, we required that if s and s' are similar states, then $P(\cdot|s,a)$ and $P(\cdot|s',a)$ have the same support modulo the similarity relation. Visually, this means that the graph representation of similar subMDPs are structurally identical while the weights on the edges (reward and transition probabilities) could differ, with the understanding that a directed edge exists between two states x and x' when $P(x'|x,a) > 0$ for some $a \in \mathcal{A}$. A possible way to describe structural misspecification (i.e. similar subMDPs need not share the same graph representation) is to elect a representative subMDP for each similarity class, and solve for the union of all representative subMDPs put together. This can potentially be done via MDP homomorphisms.

6.2 MDP homomorphisms

MDP homomorphisms are dynamic-preserving morphisms in the space of MDPs and its internal structures. The idea was proposed in 2003 in [31] with the purpose of creating a tool to study abstractions of MDPs. Specifically, it was a language to describe strong symmetries and other types of special structures in an environment when the labels of the actions may not match perfectly.

Definition 6.2.1. (Ravindran & Barto, 2003) A MDP homomorphism $h : \mathcal{M} \longrightarrow \mathcal{M}$ from $M = (\mathcal{S}, \mathcal{A}, P, r)$ to $M' = (\mathcal{S}', \mathcal{A}', P', r')$ is a tuple of surjections $\langle f, \{g_s : s \in \mathcal{S}\}\rangle$ with $h(s, a) = (f(s), g_s(a))$ where $f : \mathcal{S} \longrightarrow \mathcal{S}'$ and $g_s : \mathcal{A} \longrightarrow \mathcal{A}'$ such that $r(s, a) = r'(f(s), g_s(a))$ and $P(f^{-1}(f(s'))|s, a) = P'(f(s')|f(s), g_s(a))$.

This definition requires that the rewards and transition probabilities agree (homomorphically), while we would want to incorporate slight fluctuations in reward and transition distributions as well. For this purpose, a more fitting structure is the approximate MDP homomorphism presented in [32], where the similarity is constrained by two parameters ζ_r and ζ_P, analogous to those same parameters that were used in our characterization of noise in **Chapter 4**. Although defined slightly differently, these parameters for approximate MDP homomorphisms control the level of difference between rewards of behaviors taken in similar states and transition probabilities when behaving in similar states respectively.

Denote the true (near-hierarchical) MDP by $M^* = (\mathcal{S}^*, \mathcal{A}^*, P^*, r^*)$. Let it be given that $\left\{M_j^{(k)}\right\}_{j=1}^{J}$ are all subMDPs belonging to the same similarity class $k \in [K]$. Assume now that there exists $\left\{h_j^{(k)}\right\}_{j=1}^{J}$ homomorphisms, each mapping its corresponding index subMDP to a representative subMDP $\tilde{M}^{(k)} = \left\{\mathcal{S}'^{(k)}, \mathcal{A}'^{(k)}, P'^{(k)}, r'^{(k)}\right\}$. Gather all subMDPs into a hierarchical surrogate $\tilde{M} = \text{gather}(\tilde{M}^{(k)}, k \in [K])$, ensuring that subMDP exit states are setup correctly, and let $\tilde{\pi} = \text{plan}(\tilde{M})$. An agent running an algorithm that uses the above approximate MDP homomorphism setup would repeat the following loop until termination for every episode. While not terminated, in state $s \in \mathcal{S}^*$, the agent takes action:

$$a = \tilde{\pi}\left(h_j^{\kappa_{\mathcal{G}}(s)}(s)\right)$$

The agent translates states from the true MDP into a approximately homomorphic state in $\tilde{M}$, and executes $\tilde{\pi}$ there. This method of generalization via approximate MDP homomorphisms capture the structural misspecification described in the previous section as these are handled by the homomorphisms going between the true subMDPs and their representative subMDP.

There are several challenges with this approach. Firstly, one would need to prove the existence of a unifying representative $\tilde{M}^{(k)}$ for each k similarity classes. It is possible that when given a (sub)MDP $M = (\mathcal{S}, \mathcal{A}, P, r)$ as well as a (sub)MDP homomorphism h, one can construct a respresentative MDP $M' = (\mathcal{S}', \mathcal{A}', P', r')$ from M using h. Under certain consistency conditions of M' with respect to a lax bisimulation metric d, the value function for policies which are lifted from M' onto M enjoy strong bounds, that is, one can control the difference between $V^{M,\pi}$ where π is optimal for M and $V^{M',\pi'}$ where π' is optimal for the representative MDP constructed from h. This was the focal point of study of [42]. Electing one representative from a collection of J similar subMDPs via J different homomorphisms would require a loose notion of consistency of homomorphisms across all J subMDPs in order to have the elected M' to be consistent with all of its representees. The regret between the value function using an optimal policy π^* for a near-hierarchical MDP M^* and a value function using a policy π optimal for a hierarchical surrogate M constructed from approximate MDP homomorphisms would be T -linear with respect to an upper bound on the lax bisimulation metrics used for each similarity class, as this loss is invariant throughout each episode.

6.3 Aggregated state representation

Consider now a function $\phi : \mathcal{S} \times \mathcal{A} \longrightarrow \Phi$ where Φ denotes a set of aggregated states, and assume the latter to be finite. We call ϕ a ϵ-error aggregated state representation if for any two state-action pairs $(s, a), (s', a')$ with $\phi(s, a) = \phi(s', a'), |Q(s, a) - Q(s', a')| < \epsilon$. State aggregation is in fact a generalization of hierarchical MDPs. Let M be a hierarchical MDP and $\{M_i\}_{i=1}^{L}$ its decomposition into L subMDPs. Denote the subMDP equivalence relation by $\sim$, define $M_q = \{M_i\} /\!\sim$ to be the quotient of the subMDP collection by $\sim$. Then, one can define a 0-error aggregated state representation ϕ by simply sending state-action pairs in M to their corresponding state-action equivalent in M_q. In this way, state aggregation is a generalization of hierarchical MDPs via the above construction. When ϕ is explicitly given, [9] shows that a model-free optimistic version of Q-learning on the state aggregation enjoys T-linear regret with respect to ϵ.

Since there are no explicit conditions on ϕ, it is possible for ϕ to aggregate from a similarity relation when given a near-hierarchical MDP instead of an equivalence relation in the construction above, resulting in a non-zero ϵ-error aggregated state representation. If one is given the controls over the similarity relation with parameters ζ_r and ζ_P (**Chapter 4**), one could potentially bound ϵ and use the same algorithm for aggregated state representations with a regret only T-linear in ϵ.

Our model-based noisy PSHRL algorithm's runtime is dependent not only on the expected regret bounds but also on the efficiency of planning, where we saw that PEP can be more efficient than ordinary VI when substructures are highly repeateable. An advantage to the state aggregation algorithm mentioned above is that it is model-free; the agent only maintains a copy of a Q-function which it updates after every episode. This entirely eliminates the need to do any planning at all: the bulk of the computation resides in updating the Q-function.

Chapter 7

Discussion and Future Works

We presented theoretical results detailing the extents to which a posterior sampling hierarchical reinforcement learning algorithm is able to perform when run on a near-hierarchical problem.While there is a visible potential for increased efficiency resulting from exploiting subMDP similarities when they exist and are given, the algorithm's regret is linearly dependent on the hierarchical misspecification. When this quantity is small, this analysis shows that one can in fact make the argument that the regret is tolerable if one were to settle for a near-optimal solution. In practice, the choice to opt for a hierarchically-driven solution to gain sample efficiency at the price of optimality is made given many other considerations by the machine learning engineer.

We also saw that the type of misspecification studied is in fact a bit restrictive as it essentially forces the unweighted subMDP graphs to be the same when subMDPs are similar. Specifically, the misspecification allowed is only at the level of the reward and the probabilities in the transition function. To effectively study more complex notions of misspecification, we studied abstraction frameworks that are reliant on having representation functions for similar subMDPs. There are a number of avenues for further research which we outline below.

Firstly, under several loose assumptions on the subMDP-level value functions of options, one can study hierarchical reinforcement learning in environments with countably infinite or uncountable states. As the back-and-forth between subMDPs and subMDP representation happens via representation functions, this effectively allows for weaker subMDP state space assumptions such as linear-MDPs and block MDPs, covering a wide variety of real-life scenarios encountered in domains such as optimal control (e.g. power plant control), robotics (e.g. robotic limb controls, constrained autonomous navigation), and medicine (e.g. protein folding).

When the representation itself is not given, one can learn it on the go. Several works studying low-rank MDPs [2, 26, 49] simultaneously optimize in policy space as well as in representation-function space by generating a sequence of representations that better approximate the true unknown representation based on observed data. These algorithms in general have exploration schemes that homogeneously incorporate exploring unknown areas of the environment as well as exploring in such a way that accurate representations can be learned. With additional knowledge about the hierarchical structure of the environment, we imagine a similar recycling of samples can be studied and learning bounds dependent on K and M can be derived. Even broader, when the representation subMDP is itself not finite, these bounds will depend on the dimensions of the representation space as with algorithms for linear MDPs, or on the eluder dimension of the value functions of the representation subMDPs as with algorithms that follow a general function approximation scheme.

Another potential direction of investigation pertains to exit profile designing. We saw that in the finite case, selecting the sets of exit profiles as an ϵ-cover for the space of exit profiles allows for control over the exit profile suboptimality and thus, the value function found by PEP. In theory, the cardinality of an ϵ-cover is exponential in $\mathcal{E}_i$, so its performance is conditioned on having very small sets of exit states. In the context of a cleaner robot, this makes sense as one imagine there are only several doors that allow the robot to exit a room. Generality calls for targeted exit profile designing where exit profiles effectively capture “the outside” within subMDPs. This is a difficult task especially when done in an online learning setting as the agent is highly uncertain about the environment. A potentially interesting direction would be to translate this MDP-wide uncertainty into local subMDP-wide uncertainties by injecting exit profiles with exploration bonuses. This style of exit-profile designing is directed in a similar fashion as directed exploration via UCB or posterior sampling covered in Chapter 3.

Finally, hierarchical reinforcement learning is one of many ways to which a RL agent may leverage internal structures to a problem in order to achieve a gain in sample and computational efficiency. As hierarchical problems are ever present in real-world reinforcement learning problems, the research for more efficient hierarchical algorithms and the analysis of their theoretical performance not only establishes the ceiling of potentials for algorithms in this field but also lay the foundational framework of mathematical analysis that can be potentially used to analyze algorithms in other types of reinforcement learning problems.

Bibliography

[1] Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep* (2019), 10–4.

[2] Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank mdps. *Advances in neural information processing systems 33* (2020), 20095–20107.

[3] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine 34*, 6 (2017), 26–38.

[4] Auer, P., Jaksch, T., and Ortner, R. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems 21* (2008).

[5] Azar, M. G., Osband, I., and Munos, R. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning* (2017), PMLR, pp. 263–272.

[6] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., and Mordatch, I. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).

[7] Bellman, R. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences 42*, 10 (1956), 767–769.

[8] Desharnais, J., Edalat, A., and Panangaden, P. Bisimulation for labelled markov processes. *Information and Computation 179*, 2 (2002), 163–193.

[9] Dong, S., Van Roy, B., and Zhou, Z. Provably efficient reinforcement learning with aggregated states. *arXiv preprint arXiv:1912.06366* (2019).

[10] Dong, S., Van Roy, B., and Zhou, Z. Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research 23*, 255 (2022), 1–54.

[11] Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995* (2019).

[12] Efroni, Y., Merlis, N., Ghavamzadeh, M., and Mannor, S. Tight regret bounds for model-based reinforcement learning with greedy policies. *Advances in Neural Information Processing Systems 32* (2019).

[13] Frikha, M. S., Gammar, S. M., Lahmadi, A., and Andrey, L. Reinforcement and deep reinforcement learning for wireless internet of things: A survey. *Computer Communications 178* (2021), 98–113.

[14] Galatzer-Levy, I., Ruggles, K., and Chen, Z. Data science in the research domain criteria era: Relevance of machine learning to the study of stress pathology, recovery, and resilience. *Chronic Stress 2* (01 2018), 247054701774755.

[15] Ghavamzadeh, M., Lazaric, A., and Pirotta, M. Exploration in reinforcement learning. Tutorial at AAAI'20, 2020.

[16] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association 58*, 301 (1963), 13–30.

[17] Ishfaq, H., Cui, Q., Nguyen, V., Ayoub, A., Yang, Z., Wang, Z., Precup, D., and Yang, L. Randomized exploration in reinforcement learning with general

value function approximation. In *International Conference on Machine Learning* (2021), PMLR, pp. 4607–4616.

[18] Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is q-learning provably efficient? *Advances in neural information processing systems 31* (2018).

[19] Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory* (2020), PMLR, pp. 2137–2143.

[20] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., and Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems 23*, 6 (2021), 4909–4926.

[21] Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research 32*, 11 (2013), 1238–1274.

[22] Lu, X., Van Roy, B., Dwaracherla, V., Ibrahimi, M., Osband, I., and Wen, Z. Reinforcement learning, bit by bit. *arXiv preprint arXiv:2103.04047* (2021).

[23] Madeka, D., Torkkola, K., Eisenach, C., Foster, D., and Luo, A. Deep inventory management. *arXiv preprint arXiv:2210.03137* (2022).

[24] Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., and Alizadeh, M. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM special interest group on data communication.* 2019, pp. 270–288.

[25] McGovern, A., and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density.

[26] Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning* (2020), PMLR, pp. 6961–6971.

[27] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[28] Osband, I., Russo, D., and Van Roy, B. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems 26* (2013).

[29] Osband, I., and Van Roy, B. Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning* (2017), PMLR, pp. 2701–2710.

[30] Precup, D. *Temporal abstraction in reinforcement learning.* University of Massachusetts Amherst, 2000.

[31] Ravindran, B. Smdp homomorphisms: An algebraic approach to abstraction in semi markov decision processes.

[32] Ravindran, B., and Barto, A. G. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes.

[33] Russo, D. Worst-case regret bounds for exploration via randomized value functions. *Advances in Neural Information Processing Systems 32* (2019).

[34] Sammut, C., and Webb, G. I. *Encyclopedia of machine learning.* Springer Science & Business Media, 2011.

[35] Şimşek, Ö., and Barto, A. Skill characterization based on betweenness. *Advances in neural information processing systems 21* (2008).

[36] Solway, A., Diuk, C., Córdova, N., Yee, D., Barto, A. G., Niv, Y., and Botvinick, M. M. Optimal behavioral hierarchy. *PLoS computational biology 10*, 8 (2014), e1003779.

[37] Stolle, M., and Precup, D. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA*

2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5 (2002), Springer, pp. 212–223.

[38] Strehl, A. L., and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences 74*, 8 (2008), 1309–1331.

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

[40] Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence 112*, 1-2 (1999), 181–211.

[41] Szepesvári, C. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning 4*, 1 (2010), 1–103.

[42] Taylor, J., Precup, D., and Panagaden, P. Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems 21* (2008).

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

[44] Wang, R., Salakhutdinov, R. R., and Yang, L. Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension. *Advances in Neural Information Processing Systems 33* (2020), 6123–6135.

[45] Weissman, T., Ordentlich, E., Seroussi, G., Verdu, S., and Weinberger, M. J. Inequalities for the l1 deviation of the empirical distribution. *Hewlett-Packard Labs, Tech. Rep* (2003).

[46] Wen, Z., Precup, D., Ibrahimi, M., Barreto, A., Van Roy, B., and Singh, S. On efficiency in hierarchical reinforcement learning. *34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada* (2020).

[47] Xu, J., and Zhu, Z. Reinforced continual learning. *Advances in Neural Information Processing Systems 31* (2018).

[48] Yang, Z., Jin, C., Wang, Z., Wang, M., and Jordan, M. Provably efficient reinforcement learning with kernel and neural function approximations. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 13903–13916.

[49] Zhang, X., Song, Y., Uehara, M., Wang, M., Agarwal, A., and Sun, W. Efficient reinforcement learning in block mdps: A model-free representation learning approach. In *International Conference on Machine Learning* (2022), PMLR, pp. 26517–26547.