Université de Montréal

On Impact of Mixing Times in Continual Reinforcement Learning

par

Sharath Chandra Raparthy

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

April 24, 2023

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

On Impact of Mixing Times in Continual Reinforcement Learning

présenté par

Sharath Chandra Raparthy

a été évalué par un jury composé des personnes suivantes :

Pierre-Luc Bacon
(président-rapporteur)

Irina Rish
(directeur de recherche)

Glen Berseth
(membre du jury)

Résumé

L'apprentissage par renforcement (RL) continu est un domaine où l'agent vise à prendre des décisions optimales dans un flux continu d'environnements, de sorte que l'agent n'oublie pas de manière catastrophique les connaissances antérieures et s'adapte rapidement aux nouveaux environnements. Récemment, ce problème d'apprentissage a suscité beaucoup d'intérêt au sein de la communauté, et d'importants progrès ont été réalisés dans le développement d'algorithmes et de repères seuils d'analyse comparative. Cependant, la compréhension des facteurs qui affectent l'apprentissage dans de tels environnements a été très peu étudiée. Dans ce mémoire, nous cherchons à fournir une compréhension théorique de la difficulté d'apprentissage du RL continu à travers le prisme des *temps de mélange*. Nous formalisons d'abord le cadre du RL continu en introduisant les processus de décision markoviens (MDP) dits « extensibles » (scalable). Nous définissons ensuite la notion de temps de mélange et démontrons que les MDP extensibles ont des temps de mélange polynomiaux. Nous soutenons ensuite que les méthodes traditionnelles du RL, comme le ré-échantillonnage avec remise ensembliste (bootstrapping) et l'approximation de Monte Carlo, présentent un biais myope en raison des temps de mélange polynomiaux. Pour vérifier les affirmations théoriques, nous approximons les temps de mélange de politiques (policies) pré-entraînées, à savoir PPO, A2C, DQN, et SAC sur Atari continu et MuJoCo, et nous démontrons que lorsque nous augmentons l'échelle de l'environnement, les temps de mélange augmentent d'un facteur polynomial. Nous terminons la thèse en proposant quelques méthodes de solution qui s'attaquent au biais myopique et nous présentons quelques résultats sur des domaines GridWorlds).

Mots-clés - Apprentissage Continu, Temps de Mélange, Apprentissage par Renforcement.

Abstract

Continual Reinforcement Learning (RL) is an aspirational field where the aim of the agent is to make optimal decisions in a continuous stream of environments so that the agent does not catastrophically forget the previous knowledge and will quickly adapt to the new environments. Recently this learning problem has gained a lot of interest in the community, and there has been quite good progress in developing algorithms and benchmarks. However, there has been very little focus on understanding the factors which affect learning in such settings. In this thesis, we aim to provide a theoretical understanding of the learning difficulty of continual RL through the lens of *Mixing Times*. We first formalize the continual RL setting by introducing the so-called scalable MDPs. Then we define the notion of mixing times and demonstrate that the scalable MDPs have polynomial mixing times. We then go on to argue that the traditional RL methods like bootstrapping and Monte Carlo approximation exhibit myopic bias because of the polynomial mixing times. To verify the theoretical claims, we approximate the mixing times of pretrained policies, namely PPO, A2C, DQN, and SAC on continual Atari and MuJoCo, and demonstrate that as we scale the environment, the mixing times scale by a polynomial factor. We end the thesis by proposing some solution methods that tackle myopic bias and present some results on grid world domains.

Keywords - Continual Learning, Mixing Times, Reinforcement Learning

Contents

List of tables

List of figures

Liste des sigles et des abréviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
NLP	Natural Language Processing
RL	Reinforcement Learning
CRL	Continual Reinforcement Learning
CL	Continual Learning
MDP	Markov Decision Process
DP	Dynamic Programming
MC	Monte Carlo

TD	Temporal Difference
PG	Policy Gradient

Acknowledgements

There are so many people who supported me during my MSc. Here I attempt to thank them all but I apologise if I miss anyone. A big thank you to:

- **My Advisor**: *Irina Rish*: This thesis would not have been possible without the support that you provided throughout my masters. Thank you for giving me academic freedom to pursue ambitious projects without putting even slightest pressure on me.
- **My Closest Collaborators:** *Mathew Riemer*, for the amazing discussions we had throughout my masters. The huge chunk of this thesis would not have been possible without you so I am forever thankful for that. *Maximillian Puelma Touzel*, for teaching me the importance of rigor in research. I learned a lot from you. *Sarthak* - I cannot thank you enough for introducing me to the world of Transformers, which resulted in a spotlight paper at ICLR.
- **Friends**: *Sarthak, Divyat* and *Arnav* - Thanks for stimulating discussions, fun hangouts and literally everything. You made my MSc life a bit easier. Thanks for being there through thick and thin. *Soumye and Shruti* - Thanks for organizing amazing trips and also being supportive. *Moksh, Aniket, Prishruit and Jit* - Thanks for the super competitive foosball games. *Bhairav, Florian, Dishank, Sai and all the REAL lab mates* - Thanks for mentoring, discussions and fun hangouts during my internship. *Sangnie* - Thanks for making my last few weeks at Mila fun and pleasant. *Emmanuel Bengio* - Thanks for being a great manager and mentor. Thanks for teaching me how to do better science. *Mandana, Amin M, Amin Mansouri, Reza, Abhinav, Evgenii, Maksym, Kartik Ahuja, Ivaxi, David, Jean Chritsophe, Mahta* I am grateful for all the great moments and conversations at Mila.
- **Family**: I am extremely grateful to my family. Amma, Nanna and Annayya - Thank you for being there and supporting me through thick and thin.

Chapter 1

Introduction

One of the important characteristics of an intelligent system is the ability to quickly adapt to changes in the environment and continually learn over a wide range of environments. The last decade has been a defining moment in *Artificial Intelligence* (AI) where significant progress has been achieved in applications ranging from *Computer Vision* (CV) [**11, 13, 21**], *Natural Language Processing* (NLP) [**14, 3, 47**] to *Reinforcement Learning* (RL) [**25, 43**]. However, the focus has been predominantly on designing algorithms tailored to specific tasks and not a broad range of tasks. Hence, despite the successes, these algorithms are extremely fragile and fail in unexpected ways when we attempt to make them solve a broad range of tasks. How and why these algorithms fail and how we can design algorithms which overcome these failure modes are interesting research questions and the *Machine Learning* (ML) community has been taking steps in this direction to answer these questions.

In the similar spirit, in this thesis, we examine and study the challenges of developing algorithms that have this innate ability to continually learn in changing and never-ending environments. Specifically, we tackle the following question:

What are the key factors that affect the algorithmic performance while learning in a setting where the environments or the data-distributions change over time?

This learning problem is called *Continual Learning* (CL) where the aim is to learn from a non-stationary stream of data. This problem setting is non-stationary because the environments or the data distribution changes as a function of time and this makes the problem challenging as compared to traditional settings. For example, traditional learning settings like *Supervised Learning* (SL) assumes that the data is distributed according to a fixed distribution and the samples are *Independent and Identically Distributed* (IID). While this makes the learning problem easy, most of the samples in the real world are not IID but are correlated either temporally or in many different ways, and most importantly the data distribution is non-stationary. This makes the CL problem setting close to real world and

at the same time ambitious. The aim of CL is to learn from this non-stationary stream of data without catastrophically forgetting the previous information and also to quickly adapt to the new information.

We focus specifically on RL, a learning paradigm that involves designing computational approaches to learn from interactions. The classical RL [**46**] framework consists of an agent that interacts with the environment by taking actions and receives a reward signal that is a numerical scalar value. The aim of the agent is to maximize the cumulative reward that it receives over time. This framework is formally defined as *Markov Decision Process* (MDP), which we will introduce in the later chapters more rigorously.

Although most of the theoretical insights of the classical RL algorithms assume that MDP is fixed, there is little theoretical analysis on the effectiveness of these RL algorithms where MDP is continuously changing. This setting is called *Continual Reinforcement Learning* (CRL) where the motivations are similar to CL but with some RL flavor. In this thesis, we make the following key contributions:

(1) As our first contribution, which we will discuss in Chapter 3, we provide theoretical insights on the difficulty of the CRL problem through the lens of a fundamental Markov chain property, *Mixing Times*. More precisely, we formalize the CRL framework using the so called *Scalable MDPs* and prove that the mixing times scale by a polynomial factor in this setting. We then theoretically demonstrate the learning difficulty because of the polynomial mixing times.

(2) We back up our theory by performing experiments on the well-known Atari benchmark and demonstrate how astronomically high the mixing times could be as we scale the MDP.

(3) We propose some solution methods that could alleviate the problems in gridworld domains and lay out exciting future directions.

The reset of the thesis is organized as follows. Chapter 2 begins by providing some preliminaries required to understand the main contributions of our work. Since *Markov Chains* are the building blocks of RL, we start by introducing the theory of Markov chains and some of their fundamental properties such as *recurrence*, *periodicity* and *ergodicity* in Section 2.1. We then introduce the notion of *mixing time* of a Markov chain, a concept we discuss repeatedly throughout the thesis in Section 2.1.3. Then we slowly pivot towards RL in Section 2.2 where we introduce the MDP framework, some classic RL algorithms, Deep RL and Continual RL (CRL). Finally, in our first article, we present our paper that discusses how mixing times hinder learning in CRL scenarios and how we can overcome those difficulties.

Chapter 2

Background

2.1. Markov Chain Theory

In this section we will outline some of the important concepts from Markov chain theory. The aim is to provide a basic understanding of Markov chains and some of the fundamental properties which will be later used in the main article.

2.1.1. Markov Chain

A *markov chain* is a stochastic model which describes a sequence of possible elements in a set $\mathcal{S}$ which follow some probabilistic rules. More precisely the sequence evolution follows a property called **Markov Property**, a key property which is heavily used in not only RL theory but also other real world applications.

Definition 1 (Markov Property)**.** *Let $(S_0, S_1 \dots)$ a sequence of discrete random variables from a set $\mathcal{S}$ and P be the transition dynamics. Let $H_{t-1} = \cap_{t=0}^{t-1}\{S_t = s_t\}$ be a set of all random variables in the sequence till $t-1$. Then the Markov property states the follows:*

$$P(S_{t+1} = s_{t+1} \mid H_{t-1} \cap \dots \{S_t = s_t\}) = P(S_{t+1} = s_{t+1} | S_t = s_t)$$

Informally, this property states that the no matter how we arrived at the current state, the probability of transitioning to next state s_{t+1} is solely dependent on the current state s_t. In other words, the next state in the sequence s_{t+1} is conditionally independent of the past given the current state s_t. Any sequence which respects this property is called a *Markov chain*. It is worth mentioning that most of the fundamental algorithms in RL are built based on this property mainly because of its simplicity:

Remark 1. *Given a finite set $\mathcal{S}$, the transition dynamics P of a Markov chain can be represented by a $\mathcal{S} \times \mathcal{S}$ matrix where the entry P_{ij} is the probability that the Markov chain jumps from state i to state j.*

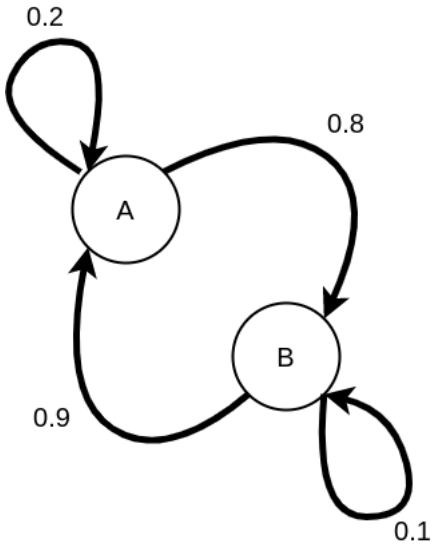


Fig. 2.1. A graphical representation of a two state Markov chain with states A and B. The directed edge from each node carries some edge weight representing the transition probability.

A graphical representation of a two-state Markov chain is shown in Figure 2.1. In this directed graph, the two nodes A and B represent the states and the edge weight correspond to the transition probability from one state to another. The transition probability matrix can be represented as follows:

$$P = \begin{bmatrix} 0.2 & 0.8 \\ 0.9 & 0.1 \end{bmatrix}$$

2.1.2. Properties of Markov Chains

We now present some of the interesting properties of the Markov chains which can be seen widely in the literature. More precisely, we will cover the following properties:

(1) Irreducibility
(2) Periodicity and Aperiodicity
(3) Ergodicity
(4) Stationary Distribution μ

Irreducibility: There are multiple ways in which one can define this property. But intuitively it states that if we start from a state $s \in \mathcal{S}$, then with a non-zero probability we will reach the state $s' \in \mathcal{S}$ and this holds for all states in $\mathcal{S}$. More formally, we define *Irreducibility* as follows:

Definition 2 (Irreducibility)**.** *Let M be a markov chain defined on a discrete set $\mathcal{S}$ with transition probability function $P : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$. We say M is irreducible if for $t \geq 0$ and*

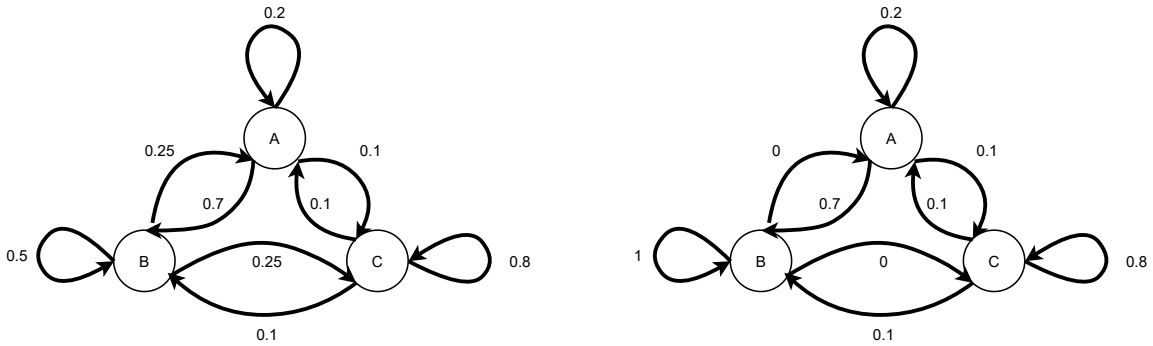


Fig. 2.2. A graphical representation of a 3-state Markov Chain with identical number of nodes but with different edge weights. **Left:** Here the graph is strongly connected with non-zero edge weights making it an irreducible Markov chain. **Right:** Here the graph is weakly connected as it contains some zero edge weights which violates Definition 2

.

$\forall s, s' \in \mathcal{S}$ *the following holds true.*

$$P(S_t = s' \mid S_o = s) > 0$$

Let us also get graphical intuition of this property. If we represent M as a directed graph $\mathcal{G}$, then we say that M is an irreducible markov chain if $\mathcal{G}$ is strongly connected. For example, in Figure 2.2 (right) we have $\mathcal{G}_1$ with three nodes representing a 3-state Markov chain. This is not an irreducible Markov chain because once the chain reaches state "**B**", it is stuck there for infinite time with probability 1. In other words, it is a weakly connected graph as it contains an absorbing state B and which violates Definition 2. On the other hand, in Figure 2.2 (left) we have $\mathcal{G}_2$ which is also a 3-state Markov chain. However, we can see that there is a non-zero probability of transitioning between every two states. This makes $\mathcal{G}_2$ a strongly connected graph resulting in an irreducible Markov chain.

Aperiodicity: Now that we have an understanding about the irreducible Markov chains, we can define **aperiodicity**. In fact, a more general property that any Markov chain possesses is *periodicity* which intuitively corresponds to the greatest common divisor of a set consisting of revisiting times of state s. We say that the markov chain is *aperiodic* when the periodicity is 1.

Definition 3 (Periodicity and Aperiodicity)**.** *Let M be an irreducible Markov chain as defined in Definition 2. Let $\tau(s) := \{t \geq 1 : P(S_t = s \mid S_0 = s) > 0\}$ be a set consisting of the time taken t to revisit the same state s. Then,*

(1) a period of a state s is defined as the greatest common divisor of $\tau(s)$

(2) periodicity of M is defined as period which is common to state states $s \in \mathcal{S}$.

(3) M is aperiodic if the periodicity is 1.

Using the Definition 2 and 3, we can define *ergodicity* as follows:

Definition 4 (Ergodicity)**.** *A markov chain M is ergodic iff M is irreducible and aperiodic.*

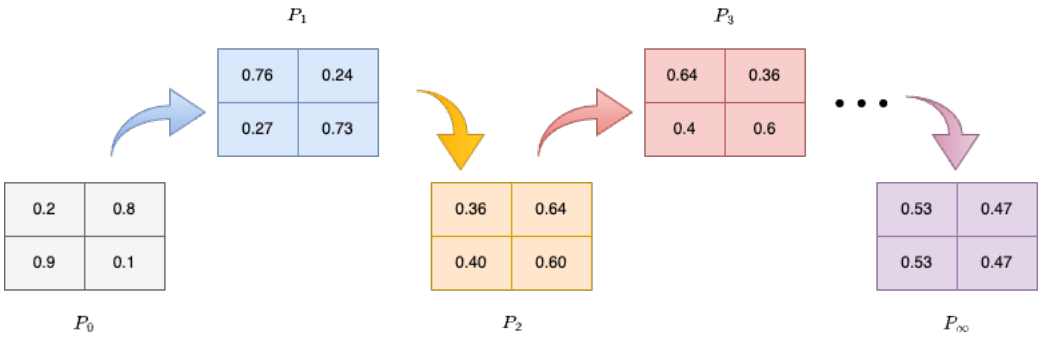


Fig. 2.3. The evolution of the Markov chain depicted in fig. 2.1. We see that the Markov chain converges to a fixed point eventually as $n \to \infty$

If we let the markov chain run for indefinite amount of time, then it converges to a limiting distribution which we called a *stationary distribution*. To understand this, lets consider the markov chain presented in Figure 2.1 with the transition dynamics P and calculate the n-step transitions P_n where $n = 0, 1, 2, \ldots, \infty$ and $P_0 = P$. We define n-step transitions as follows: $P_n = P_{n-1} \times P$. We present the resultant $n-$ step transitions in Figure 2.3. If we observe this figure, we can see that the n-step transition matrix eventually converges to a fixed point and we refer to this fixed point as a *stationary distribution* π. Formally we define the stationary distribution of a markov chain as follows;

Definition 5 (Stationary Distribution)**.** *Given a markov chain M with a transition matrix P, the stationary distribution π induced by M respects the following:*

$$\pi P = \pi$$

.

While in Figure 2.3, we calculate π by iterative method, there exists direct methods which use tools from linear algebra to calculate π. More precisely, the stationary distribution equation in Definition 5 looks similar to system of linear equations of form $Ax = \lambda x$. By rewriting the steady state equation as $(\pi P)^\top = \pi^\top$, the stationary distribution corresponds to the eigen vectors of $P^\top$ with eigenvalue equals to 1. Hence calculating π boils down to finding the eigenvector. It is worth mentioning that the above methods assumes the Markov chain to be finite and discrete. Calculating π for high dimensional and continuous state space Markov chains is still an open research problem and in this thesis we provide algorithms to estimate π using Neural Networks (NNs).

Now we will state one of the famous theorem in Markov chain literature which talks about the uniqueness of the steady state distribution π. Here we will just state the theorem and the proof can be found in [**22**].

Theorem 1. *If a markov chain M is ergodic, then the stationary distribution corresponding to M is unique and independent of any state $s \in \mathcal{S}$*

This uniqueness proof is highly dependent on the ergodicity of the Markov chain. In the literature, these ergodic chains are referred as "*Uni-chain*" Markov chains. There also exists a class of chains referred as "*Multi-chain*" markov chains where the uniqueness condition is dependent on the start states.

2.1.3. Mixing Times

In Figure 2.3, we observed that the markov chain eventually converges to a stationary distribution and from Theorem 1 we can say that this distribution is unique if the markov chain is ergodic (Definition 4). In this section, we are interested in studying some metrics which talk about the convergence rate of a markov chain. More precisely, we are interested in quantifying the convergence speed by defining the so called *Mixing Time*. Mixing time is a metric which quantifies the amount of time it takes for a markov chain to converge to a stationary distribution. While there exists many distance metrics which measure the distance between two distributions of interest, we choose *Total Variation (TV) distance* d_{TV} as a distance metric to quantify mixing times. In our case, these two distributions are *current n-step transition distribution* (P_n) and *stationary distribution* μ.

Given two distributions μ_1 and μ_2 defined on a set $\mathcal{S}$, the total variation distance is the maximum distance between the probabilities assigned to a single event by μ_1 and μ_2. This is defined as follows:

$$d_{TV}(\mu_1, \mu_2) := ||\mu_1 - \mu_2||_{TV} = \max_{A \in \mathcal{S}} |\mu_1(A) - \mu_2(A)|$$

Now, with the help of the above measure, we can define the mixing time. The mixing time is a quantity which signifies the convergence speed of a Markov chain. Given the current n-step transition distribution P_n and the stationary distribution μ, the mixing time is the minimum time required t so that the total variation distance is less than ϵ. Since the metric depends on the value of ϵ we choose, we refer to mixing times t_{mix} as ϵ mixing times $t_{\text{mix}}(\epsilon)$.

Definition 6 (ϵ-Mixing Times)**.** *Let d_{TV} be total variation distance metric defined on $\mathcal{S}$. Let P_t be the t-step transition distribution and μ be the stationary distribution. Then the mixing time $t_{mix}(\epsilon)$ of a markov chain M is defined as*

$$t_{mix}(\epsilon) := \min\{t : d_{TV}(P_t, \mu) < \epsilon\}$$

In this thesis, we refer to the n-step transition distribution as *transient distribution* and talk about the behavior of the algorithm of interest when it is a transient distribution and when it reaches the stationary distribution. Since the mixing times quantify how quickly we move from transient to stationary distribution, we use this metric for our algorithmic analysis throughout this thesis.

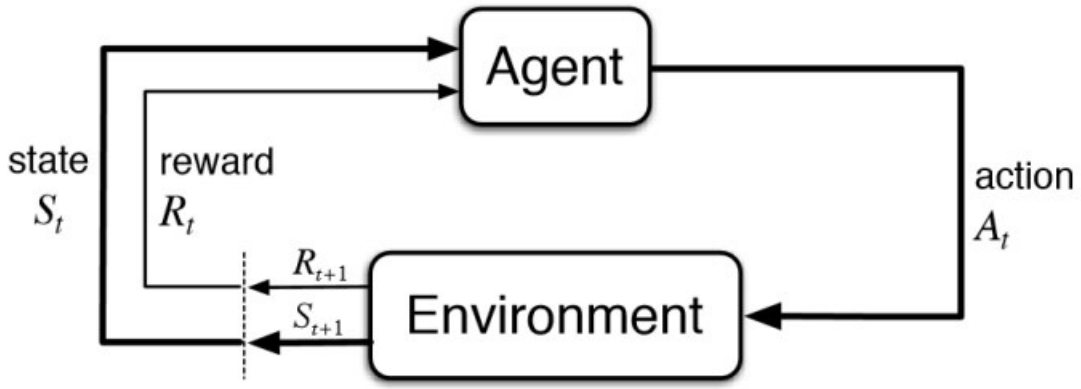


Fig. 2.4. The agent-environment interaction in Reinforcement Learning [**46**]

2.2. Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm in which the decision maker learns to take *optiml decisions* by interacting with the underlying *system*. We refer to *decision maker* as *agent* and the underlying *system* as *environment*. The is agent-environment interaction is succinctly summarized in the Figure 2.4. The aim of the RL agent is to act in a way that maximizes a given performance measure.

The agent interacts with the environment at different time instants where the time can be discrete or continuous. We call this time instant a *decision step* or simply a *timestep*. There is also the notion of *states*, which is the rendering of the current representation of the environment. At each timestep t the agent acts by taking an action a_t from the space of available actions $\mathcal{A}$. Upon acting the agent receives the next state s_{t+1} and reward r_{t+1} from the environment. This agent-environment interaction continues until the agent achieves a specific *goal*. This interplay between agent and the environment results an experience, which we refer to as a *trajectory*. A trajectory τ is the sequence of tuples: $\{(S_0, A_0, R_0), (S_1, A_1, R_1) \dots\}$. The aim of RL is to build algorithms in such a way that the cumulative sum of the rewards is maximized. Most of the fundamental algorithms in RL are built based on a formal mathematical framework called *Markov Decision Process* [**5**] which describes the agent-environment interaction in terms of states, actions, and rewards.

2.2.1. Markov Decision Processes

A *Markov Decision Process* consists of a set of *states* $\mathcal{S}$, a set of *actions* $\mathcal{A}$, a transition probability function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Concisely, we can represent MDP a d 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ The reason why this is referred to as a "markov" decision process is that the transition dynamics function obeys the *markov property* which we defined in Definition 1. In most of the RL literature, the discount factor γ is embedded

in the definition of the MDP but in this thesis we are consciously omitting that because we focus on problems where the discount factor is 1.

If the states and actions are finite discrete sets, then MDP is called a finite MDP and some of the fundamental RL theory assumes that the given MDP is finite. In this case, the transition function is a well defined discrete probability function $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. Although the transition function is concerned only with the next states, we can also define the so-called *transition dynamics* which closely resembles *environment*. More precisely the transition dynamics $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{R} \to [0, 1]$ is a joint probability function which returns the next state and also the reward.

2.2.2. Objective Functions for Reinforcement Learning

In the previous section, we stated that the goal of an agent RL is to make some intelligent decisions in such a way that the cumulative sum of rewards is maximized. Let us now formally discuss different variants of this. In the simplest case, we can define the cumulative sum as follows.

Definition 7 (Return)**.** *Given a trajectory* $\tau = \{(S_{t+1}, A_{t+1}, R_{t+1}), (S_{t+2}, A_{t+2}, R_{t+2}) \dots, (S_T, A_T, R_T)\}$*, we can define the return as follows:*

$$G_t = R_{t+1} + \cdots + R_T \tag{2.2.1}$$

The *return* G_t in Definition 7 could be unbounded when $T \to \infty$ which is undesirable. As a workaround for this, we can introduce a discount factor γ that controls the myopic nature agent and also keeps the sum bounded. We can now define the *discounted return* as follows.

Definition 8 (Discounted Return)**.** *Given a trajectory* $\tau = \{(S_{t+1}, A_{t+1}, R_{t+1}), (S_{t+2}, A_{t+2}, R_{t+2}) \dots\}$ *and the discount factor* γ*, we can define the* γ*-discounted return as follows:*

$$G_t^\gamma := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots = \sum_{k=0}^{\infty} R_{t+k+1} \tag{2.2.2}$$

Instead of introducing the discount factor to keep the sum bounded, we can also take the average of the reward over time. This yields the so-called average reward, which is defined as follows.

Definition 9 (Average Reward)**.** *Given a trajectory* $\tau = S_0, A_0, R_0, S_1, A_1, R_1$*, the average reward is defined as*

$$G_t^{avg} = \frac{1}{T - t - 1} \sum_{k=0}^{T-t-1} R_{t+k+1} \tag{2.2.3}$$

Interpretation of the discount factor There are different interpretations of the discount factor in the RL literature. One widely known reason is that the discount factor controls the relative importance of the future rewards. If $\gamma = 0$, the agent is shortsighted

as it only considers only the immediate reward. As $\gamma \to 1$, the myopicity decreases and when $\gamma = 1$, the agent gives equal importance to all the rewards. On the other hand, it is mathematically convenient to have the discount factor in the return definition, which makes it bounded. Despite these advantages, we argue that the discount factor is ill-suited for problems which involve long-term reasoning amidst the non-stationary environments in the main article.

2.2.3. Policy, Value Function and Bellman Equations

We know that the aim of an RL agent is to *act* intelligently in a given state s in such a way that it maximizes the cumulative return and *policy* governs what action to take in that state s. Formally, a policy $\pi : \mathcal{S} \to \mathcal{A}$ is a function which maps the states to probabilities of selecting each action. Moreover, we also need a have a notion of how good our policy is and this is given by *value function* which we formally define as follows.

Definition 10. *The value v_π of a state s and a policy π is the expected return obtained from that state if we follow the policy π from thereon.*

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\ &:= \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1} \mid S_t = s \right] \end{aligned}$$

We can also define a value function as a function of state s and the action a for a given policy π as follows. We refer to this as the *action-value function.*

Definition 11. *The action-value function $q_\pi(s, a)$ of a policy π is defined as the expected return we get if the agent starts in state s, takes an arbitrary action a, and then acts according to the policy π thereon.*

$$\begin{aligned} q^\pi(s, a) &:= \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &:= \mathbb{E}_\pi \left[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \end{aligned}$$

In RL the policy π, the value function $v_\pi(s)$ and the action-value function $q_\pi(s, a)$ are considered the bread and butter of almost all algorithms that optimize to obtain the best possible solution and this "best possible solutions" are defined with the help of *optimality equations*. We formally define the optimality equations as follows.

Definition 12 (Optimality Equations)**.** *We define:*

(1) The ***optimal value function*** *$v^\star_\pi(s)$ as the best possible value function that an agent can get in a given MDP*

$$v^\star_\pi(s) = \max_\pi v_\pi(s)$$

*(2) The **optimal action-value function** $q_\pi^\star(s,a)$ as the best possible action value function that an agent can obtain in a given MDP*

$$q_\pi^\star(s,a) = \max_\pi q_\pi(s,a)$$

*(3) The **optimal policy** $\pi^\star$ as the dominating policy over all the other policies.*

$$\pi^\star \geq \pi \iff v_\pi^\star \geq v_\pi \quad \forall \pi$$

The value function is recursive in nature and almost all the algorithms leverage this fundamental property to estimate the value/action-value functions. We can rewrite the value function as follows.

$$\begin{aligned} v_\pi(s) &:= \mathbb{E}_\pi\left[G_t \mid s_t = s\right] \\ &= \mathbb{E}_\pi\left[R_t + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_t = s]\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right] \end{aligned} \tag{2.2.4}$$

The above recursive equation is called the *Bellman equation* [**5**] for v^π which expresses the relationship between the value at state s as a function of the value at the successor state. Similarly we can also write the actions-value function in a recursive way as follows:

$$q_\pi(s,a) := r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \sum_{a'} \pi(a \mid s) q_\pi(s', a')$$

Similar to Definition 12, we can also define the optimality equations for the bellman equations as follows:

Definition 13 (Bellman Optimality Equations)**.** *We say that the optimal value function obeys the following expectation equations:*

$$\begin{aligned} v_\pi^\star(s) &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma v_\pi^\star(s')\right] \\ q_\pi(s,a) &:= r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \max_{a'} \pi(a \mid s) q_\pi^\star(s', a') \end{aligned} \tag{2.2.5}$$

2.2.4. Dynamic Programming

In this section, we will go through some of the celebrated solution methods for solving a given MDP $\mathcal{M}$. While solving an MDP sounds a bit arbitrary, we are specifically interested in the following problems:

(1) **Policy Evaluation:** Given a policy π, what is the value function v_π?

(2) **Policy Optimization**: Given a policy π and a value function estimation algorithm, how can we learn an optimal policy $\pi^\star$?

Depending on the type of MDP, the solution methods vary and in this section we will specifically discuss solution methods for finite MDPs with the perfect model of the environment given. We call these solution methods Dynamic Programming (DP), which we use to obtain the optimal value functions and policies. The key idea of DP is to use value functions as a guide to search for better policies. The two most famous DP algorithms for finite MDPs are *a) policy iteration* and *b) value iteration*. The intuition behind these algorithms is to turn the bellman equations discussed in Equation (2.2.5) and (2.2.4) into update rules and iterate over multiple steps until convergence. Then we can use these converged value function to perform policy optimization. In policy iteration, we first perform the policy evaluation using the Bellman equations Equation (2.2.4) and then perform the policy optimization by taking the $\arg\max_a \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi^k(s') \right]$. We repeat these two steps until we reach the optimal policy. We summarize this in Algorithm 1.

Algorithm 1 Policy Iteration

Initialize $v_0(s)$ to zero $\forall s \in \mathcal{S}$ and ϵ
Iterate: ▷ Policy Evaluation Phase
$\forall s \in \mathcal{S}$ calculate $v_\pi^{k+1}(s) = \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi^k(s') \right]$
Stopping criterion: $\forall s \in \mathcal{S}, |v_\pi^{k+1}(s) - v_\pi^k(s)| < \epsilon$
Iterate: ▷ Policy Improvement Phase
$\forall s \in \mathcal{S}$ update the policy $\pi(s) = \arg\max_a \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi^k(s') \right]$
Return Optimal π and optimal v

In case of *Value iteration*, we truncate the policy evaluation loop without losing the convergence guarantees by directly turning the Bellman optimality equations described in eq. (2.2.5) as update rules. More specifically by using these optimality equations, we only perform the policy evaluation for $k = 1$ step and then immediately perform the improvement step. We summarize this in Algorithm 2

Algorithm 2 Value Iteration

Initialize $v_0(s)$ to zero $\forall s \in \mathcal{S}$ and ϵ
Iterate:
$\forall s \in \mathcal{S}$ calculate $v_\pi^{k+1}(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi^k(s') \right]$
Stopping criterion: $\forall s \in \mathcal{S}, |v_\pi^{k+1}(s) - v_\pi^k(s)| < \epsilon$
Return optimal v and optimal $\pi(s) = \arg\max_a \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi^k(s') \right] \forall s \in \mathcal{S}$

While DP has strong theoretical convergence guarantees, it has limited practical applicability. One of the main drawbacks of these methods is that they require explicit model of the environment, which is not the case in real-world scenarios. Moreover, the convergence guarantees no more hold for larger and more general MDPs which can be attributed to *curse of dimensionality*. Alternatives such as *asynchronous* DP are proposed which are in-place

methods that do not require the entire state sweeps to perform policy evaluation. Despite these improvements, the real-world applicability is limited.

2.2.5. Monte Carlo Approximation

As discussed in previous section, one of the main drawbacks of DP methods is that they require the true model of the environment, which limits the applicability of the algorithms for real-world applications. In this section we will discuss some of the important methods which do not explicitly require the true environment but only need samples from them.

Monte Carlo (MC) approximation provides an alternative way to estimate the value/action-value function when the explicit model of the environment is not available. MC methods use the trajectory information to learn unbiased estimates based on averaging sample returns. In particular, we are interested in estimating the value $v_\pi s$ of a state s under the policy π and there are two well-known ways to calculate the sample return averages using MC. The first method is called *first visit* MC where the value of a state s is computed as an average of returns $G(s)$ found in trajectories where s occured first. The second method is called *every visit* MC, where the average returns are calculated for all the visits to s. Theoretically, both methods converge to the optimal values as the number of visits to s tends to infinity.

As mentioned above, MC methods often give an unbiased estimate of $v_\pi(s)$ and are shown to implicitly account for partial observability. But the downside of these methods is that they need to wait until the end of episode to calculate the value estimates. This makes the applicability of this method to only episodic settings.

2.2.6. Temporal Difference Learning

Although MC methods provide unbiased value estimates in theory, their validity is often constrained to episodic settings. Here, we discuss an alternative way to estimate value function. We are interested in estimating the value function of a given policy v_π purely from the samples and *Temporal Difference* (TD) methods provide a way to do this. The core idea here is *bootstrapping* - they update the value estimate based in part of other value estimate. When we pay close attention to the bellman equations, we can see that DP methods also perform bootstrapping. However, in TD learning, since we do not have access to the true model, we learn the value functions from environment interactions and update the value function as follows:

$$v_\pi(S_t) \leftarrow v_\pi(S_t) + \alpha \left[R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi(S_t)\right] \tag{2.2.6}$$

Here the TD error of a one-step transition is defined as $\delta_t \coloneqq R_{t+1} + \gamma v_\pi(S_{t+1}) - v_\pi(S_t)$. We can apply the same idea to action-value functions and the resulting algorithm is called

SARSA because it uses the tuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$.

$$q_\pi(S_t, A_t) \leftarrow q_\pi(S_t, A_{t+1}) + \alpha \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_t) - q_\pi(S_t, A_t)\right]$$

Note that in this algorithm we estimate $q_\pi(S_t, A_t)$ for the current policy π which makes **SARSA** an on-policy algorithm. We also have an off-policy counterpart called **Q-learning** [**50**] which is considered as one of the early breakthroughs in RL. In Q-learning, we learn the action-value function independent of the current policy making it an off-policy algorithm. The update rule is similar to of SARSA but has a flavour of bellman optimality equations.

$$q_\pi(S_t, A_t) \leftarrow q_\pi(S_t, A_{t+1}) + \alpha \left[R_{t+1} + \gamma \max_a q_\pi(S_{t+1}, a) - q_\pi(S_t, A_t)\right]$$

One of the advantages of TD methods over DP methods is that the former do not require any explicit model of the environment but still guarantees the convergence in tabular environments. Moreover, since we are updating the values via bootstrapping, we can learn the values from the incomplete episodes, as opposed to MC methods which must wait until the end of episode, making them independent of the temporal span and hence a suitable candidate for online settings. However because of this the estimates are biased of $v_\pi(s_t)$ which could be irreducible in some settings.

2.2.7. Policy Gradient Methods

So far we have discussed methods that involve estimating the value function - given a policy π what is the value function $v_\pi(s) \;\; \forall s \in \mathcal{S}$. This step is called *policy evaluation*, and the class of methods which perform policy evaluation as an integral step are referred to as *value-based* methods. However, in RL, we are often interested in finding policies that give the maximum cumulative return. While value-based methods provide an implicit way to estimate the policies by taking $\arg\max$, thereby making them greedy, the second class of methods that optimize for policy improvement directly without estimating the values exists. These are called *policy-based methods*.

In this section, we will discuss a class of policy-based methods - *Policy Gradients* [**45**]. These are gradient-based approaches where the policy is directly estimated by gradient-based optimization. More specifically, we parameterize the policy with parameters θ and directly perform gradient ascent on the objective function of interest. We discussed objective functions in RL in Section 2.2.2 and in this section we will consider the discounted return objective (eq. (2.2.2)). The derivation of the policy gradients includes the famous logarithm-derivative trick, and the resultant policy gradient takes the following form:

$$\mathbb{E}_\tau \left[\nabla_\theta G_t\right] = \mathbb{E}_{\tau \sim \pi_\theta} \left[G_t \nabla_\theta \ln(\pi_\theta(A_t|S_t))\right] \tag{2.2.7}$$

Typically the gradient updates are based on the data collected by the current policy, making it an on-policy algorithm. Moreover, the expectation is approximated in a Monte Carlo way. Due to this, the naive PGs have high variance and to tackle this issue we can use state-only dependent baseline functions such as "on-policy value function $v_\pi(s_t)$ or advantage function $A_\pi(s_t, a_t) = Q_\pi(s_t, a_t) - V_\pi(s_t)$. Apart from this, PG methods have poor sample complexity, making the convergence very slow.

2.3. Deep Reinforcement Learning

In previous sections, we discussed various RL algorithms designed to find the optimal value function or a policy of a given MDP. However, while these algorithms guarantee convergence in tabular settings, the same guarantees do not hold as the scale and complexity of the problem increase. As the scale of the problem increases, the RL algorithms need to deal with exponential states and action spaces. Hence, classic RL methods fail because of the *curse of dimensionality*. To tackle this issue, a new sub-field - Deep Reinforcement Learning, emerged, which uses function approximation as a critical tool to generalize what the agent learns about one state to other states. Specifically, deep RL uses the parametric deep neural networks and poses the problem of approximating Q, V, and π as an optimization problems. These methods use backpropagation to calculate the gradients and optimization methods like *Stochastic Gradient Descent* for learning.

Deep Q-Networks (DQN) [**26**] is considered one of the seminal works which used deep neural networks for learning Q values and demonstrated human-level performance on Atari benchmark [**4**]. This work employs a few exciting bells and whistles to the algorithm apart from deep neural networks. These include preprocessing input images and stacking them, using target networks in the loss function, and using experience replay. The main idea behind the target network is to use the old version of the network to compute the target q values and use this to guide the current q value estimates close to the target. This is shown to help stabilize the learning dynamics. As mentioned earlier, DQN uses experience replay, where the transitions are stored in a buffer instead of throwing them away. These transitions are later "replayed" and used to update the Q values.

Another notable work in the space of policy-based methods is Proximal Policy Optimization (PPO) [**40**]. As mentioned in Section 2.2.7, one of the downsides of PG methods is that they are prone to high variance. In PPO, the authors hypothesize that this variance is due to the significant distance between the old policy $\pi_{\theta\text{old}}$ and the new policy, which is the current policy. As a workaround for this, they introduce the following surrogate objective;

$$L_{\text{CPI}}(\theta) = \mathbb{E}_\tau \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t \right]$$

where A_t is the advantage function as described in section 2.2.7. Here CPI stands for *conservative policy iteration* because when we make policy updates we want to be conservative and it should not lead to an excessively large update. To make sure the policy updates are "*conservative*", the authors propose a clipping objective which penalizes the large policy updates. The clip objective $J_{\text{CLIP}}(\theta)$ takes the following form:

$$J_{\text{CLIP}}(\theta) = \mathbb{E}_\tau \left[\min \left(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right] \tag{2.3.1}$$

where ϵ is a hyperparameter. PPO is considered as state-of-the-art for many problems in deep RL and it is a go-to algorithm to train agents.

2.4. Continual Learning

Continual Learning (CL) is a sub-field in ML which is gained traction over the past few years. One way to look at CL is that the model has to learn from the endless stream of sensory information. This makes the problem non-stationary because the data distribution changes as a function of time. The objective of CL is to learn from the non-stationary distributions in such a way that the learner can transfer and adapt to the new tasks and retain the knowledge from the previous tasks. More specifically, [**19**] summarizes the following desiderata for continual learners:

(1) *Context-dependent learning*: Given a context (ex: task id), the agent should learn the tasks in a context-dependent manner.

(2) *Context agnostic learning*: The agent should be able to learn even when the task context is not provided, making it a partially observable problem and challenging.

(3) *Incremental Learning*: The agent learns irrespective of whether the number of tasks is fixed or growing incrementally.

(4) *Minimize the Catastrophic Forgetting*: The agent should be able to retain some knowledge from the history.

(5) *Fast Adaptation*: The agent should be able to adapt to the changes experienced over time quickly.

A learning agent with all the qualities mentioned above can be considered a generalist agent. However, given the complexity and the challenges each desideratum mentioned above presents, researchers are tackling each problem in isolation, thereby addressing less ambitious but still challenging sub-problems. One of the key challenges we analyze in this thesis is *catastrophic forgetting*. Catastrophic forgetting is a common phenomenon we observe in models when naively trained on non-stationary data streams. Intuitively, due to the inherent non-stationarity, the model adapts to recent experiences while significantly forgetting about past experiences. Thus the model faces *stability-plasticity* dilemma where we want the model to be plastic enough for the new experiences but at the same time stable enough

to prevent forgetting previous knowledge. Several works have attempted to tackle this issue which include parameter storage methods [**2, 10, 38**], distillation methods [**7, 17**], rehearsal methods [**36, 34**] etc.

Continual Reinforcement Learning (CRL), in contrast to CL, has a sequential decision-making component which essentially means that the agent should be able to make decisions in a non-stationary environment induced by the tasks. This poses a unique problem as opposed to continual supervised issues because of the poor *credit-assignment* [**19**]. Due to memory constraints, the continual learner will not have direct access to all the previous experiences. Hence, it is difficult to assign credit to the events it experiences over its lifetime. In this thesis, we examine this problem a bit more through the lens of *mixing times* we defined in Definition 6 and show how mixing times contribute to catastrophic forgetting. Now we move on to the next chapter, where we present our work on formalizing the continual RL with the help of mixing times, and we demonstrate how hard continual RL is.

First Article.

Continual Learning In The Environments with Polynomial Mixing Times

by

Matthew Riemer[⋆1], Sharath Chandra Raparthy[⋆2], Ignacio Cases[3], Gopeshh Subbaraj[4], Maximilian Puelma Touzel[5], and Irina Rish[6]

([1]) Université de Montréal, IBM Research, Mila

([2]) Université de Montréal, Mila

([3]) Massachusetts Institute of Technology

([4]) Université de Montréal, Mila

([5]) Université de Montréal, Mila

([6]) Université de Montréal, Mila

This article was submitted in the *proceedings of Neural Information Processing Systems* (NeurIPS 2022).

The main contributions of Sharath Chandra Raparthy for this articles are presented.

- This project was started in 2020 by Matthew Riemer and myself. We both equally contributed ($\star$) to the ideation of the project and also came up with the experimental design choices. Matthew contributed to most of the theory in the paper which includes formulating the propositions, theorems and proofs. I contributed to building

the codebase for all the experiments from the scratch which includes writing code for new environments, coding up all the presented algorithms and also visualizing the results. Matthew wrote the introduction and most of the theory sections. I wrote the experiments and results section.

- Ignacio helped with all the figures in the paper and also with some proofs.
- Gopeshh helped with running some mixing time experiments.
- Maximilian immensely helped throughout the paper. He contributed to the scalable MDP formulation and also the corresponding figure. He cleaned up the writing multiple times and helped us shaping the paper for the conference. He also provided valuable supervision in formalizing some theory in the paper.
- Irina provided supervision for this project and was the PI.

RÉSUMÉ. Le temps de mélange de la chaîne de Markov induite par une politique limite ses performances dans les scénarios réels d'apprentissage continu. Pourtant, l'effet des temps de mélange sur l'apprentissage dans l'apprentissage par renforcement (RL) continu reste peu exploré. Dans cet article, nous caractérisons des problèmes qui sont d'un intérêt à long terme pour le développement de l'apprentissage continu, que nous appelons processus de décision markoviens (MDP) « extensibles » (scalable), à travers le prisme des temps de mélange. En particulier, nous établissons théoriquement que les MDP extensibles ont des temps de mélange qui varient de façon polynomiale avec la taille du problème. Nous démontrons ensuite que les temps de mélange polynomiaux présentent des difficultés importantes pour les approches existantes, qui souffrent d'un biais myope et d'estimations à base de ré-échantillonnage avec remise ensembliste (bootstrapping) périmées. Pour valider notre théorie, nous étudions la complexité des temps de mélange en fonction du nombre de tâches et de la durée des tâches pour des politiques très performantes déployées sur plusieurs jeux Atari. Notre analyse démontre à la fois que des temps de mélange polynomiaux apparaissent en pratique et que leur existence peut conduire à un comportement d'apprentissage instable, comme l'oubli catastrophique dans des contextes d'apprentissage continu.
Mots clés : Apprentissage par renforcement continu, temps de mélange

ABSTRACT. The mixing time of the Markov chain induced by a policy limits performance in real-world continual learning scenarios. Yet, the effect of mixing times on learning in continual reinforcement learning (RL) remains underexplored. In this paper, we characterize problems that are of long-term interest to the development of continual RL, which we call scalable MDPs, through the lens of mixing times. In particular, we theoretically establish that scalable MDPs have mixing times that scale polynomially with the size of the problem. We go on to demonstrate that polynomial mixing times present significant difficulties for existing approaches, which suffer from myopic bias and stale bootstrapped estimates. To validate our theory, we study the empirical scaling behavior of mixing times with respect to the number of tasks and task duration for high performing policies deployed across multiple Atari games. Our analysis demonstrates both that polynomial mixing times do emerge in practice and how their existence may lead to unstable learning behavior like catastrophic forgetting in continual learning settings.
Keywords: Continual Reinforcement Learning, Mixing Times

3.1. Introduction

Continual reinforcement learning (RL) [**19**] is an aspirational field of research confronting the difficulties of long-term, real-world applications by studying problems of increasing scale, diversity, and non-stationarity. The practical requirement for researchers to work on problems of reasonable complexity in the short-term presents a meta-challenge: choosing the

right small-scale problems so that the approaches we develop scale up to the use cases of the future. Here, we address this meta-challenge by formalizing RL problems that vary in size and by analyzing the scaling behavior of popular RL algorithms. In particular, we analyze on the often-ignored *mixing time* that expresses the amount of time until the agent-environment dynamics converge to some stationary behaviour.

Towards this end, we specifically make the following contributions in this work:

(1) **Scalable MDPs:** We propose the formalism of *scalable MDPs* in Definition 14 to characterize an abstract class of MDPs where the MDPs within the class are differentiated based on a changing scale parameter. Understanding the influence of this scale parameter on learning can promote better understanding of the meta-challenge of extrapolating our results on small scale problems to large scale problems of the same class.

(2) **Polynomial Mixing Times:** Theorem 2 establishes the key result of this paper that as any scalable MDP is scaled, its mixing time must grow polynomially as a function of the growing state space. This has major implications for regret analysis in these MDPs.

(3) **Myopic Bias During Scaling:** We demonstrate in Corollaries 2, 3, and 4 that Theorem 2 implies traditional approaches to RL cannot efficiently scale to problems of large size without experiencing myopic bias in optimization that slows down learning.

(4) **Empirical Analysis of Mixing for Continual RL:** We back up our theory with empirical analysis of mixing time scaling in Continual RL settings based on the Atari benchmark. Our analysis provides insight into why agents experience significant catastrophic forgetting or other optimization instability when learning in these domains. We hope that our analysis into the fundamental driver of these issues can open the door for more successful and principled approaches to Continual RL domains like these moving forward.

3.2. Formalizing Our Continual RL Setting

Aiming to characterize a broad range of settings, we focus on the formulation of RL in continuing environments. As explained in [**19**], not only are supervised learning and episodic RL special cases of RL in continuing environments, but so are their non-stationary variants (continual supervised learning and continual episodic RL), despite violating the stationarity assumptions of their root settings. Indeed, approaches that do not acknowledge the more general RL in continuing environments setting exhibit myopic bias in their optimization when faced with non-stationarity [**19**].

Unfortunately, the popular discounted reward setting inserts the very same kind of myopic bias in optimization that we would like to avoid [**41**]. As typically implemented, discounting

does not correspond to the maximization of any objective function over a set of policies [**27**] and the policy gradient is not the gradient of any function [**28**]. These fundamental issues do not resolve as the discount factor approaches 1 [**27**] and discounting does not influence the ordering of policies, suggesting it likely has no role to play in the definition of the control problem [**46**]. In contrast, the average reward per step objective, which explicitly includes an average over the stationary distribution, avoids inducing myopic biases and hence is well-suited for continual RL problems [**46**].

3.3. Average Reward RL in Continuing Environments

RL in continuing environments is typically formulated using a finite, discrete-time, infinite horizon Markov Decision Process (MDP) [**32, 45**], which is a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $R : \mathcal{S} \times \mathcal{A} \to [0, R^{\max}]$ is the reward function, and $T : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0,1]$ is the environment transition probability function. At each time step, the learning agent perceives a state $s \in \mathcal{S}$ and takes an action $a \in \mathcal{A}$ drawn from a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ with internal parameters $\theta \in \Theta$. The agent then receives a reward $R(s,a)$ and with probability $T(s'|s,a)$ enters next state s'. Markov chains may be periodic and have multiple recurrent classes, but optimality is difficult to define in such cases [**49**], making the following assumption necessary for analysis:

Assumption 1. *All stationary policies are aperiodic and unichain, meaning they give rise to a Markov chain with a single recurrent class that is recurrent in the Markov chain of every policy.*

Any RL problem may be modified such that Assumption 1 holds by adding an arbitrarily small positive constant ϵ to all transition probabilities in $T(s'|s,a)$ and renormalizing in which case the effect on the objective of each stationary policy is $O(\epsilon)$ [**8**]. An important corollary to Assumption 1 is that the *steady-state distribution* μ^π induced by the policy π is independent of the initial state:

Corollary 1. *All stationary policies π induce a unique steady-state distribution $\mu^\pi(s) = \lim_{t\to\infty} P^\pi(s_t|s_0)$ that is independent of the initial state such that $\sum_{s\in\mathcal{S}} \mu^\pi(s) \sum_{a\in\mathcal{A}} \pi(a|s) T(s'|s,a) = \mu^\pi(s') \;\; \forall s' \in \mathcal{S}$.*

Corollary 1 implies that the long-term rewards of any π will be independent of the current state. As such, the average reward per step objective $\rho(\pi)$ can be defined independently of its starting state [**46**]:

$$\begin{aligned}\rho(\pi) &:= \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}_\pi \Big[R(s_t,a_t) \Big] = \lim_{t\to\infty} \mathbb{E}_\pi \Big[R(s_t,a_t) \Big] \\ &= \sum_{s\in\mathcal{S}} \mu^\pi(s) \sum_{a\in\mathcal{A}} \pi(a|s) R(s,a) \; . \end{aligned} \tag{3.1}$$

Computing the average reward with the last expression is limited by the amount of time the Markov chain induced by the policy $T^\pi(s'|s) = \sum_{a\in\mathcal{A}} \pi(a|s)T(s'|s,a)$ needs to be run for before reaching the steady-state distribution $\mu^\pi(s)$. This amount of time is referred to in the literature as the mixing time of the induced Markov chain. We denote $t^\pi_{\text{mix}}(\epsilon)$ as the ϵ-*mixing time* of the chain induced by π:

$$t^\pi_{\text{mix}}(\epsilon) := \min\left\{h \;\middle|\; \max_{s_0\in\mathcal{S}} d_{\text{TV}}\Big(P^\pi(s_h|s_0), \mu^\pi(s)\Big) \leq \epsilon\right\}$$

where d_{TV} is the total variation distance. The so-called *conventional mixing time* is defined as $t^\pi_{\text{mix}} \equiv t^\pi_{\text{mix}}(1/4)$. The conventional mixing time only gives insight about distributional mismatch with respect to the steady-state distribution, which led [**18**] to introduce the notion of a mismatch with respect to the reward rate. The ϵ-*return mixing time* is a measure of the time it takes to formulate an accurate estimate of the true reward rate. More formally, if we denote the h-step average undiscounted return starting from state s as $\rho(\pi, s, h)$, then we define the ϵ-*return mixing time* as:

$$t^\pi_{\text{ret}}(\epsilon) := \min\left\{h \;\middle|\; |\rho(\pi, s_0, h') - \rho(\pi)| \leq \epsilon, \quad \forall s_0 \in \mathcal{S} \text{ and } \forall h' \geq h\right\} \tag{3.2}$$

We will come back to this definition when we present our experiments. As emphasized in [**22**], however, one should not get bogged down in the use-case specific definitions of mixing time (we prove their equivalent scaling behaviour in fact, *c.f.* Proposition 1). Rather, it is their shared property of being determined by both a policy π and environment $\mathcal{M}$, and not simply a property of the environment itself, that is important. The environment's contribution can be assessed through mixing times obtained from extreme policies such as the optimal policy, $t^{\pi^*}_{\text{mix}}$.

An alternative approach for quantifying mixing is by considering the structure of the transition matrix $T^\pi(s'|s)$ induced by π. It is well known that the mixing properties of a Markov chain are governed by the spectral gap derived from the eigenvalues of the matrix $T^\pi(s'|s)$. Unfortunately, it is difficult to reason about the spectral gap of a class of MDPs directly. Towards this end, one useful interpretation of a Markov chain is as a random walk over a graph $\mathcal{G}(\pi,T)$ with vertex set $\mathcal{S}$ and edge set $\{(s,s')\}$, for all s and s' satisfying $T^\pi(s'|s) + T^\pi(s|s') > 0$. The *diameter* D^π of the Markov chain induced by π is thus the diameter or maximal graph distance of $\mathcal{G}(\pi,T)$. It is defined using the *hitting time* $t^\pi_{\text{hit}}(s_1|s_0)$, the first time step in which s_1 is reached following the Markov chain $T^\pi(s'|s)$ from s_0,

$$D^\pi := \max_{s_0,s_1\in\mathcal{S}} \mathbb{E}_\pi\Big[t^\pi_{\text{hit}}(s_1|s_0)\Big] \;\; ; \;\; D^* := \min_\pi D^\pi \; . \tag{3.3}$$

D^* then denotes the *minimum diameter* achieved by any policy. All MDPs that follow Assumption 1 have finite diameter D^π for all policies [**15**]. [**22**] (eq. 7.4) established the relationship, $2t^\pi_{\text{mix}} \geq D^\pi$, so that the diameter provides a lower bound for the mixing time and thus serves as a very relevant quantity when conducting mixing time analyses.

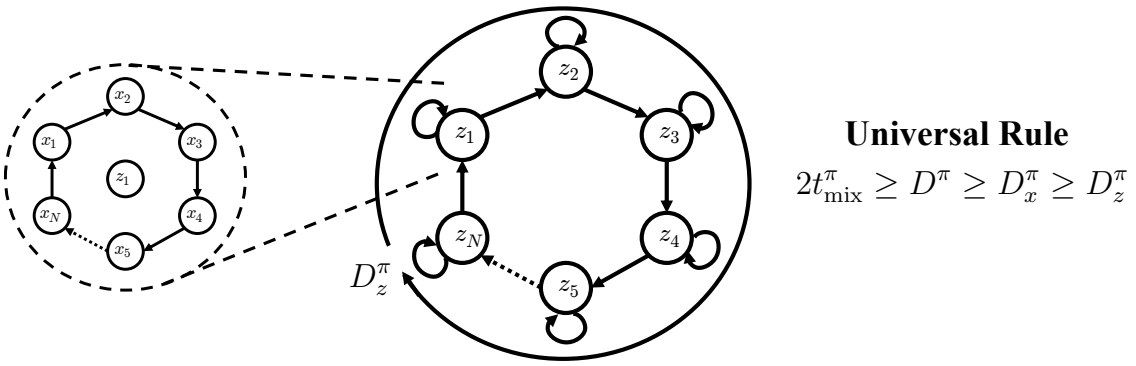


Fig. 3.5. Continual RL Setting: The state space $s \in \mathcal{S}$ is decomposed as $s = [x,z]$ where $z \in \mathcal{Z}$ is the task and $x \in \mathcal{X}_z$ is the within task state. The mixing time, t_{mix}^π, is lower-bounded by the diameter over the full state space D^π. The latter is lower-bounded by the diameter over the within task state space D_x^π, which in turn is lower-bounded by the diameter over the space of tasks, D_z^π.

3.4. The Role Of Tasks And Non-stationarity

The preceding single-task formulation of RL in continuing environments can, in fact, be used to formulate continual RL, which typically makes notions of multiple tasks and non-stationarity explicit. This is indeed a useful construct because arbitrary non-stationarity precludes any consistent signal to learn from, and thus further assumptions about the environment structure must be made to make progress [**19**]. In this paper, we consider tasks as sub-regions of the total MDP (which alternatively can be thought of as independent MDPs connected together) and we assume the transition dynamics between and within tasks are both stationary. As highlighted in [**19**], this setting is quite general as it is capable of modeling many continual RL problems. For example, it easily extends to partially observable variants: if the task component is not directly observed, the problem appears non-stationary from the perspective of an agent that learns from observations of only the within-task state. That said, the continual learning literature demonstrates that optimization issues are generally experienced whether task labels are observed or not [**9**]. Our analysis of mixing times in this paper is an attempt to understand these difficulties even when the total MDP is fully observable and stationary.

To help the reader picture t_{mix}^π in continual RL, we offer the example depicted in Figure 3.5. Here, the agent's state space $\mathcal{S}$ is decomposed into a task component, $z \in \mathcal{Z}$ and a within-task component, $x \in \mathcal{X}_z$. For a policy, π, the pair of Markov chains induced by marginalizing over x and over z have diameters D_z^π and D_x^π, respectively. Using the result connecting mixing times and diameters here, we can establish a universal rule for problems of this type: $2t_{\text{mix}}^\pi \geq D^\pi \geq D_x^\pi \geq D_z^\pi$, where D^π is the diameter over the entire state space. This rule highlights the intimate connection between mixing times and continual RL: task

locality and bottleneck structure inherently lead to environments with correspondingly high mixing times due to high minimum diameters between states in different tasks.

3.5. Scalable MDPs

In this section, we formalize the notion of scaling in the context of MDPs and provide intuition about the effect of this scaling with the aid of the schematic shown in Figure 3.6. We assume MDPs can be described in terms of an n-dimensional parameter vector $q \in \mathbb{R}^n$, and that some subset of these dimensions are the parameters of interest that will be scaled up. We would like to analyze the effect of this scaling on mixing times.[1] With reference to Figure 3.5, these parameters can be non-spatial such as the number of tasks, or spatial such as the size of an individual task. More generally and with reference to Figure 3.6, the effect of scaling up parameters can contribute to longer mixing times by adding bottleneck states (akin to increasing the number of tasks) and increasing the size of the regions generated by a bottleneck (akin to increasing the size of each task). How parameters are scaled is specified by a scaling function, σ controlled by scaling parameter, ν.[2] All MDPs accessible through σ by varying ν form a family of MDPs, $\mathbb{C}_\sigma = \{\mathcal{M}_\nu\}$, where $\mathcal{M}_\nu$ is the MDP specified by q_ν. In particular, we consider *proportional scaling* functions, for which a subset of elements of q are scaled linearly with ν. In our experiments, we show results for proportional scaling of two MDP parameters central to continual RL: the number of tasks and the time between task switches.

As the MDP scales up according to σ, the overall state space size will grow with $|\mathcal{S}| \to \infty$ as $\nu \to \infty$. Regions within the state space can also grow and the steady-state probability on them from π may change as a result. A region $\mathcal{R} \subseteq \mathcal{S}$ is a connected subset of states with steady-state probability $\mu^\pi(\mathcal{R}) = \sum_{s \in \mathcal{R}} \mu^\pi(s)$. The boundary of $\mathcal{R}$ is a subset $\partial\mathcal{R} \subseteq \mathcal{R}$ with states having finite probability of transitioning to at least one state that is outside of $\mathcal{R}$, $\sum_{s' \in \mathcal{S} \setminus \mathcal{R}} T^\pi(s'|s) > 0 \;\; \forall s \in \partial\mathcal{R}$. We denote the scalable regions and boundaries $\mathcal{R}_\nu$ and $\partial\mathcal{R}_\nu$, respectively. If the size of a region's boundary grows faster than the region's interior, there is a finite state space size at which the region no longer has an interior. In that case, the problem type is of bounded complexity and not relevant to the development of scaling friendly algorithms. We are thus interested in problems where scalable regions can maintain an interior as they are scaled (as measured by steady-state probability) in the limit of large ν.

[1] Scalar parameters with discrete domains are incorporated by embedding them into $\mathbb{R}$.

[2] Formally, this is a scaling deformation, $\sigma : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ parameterized by an order parameter $\nu \in \mathbb{R}$ that takes any q_0 to $q_\nu = \sigma(q_0,\nu)$, with $\sigma(\cdot,0)$ as the identity map. Proportional scaling has $q_{\nu,i} \propto \nu$ for all i indexing scaled parameters such that $q_\nu = q_0 + \nu\Delta q$, with $\Delta q \in \mathbb{R}^n$ giving the rates of linear growth in q with ν up from q_0.

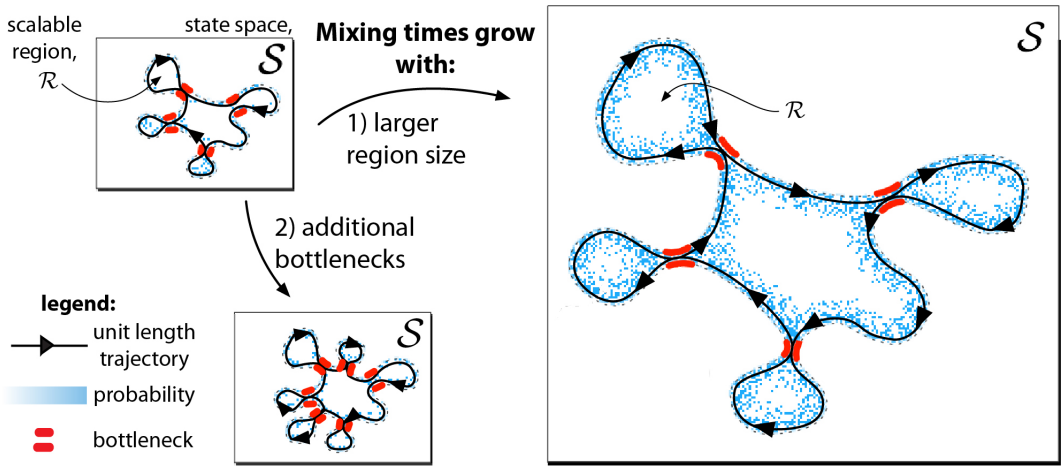


Fig. 3.6. Mixing times grow as MDPs are scaled up. *Top left*: A more general version of the continual RL setting shown in Figure 3.5, where individual tasks correspond to regions, $\mathcal{R}$, of the state space, $\mathcal{S}$, connected through bottlenecks (see legend). An example of the possible steady-state probability of π^* is shown in blue gradient. The mixing time of an MDP can grow by increasing its diameter (number of equidistant arrow heads) via (1) scaling that increases the size of visited regions $\mathcal{R}$ and thus the expected residence time $t_{\mathcal{R}}^{\pi^*}$ of π^* (right) and/or (2) scaling that increases the number of bottlenecks between regions of the state space (bottom left).

Definition 14. *A **scalable MDP** is a family of MDPs $\mathbb{C}_\sigma = \{\mathcal{M}_\nu\}$ arising from a proportional scaling function σ satisfying the property that there exists an initial scalable region $\mathcal{R}_0$ with finite interior, $\mu^{\pi^*}(\partial\mathcal{R}_0) < \mu^{\pi^*}(\mathcal{R}_0)$, that scales so that $\mu^{\pi^*}(\partial\mathcal{R}_\nu) < \mu^{\pi^*}(\mathcal{R}_\nu)$ as $\nu \to \infty$ and thus $|\mathcal{S}| \to \infty$.*

3.6. Polynomial Mixing Times

Prior work analyzing the scaling of mixing times makes the practical distinction between *slow mixing* Markov chains that scale exponentially with a size parameter and *rapid mixing* Markov chains that scale at most polynomially. Since the learning speed of any algorithm is lower bounded by the mixing time of the optimal policy [**18**], environments with a slow mixing Markov chain induced by the optimal policy may very well be outside the practical reach of RL algorithms. In this section we would like to further characterize mixing times over the space of practically solvable problems of sufficient complexity to still be of interest for real-world applications.

For continuing environments, the tightest known lower bound satisfying Assumption 1[3] has H-step regret $\text{Regret}(H) \in \Omega(\sqrt{D^{\pi^*}|\mathcal{S}||\mathcal{A}|H}) \subseteq \Omega(\sqrt{D^*|\mathcal{S}||\mathcal{A}|H})$ [**15**]. Additionally, because we know that $D^* \geq \log_{|\mathcal{A}|}(|\mathcal{S}|) - 3$ [**15**], we can bound our regret in the general case as $\text{Regret}(H) \in \tilde{\Omega}(\sqrt{|\mathcal{S}||\mathcal{A}|H})$. Few practical problems are simple enough to exhibit

[3]The bias span is only lower for weakly communicating MDPs violating Assumption 1 so that $D^* = \infty$.

diameters with such logarithmic contributions. More typical and thus of greater interest are diameters (and mixing times) having polynomial scaling in $|\mathcal{S}|$. Focusing on polynomial scaling is thus warranted, and provides more stringent lower bounds on regret. We thus consider the following definition:

Definition 15. *A set or family of MDPs $\mathbb{C} \subseteq \mathbb{M}$ has a **polynomial mixing time** if the environment mixing dynamics contributes a $\Omega(|\mathcal{S}|^k)$ multiplicative increase for some $k > 0$ to the intrinsic lower bound on regret as $|\mathcal{S}| \to \infty \ \forall \mathcal{M} \in \mathbb{C}$.*

An immediate utility of Definition 15 is that it subsumes the diversity of different diameter and mixing time definitions by explicitly stating their equivalence in scaling with respect to the state space size:

Proposition 1. *If all MDPs $\mathcal{M}$ within the subclass of MDPs $\mathbb{C} \subseteq \mathbb{M}$ have $t^{\pi^*}_{\text{ret}}$, $t^{\pi^*}_{\text{ces}}$, $t^{\pi^*}_{\text{mix}}$, D^{π^*}, or $D^* \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$ we can say that $\mathbb{C}$ has a polynomial mixing time.*[4]

Proof. It holds directly from [**15**] Theorem 5 that following **??** 1 from the main text yields a lower bound on regret for RL algorithms $\text{Regret}(H) \in \Omega(\sqrt{D^*|\mathcal{S}||\mathcal{A}|H})$, which implies that if $D^* \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$, there must be a *polynomial mixing time*. We then begin by noting how this regret bound holds equally for D^{π^*}. We proceed to demonstrate that if $t^{\pi^*}_{\text{mix}}$ is polynomial in $|\mathcal{S}|$, the same must be true for D^{π^*}. We finally establish the relationship between $t^{\pi^*}_{\text{ret}}$ as well $t^{\pi^*}_{\text{ces}}$ and $t^{\pi^*}_{\text{mix}}$. As such, if any of these metrics have a polynomial dependence, the regret bound must as well.

Analysis for D^{π^*}: As discussed extensively in [**29**], the common approach to proving regret bounds in RL and bandit settings it to develop a counter-example for which you can demonstrate that it is impossible for an algorithm to get below a certain regret as a function of the problem parameters. This is the approach taken by [**15**] to establish the $\text{Regret}(H) \in \Omega(\sqrt{D^*|\mathcal{S}||\mathcal{A}|H})$ bound. However, a careful analysis of their proof reveals that for the problem they consider $D^* = D^{\pi^*}$. As such the proof of Theorem 5 in [**15**] can equally be used to establish that $\text{Regret}(H) \in \Omega(\sqrt{D^{\pi^*}|\mathcal{S}||\mathcal{A}|H})$. Note that since by definition $\Omega(\sqrt{D^{\pi^*}|\mathcal{S}||\mathcal{A}|H}) \subseteq \Omega(\sqrt{D^*|\mathcal{S}||\mathcal{A}|H})$, the more general result is for D^*. Moreover, if this were not the case it would invalidate the lower bound presented by [**18**] in terms of $t^{\pi^*}_{\text{ret}}$. As such, it is clear that we cannot simply ignore the size of the problem from the perspective of the optimal policy π^*.

Analysis for $t^{\pi^*}_{\text{mix}}$: It is well known that in bounded degree transitive graphs the mixing time $t_{\text{mix}} \in O(D^3)$ where D is the graph diameter [**22**]. Note, however, that it is widely conjectured to be at most $t_{\text{mix}} \in O(D^2)$ [**22**]. Taking the more general case, this implies then that for any policy π and MDP $\mathcal{M}$, $D^\pi \in \Omega((t^\pi_{\text{mix}})^{1/3})$. Therefore, if $t^{\pi^*}_{\text{mix}} \in \Omega(|\mathcal{S}|^k)$ for some

[4]The ϵ-return mixing time $t^\pi_{\text{ret}}(\epsilon)$ is for assessing mixing from the perspective of accumulated rewards, whereas the ϵ-Cesaro mixing time $t^\pi_{\text{ces}}(\epsilon)$ is for periodic problems that do not converge in the limit from Corollary 1. See Appendix D for additional details.

$k > 0$, then the regret must be in $\text{Regret}(H) \in \Omega(\sqrt{|\mathcal{S}|^{k/3}|\mathcal{S}||\mathcal{A}|H})$. Therefore, Definition 2 is satisfied.

Analysis for $t^{\pi^*}_{\text{ret}}$: It was established in Lemma 1 of [**18**] that $t^{\pi}_{\text{mix}} \in \Omega(t^{\pi}_{\text{ret}})$ for any policy π. So this then implies that $t^{\pi^*}_{\text{mix}} \in \Omega(t^{\pi^*}_{\text{ret}})$. Therefore, we can follow the result from the last paragraph to see that if $t^{\pi^*}_{\text{ret}} \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$, it must also be true that Definition 2 is satisfied.

Analysis for $t^{\pi^*}_{\text{ces}}$: It has been established that $7t_{\text{mix}} \geq t_{\text{ces}}$ for any finite chain [**22**]. Therefor, $t^{\pi^*}_{\text{mix}} \in \Omega(t^{\pi^*}_{\text{ces}})$. So by extension of our results presented above, if $t^{\pi^*}_{\text{ces}} \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$, Definition 15 must hold. ■

We can thus hereon focus on formulating scaling MDPs via their state space size. One way to understand how a region, $\mathcal{R}$, contributes to the mixing time is through its *residence time*, $t^{\pi^*}_{\mathcal{R}}$, *i.e.* the average time that π^* spends in $\mathcal{R}$ during a single visit. Using results from the theory of bottleneck ratios in Markov chains [**22**], we show that if scaling the MDP leads to $t^{\pi^*}_{\mathcal{R}}$ increasing by a polynomial factor in $|\mathcal{S}|$ for any, not-too-big $\mathcal{R}$, then the mixing time also increases at a polynomial rate:

Proposition 2. *Any scalable MDP family $\mathbb{C}_\sigma$ exhibits a polynomial mixing time if there exists a scalable region $\mathcal{R}_\nu$ such that $\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}_\nu}] \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$..*

Proof. We first present the bottleneck ratio of a region. We then demonstrate the relationship connecting the bottleneck ratio to the residence time. We conclude by establishing the connection between the bottleneck ratio and the mixing time of the optimal policy. This then allows us to connect the residence time of the optimal policy to its mixing time, which allows us to establish a polynomial mixing time leveraging the result from Proposition 1.

We begin by defining an edge measure (or ergodic flow) as $\xi^\pi(s'|s) := \mu^\pi(s)T^\pi(s'|s)$ [**22**]. For reasoning about regions of state space $\mathcal{R}$, we can further define an edge measure as $\xi^\pi(\mathcal{S} \setminus \mathcal{R}|\mathcal{R}) = \sum_{s' \in \mathcal{S}\setminus\mathcal{R}} \sum_{s \in \mathcal{R}} \xi^\pi(s'|s)$ and steady-state probability $\mu^\pi(\mathcal{R}) = \sum_{s\in\mathcal{R}} \mu^\pi(s)$. This can then be used to define the bottleneck ratio of the set $\mathcal{R}$ [**22**]:

$$\mho^\pi(\mathcal{R}) := \frac{\xi^\pi(\mathcal{S} \setminus \mathcal{R}|\mathcal{R})}{\mu^\pi(\mathcal{R})} = \frac{1}{\mathbb{E}_{\mu^\pi}[t^\pi_{\mathcal{R}}]} \tag{3.4}$$

With the last equality we note the relationship between the bottleneck ratio of a region $\mathcal{R}$ and the *expected residence time* $\mathbb{E}_{\mu^\pi}[t^\pi_{\mathcal{R}}]$, which is the expected probability of being in $\mathcal{R}$ divided by the expected probability of exiting $\mathcal{R}$. We can finally now define the bottleneck ratio of the entire Markov chain $T^\pi(s'|s)$ (also called the conductance or Cheeger constant) as $\mho^\pi_* := \min_{\mathcal{R}\subset\mathcal{S}:\mu^\pi(\mathcal{R})\leq 1/2} \mho^\pi(\mathcal{R})$ [**22**]. Importantly the bottleneck ratio can be used to both upper bound and lower bound on the mixing time [**22**]. For our purposes, we will primarily utilize the fact that $t^\pi_{\text{mix}}(1/4) \geq 1/4\mho^\pi_*$ as stated in Theorem 7.4 of [**22**]. We can then consider the context of π^*:

$$
\begin{aligned}
t^{\pi^*}_{\text{mix}}(1/4) \geq \frac{1}{4\mho^{\pi^*}_*} &= \max_{\mathcal{R}\subset\mathcal{S}:\mu^{\pi^*}(\mathcal{R})\leq 1/2} \frac{1}{4\mho^{\pi^*}(\mathcal{R})} \\
&= \max_{\mathcal{R}\subset\mathcal{S}:\mu^{\pi^*}(\mathcal{R})\leq 1/2} \frac{\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}}]}{4}
\end{aligned}
\tag{3.5}
$$

This in turn implies the following result for the conventional mixing time $t^{\pi^*}_{\text{mix}}(1/4)$:

$$
t^{\pi^*}_{\text{mix}}(1/4) \in \Omega(\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}}]) \;\; \forall \mathcal{R} \subset \mathcal{S} : \mu^{\pi^*}(\mathcal{R}) \leq 1/2 \tag{3.6}
$$

This result is nearly what we are looking for. However, we would like to express the mixing time at an arbitrary precision ϵ in terms of the residence time rather than only considering $\epsilon = 1/4$. To achieve this we note that the proof of Theorem 7.4 in [**22**] can easily be extended for arbitrary ϵ with $\epsilon = 1/4$ only being used as a convention in the literature. Taking the results presented after equation 7.10 in [**22**] and rearranging them with our notation, we find that:

$$
t^{\pi^*}_{\text{mix}}(\epsilon) \geq \frac{1 - \epsilon - \mu^{\pi^*}(\mathcal{R})}{\mho^{\pi^*}_*} \tag{3.7}
$$

As a result as long as we restrict $\mu^{\pi^*}(\mathcal{R}) \leq 1 - \epsilon - \delta$ for some $0 < \delta < 1 - \epsilon$ we can conclude that $t^{\pi^*}_{\text{mix}}(\epsilon) \geq \delta t^{\pi^*}_{\mathcal{R}}$. This implies that we can state the following asymptotic result as long as the inequality on the density is strict, leading to some effective positive δ:

$$
t^{\pi^*}_{\text{mix}}(\epsilon) \in \Omega(\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}}]) \;\; \forall \mathcal{R} \subset \mathcal{S} : \mu^{\pi^*}(\mathcal{R}) < 1 - \epsilon \tag{3.8}
$$

Therefore, if any scalable region $\mathcal{R}_\nu$ exists where $\mu^{\pi^*}(\mathcal{R}_\nu) < 1-\epsilon$ and $\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}_\nu}] \in \Omega(|\mathcal{S}|^k)$ for some $k > 0$, we also know that $t^{\pi^*}_{\text{mix}}(\epsilon) \in \Omega(|\mathcal{S}|^k)$ and thus a polynomial mixing time is ensured following Proposition 1. Moreover, we can finalize our proof of Proposition 2 by noting that any arbitrarily large scalable region of density greater than the bound i.e. $\mu^{\pi^*}(\mathcal{R}_\nu) \geq 1 - \epsilon$ that scales polynomially must also contain a scalable sub-region $\mathcal{R'}_\nu \subset \mathcal{R}_\nu$ of arbitrarily small density $\mu^{\pi^*}(\mathcal{R'}_\nu) < 1 - \epsilon$ that scales at least at the same rate. As such, there is no need to provide an upper bound for the steady-state region probability density in Proposition 2. ■

The purpose of Proposition 2 and Figure 3.6 are to provide intuition to readers about why scalable MDPs inherently must have polynomial mixing times. Our paper's main result builds off Proposition 2, to make a general statement about the set of all possible scalable MDPs:

Theorem 2. *(Polynomial Mixing for Scalable MDPs): Any scalable MDP $\mathbb{C}_\sigma$ has a polynomial mixing time.*

PROOF. We begin by further analyzing the bottleneck ratio and residence time introduced in Proposition 2 by connecting it to the relative densities of a region and its boundary. We then consider how, under the conditions we consider, a proportionally scaled region must grow faster than its boundary and the region to boundary ratio is lower bounded by the respective increase in their sizes as ν increases. Finally, we establish that the scaling of some regions of any scalable MDP undergoing proportional scaling is polynomially larger than the scaling of its boundary and prove polynomial mixing through the use of Proposition 1.

We begin by further analyzing the ergodic flow of any policy π in region $\mathcal{R}$ and boundary $\partial\mathcal{R}$:

$$\begin{aligned}\xi^{\pi}(\mathcal{S} \setminus \mathcal{R}|\mathcal{R}) &= \sum_{s' \in \mathcal{S}\setminus\mathcal{R}} \sum_{s \in \mathcal{R}} \mu^{\pi}(s) T^{\pi}(s'|s) \\ &= \sum_{s' \in \mathcal{S}\setminus\mathcal{R}} \sum_{s \in \partial\mathcal{R}} \mu^{\pi}(s) T^{\pi}(s'|s) \leq \mu^{\pi}(\partial\mathcal{R})\end{aligned} \tag{3.9}$$

As such, we can use this result to provide a lower bound for the expected residence time:

$$\mathbb{E}_{\mu^{\pi}}[t^{\pi}_{\mathcal{R}}] = \frac{\mu^{\pi}(\mathcal{R})}{\xi^{\pi}(\mathcal{S} \setminus \mathcal{R}|\mathcal{R})} \geq \frac{\mu^{\pi}(\mathcal{R})}{\mu^{\pi}(\partial\mathcal{R})} \tag{3.10}$$

This in turn implies the following relation to the mixing time by subbing in equation 3.8:

$$t^{\pi^*}_{\text{mix}}(\epsilon) \in \Omega(\mathbb{E}_{\mu^{\pi^*}}[t^{\pi^*}_{\mathcal{R}}]) \in \Omega\left(\frac{\mu^{\pi^*}(\mathcal{R})}{\mu^{\pi^*}(\partial\mathcal{R})}\right) \;\; \forall \mathcal{R} \subset \mathcal{S} : \mu^{\pi^*}(\mathcal{R}) < 1 - \epsilon \tag{3.11}$$

From this analysis it is clear that we can characterize polynomial mixing times by understanding how the probability mass on any scalable region and the mass on its boundary scale with the state space. Formally, for some scaling function σ with scaling parameter ν giving a state space $\mathcal{S}_\nu$, we would like to understand how the mass on a scalable region $\mathcal{R}_\nu$ and its boundary, $\partial\mathcal{R}_\nu$, $\mu^{\pi^*}(\mathcal{R}_\nu)$ and $\mu^{\pi^*}(\partial\mathcal{R}_\nu)$, respectively, scale with $|\mathcal{S}_\nu|$. Without loss of generality, we set a reference MDP at $\nu = 0$ and denote some region in it as $\mathcal{R}_0$. We suppress the dependence of π^* and μ^{π^*} on ν in our notation for clarity. We now derive the main result. By the definition of a scalable MDP, the bulk-to-boundary mass ratio of a region can not decrease as ν scales up the state space. Thus,

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \geq \frac{\mu^{\pi^*}(\mathcal{R}_0)}{\mu^{\pi^*}(\partial\mathcal{R}_0)} \,. \tag{3.12}$$

Since mass on a scaled region and its boundary cannot increase with ν, $\mu^{\pi^*}(\mathcal{R}_\nu) \leq \mu^{\pi^*}(\mathcal{R}_0)$ and $\mu^{\pi^*}(\partial\mathcal{R}_\nu) \leq \mu^{\pi^*}(\partial\mathcal{R}_0)$. Thus,

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \geq \frac{\mu^{\pi^*}(\mathcal{R}_0)}{\mu^{\pi^*}(\partial\mathcal{R}_0)} \cdot \frac{1 - \frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\mathcal{R}_0)}}{1 - \frac{\mu^{\pi^*}(\partial\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_0)}}$$
$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \geq \frac{\mu^{\pi^*}(\mathcal{R}_\nu) - \mu^{\pi^*}(\mathcal{R}_0)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu) - \mu^{\pi^*}(\partial\mathcal{R}_0)} \,. \tag{3.13}$$

To demonstrate the scaling of the right hand side, we can approximate the numerator and denominator. For sufficiently large $\mathcal{R}$, the variation of the size of and mass on the region and its boundary vary in an approximately continuous way with ν so we can employ the inverse function theorem to expand the mass on a region in its size, $|\mathcal{R}_\nu|$, and the same with its boundary, $|\partial\mathcal{R}_\nu|$. To first order around $\nu = 0$, we have

$$\mu^{\pi^*}(\mathcal{R}_\nu) \approx \mu^{\pi^*}(\mathcal{R}_0) + \left.\frac{\partial\mu^{\pi^*}}{\partial|\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\mathcal{R}| \,, \tag{3.14}$$

$$\mu^{\pi^*}(\partial\mathcal{R}_\nu) \approx \mu^{\pi^*}(\partial\mathcal{R}_0) + \left.\frac{\partial\mu^{\pi^*}}{\partial|\partial\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\partial\mathcal{R}| \,, \tag{3.15}$$

for deviations, $\Delta|\mathcal{R}| := |\mathcal{R}_\nu| - |\mathcal{R}_0|$ and $\Delta|\partial\mathcal{R}| := |\partial\mathcal{R}_\nu| - |\partial\mathcal{R}_0|$ (note that the notation $\partial/\partial|\partial\mathcal{R}|$ is the partial derivative with respect to the size of the region's boundary, $\partial\mathcal{R}$). Then, arranging the terms:

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu) - \mu^{\pi^*}(\mathcal{R}_0)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu) - \mu^{\pi^*}(\partial\mathcal{R}_0)} \approx \frac{\left.\frac{\partial\mu^{\pi^*}}{\partial|\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\mathcal{R}|}{\left.\frac{\partial\mu^{\pi^*}}{\partial|\partial\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\partial\mathcal{R}|} \,. \tag{3.16}$$

Subbing into equation 3.13 above we get:

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \geq \frac{\left.\frac{\partial\mu^{\pi^*}}{\partial|\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\mathcal{R}|}{\left.\frac{\partial\mu^{\pi^*}}{\partial|\partial\mathcal{R}_\nu|}\right|_{\nu=0} \Delta|\partial\mathcal{R}|} \,. \tag{3.17}$$

Noting that the derivatives are of equal sign (less than or equal to zero) following Definition 1, we can now state the following scaling dependence:

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \in \Omega\left(\frac{\Delta|\mathcal{R}|}{\Delta|\partial\mathcal{R}|}\right) \;\; \forall \mathcal{R}_\nu \subset \mathcal{S}_\nu \,. \tag{3.18}$$

To further analyze this situation, let's consider the case where the scalable MDP is proportionally scaled among $n' \leq n$ of its n continuous parameters and $n'' \leq n'$ of these parameters result in scaling of a particular region $\mathcal{R}_\nu$. To provide a scaling dependence to $\frac{\Delta|\mathcal{R}|}{\Delta|\partial\mathcal{R}|}$, we utilize the relationship between the scaling of $|\mathcal{R}_\nu|$ and the scaling of the $n'' \leq n'$ intrinsic dimensions controlling its size. As in the main text, we consider the important example of proportional scaling such that the scaling of these n'' dimensions inherits the scaling behaviour of an n''-dimensional hyper-sphere with radius ν. As such the

state space scaling follows $|\mathcal{R}_\nu| \in \Theta(\nu^{n''})$. Moreover, the volume-to-surface area ratio follows $\Delta|\mathcal{R}|/\Delta|\partial\mathcal{R}| \in \Theta(\nu)$ in the case of the proportional scaling of Definition 1.[5] This then implies that $\Delta|\mathcal{R}|/\Delta|\partial\mathcal{R}| \in \Theta(|\mathcal{S}_\nu|^{1/n'})$, since $|\mathcal{S}_\nu| \in \Theta(\nu^{n'})$ where the initial size $|\mathcal{S}_0|$ can be omitted from asymptotic notation. Finally, we can substitute this expression into equation 3.18:

$$\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)} \in \Omega\left(\frac{\Delta|\mathcal{R}|}{\Delta|\partial\mathcal{R}|}\right) \in \Omega\left(|\mathcal{S}_\nu|^{1/n'}\right) \in \Omega\left(|\mathcal{S}_\nu|^{1/n}\right) \ \forall \mathcal{R}_\nu \subset \mathcal{S}_\nu \tag{3.19}$$

We can then plug this result into equation 3.20, yielding a lower bound for $t^{\pi^*}_{\text{mix}}(\epsilon)$:

$$t^{\pi^*}_{\text{mix}}(\epsilon) \in \Omega\left(\frac{\mu^{\pi^*}(\mathcal{R}_\nu)}{\mu^{\pi^*}(\partial\mathcal{R}_\nu)}\right) \in \Omega\left(\frac{\Delta|\mathcal{R}|}{\Delta|\partial\mathcal{R}|}\right) \in \Omega\left(|\mathcal{S}_\nu|^{1/n}\right) \ \forall \mathcal{R}_\nu \subset \mathcal{S}_\nu : \mu^{\pi^*}(\mathcal{R}_\nu) < 1-\epsilon \tag{3.20}$$

As we noted in our proof of Proposition 2, any arbitrarily large scalable region of density greater than the bound i.e. $\mu^{\pi^*}(\mathcal{R}_\nu) \geq 1-\epsilon$ that scales polynomially must also contain a scalable sub-region $\mathcal{R'}_\nu \subset \mathcal{R}_\nu$ of arbitrarily small density $\mu^{\pi^*}(\mathcal{R'}_\nu) < 1-\epsilon$ that scales at least at the same rate. As such, there is no need to provide an upper bound for the steady-state region probability density in Theorem 1. If a scalable MDP is defined, it must be that such a region exists. As we know that $n \geq n' \geq n'' \geq 1$, there must exist a scalable region $\mathcal{R}_\nu$ that has a residence time scaling with $\Omega(|\mathcal{S}_\nu|^{1/n})$ that meets our definition of a polynomial with exponent $k > 0$ for any scalable MDP with a finite description in terms of n continuous variables. The existence of a polynomial mixing time is then proven using Proposition 1. ■

3.7. Myopic Bias During Scaling

Monte Carlo sampling and bootstrapping are the two primary policy evaluation frameworks for RL [**46**]. Both implicitly assume that a finite and fixed maximum frequency $f^*(\pi) \in [0,1]$ of policy improvement steps is possible for unbiased updates regardless of the problem size. We now propose three corollaries of Theorem 2 that together argue how polynomial mixing times invalidate this assumption simply because $f^*(\pi) \geq 1/t^\pi_{\text{mix}}$, and t^π_{mix} grows over scalable MDPs. In practice, algorithm designers cannot afford to wait until reaching the mixing time before updating their model when mixing times are high. This results in a myopic bias in the policy improvement steps taken.

Monte Carlo sampling: an approach for retrieving unbiased, although often high variance, estimates of performance of a single policy by direct sampling from the environment or a high quality model of the environment. In continuing environments with polynomial mixing times, this sampling procedure poses a problem when optimizing for $\rho(\pi)$: To get an unbiased estimate of $\rho(\pi)$, we must be able to sample from the steady-state distribution $\mu^\pi(s)$, which is only available after t^π_{mix} steps in the environment. Moreover, as demonstrated

[5] This can be seen as using an n-dimensional generalization of Galileo's famous square-cube law.

by [**52**] in the general case where no upper bound on the mixing time is known apriori, $O(|\mathcal{S}|t^{\pi}_{\text{mix}})$ samples are needed to retrieve a single unbiased sample from $\mu^{\pi}(s)$. As such, it is clear that the length of the policy evaluation phase strongly depends on the mixing time of the current policy π and thus the maximum frequency $f^*(\pi)$ of unbiased policy improvement steps decreases as the mixing time increases:

Corollary 2. *A Monte Carlo sampling algorithm for policy π in scalable MDP $\mathbb{C}_\sigma$ has a maximum frequency of unbiased policy improvement steps $f^*(\pi) \to 0$ as $|\mathcal{S}| \to \infty$.*

For scalable MDPs of significant size, Corollary 2 implies that model-free Monte Carlo methods perform unbiased updates arbitrarily slowly and that model-based Monte Carlo methods will need arbitrary amounts of compute for unbiased updates even when a true environment model is known. As such, to address scaling concerns over large horizons, bootstrapping methods based on the Bellman equation are generally recommended [**46**].

Bootstrapping for Evaluation: a process of estimating the value function of the current policy at the current state $V^{\pi}(s)$ in terms of our prior estimate of the value of our policy at the next state $V^{\pi}(s')$ following the Bellman equation. This process is referred to as *iterative policy evaluation* [**45**] and is known to converge to the true V^{π} in the limit of infinite evaluations of a fixed policy. However, in practice, RL algorithms constantly change their policy as they learn and only can afford partial backups during policy evaluation when applied to large-scale domains. Indeed, a foundational theoretical principle in RL is that as long as the agent constantly explores all states and actions with some probability, bootstrapping with only partial backups will still allow an agent to learn π^* in the limit of many samples [**45**]. Sample efficiency can still be quite poor since partial backups insert bias into each individual policy evaluation step. This bias, referred to as *staleness* [**6**], arises when the value function used for bootstrapping at the next state is reflective of an old policy that is currently out of date. Unfortunately, if we want to avoid staleness bias during learning, the length of policy evaluation for each policy must once again depend strongly on the mixing time:

Corollary 3. *A bootstrapping algorithm with policy π in scalable MDP $\mathbb{C}_\sigma$ has a maximum frequency of unbiased policy improvement steps $f^*(\pi) \to 0$ as $|\mathcal{S}| \to \infty$.*

Bootstrapping for Improvement: the process of using bootstrapping for policy improvement as popularized by dynamic programming algorithms such as policy iteration and value iteration, as well as by RL frameworks such as temporal difference (TD) learning and actor-critic. The theoretical foundation of these approaches is the *policy improvement theorem*, which demonstrates the value of taking a greedy action with respect to the estimated action-value function $Q^{\pi}(s,a)$ at each step. At each state $s \in \mathcal{S}$ we can consider a modified policy $\pi_m(a,s,\pi)$ for each potential action $a \in \mathcal{A}$ that defaults to the policy π at all other states. The greedy policy π' is then defined as $\pi'(s) = \text{argmax}_a\, Q^{\pi}(s,\pi_m(a,s,\pi))$ and $\pi'(s') = \pi(s')\ \forall s' \in \mathcal{S} \setminus s$. The policy improvement theorem then tells us that policy changes

towards π' are worthwhile as $V^{\pi}(s) \leq Q^{\pi}(s,\pi'(s)) \leq V^{\pi'}(s)$ eventually yielding π^* in the limit of many changes [**46**]. However, this improvement is unlikely to be efficient when mixing times are high. To demonstrate this, we decompose value into the transient and limiting components respectively $V^{\pi}(s) = V^{\pi}_{\text{trans}}(s) + V^{\pi}_{\text{lim}}(s)$. While clearly $V^{\pi}(s) \leq Q^{\pi}(s,\pi'(s))$, the functions both follow π into the future, so their limiting distribution is the same implying that $V^{\pi}_{\text{trans}}(s) \leq Q^{\pi}_{\text{trans}}(s,\pi'(s))$ and $V^{\pi}_{\text{lim}}(s) = Q^{\pi}_{\text{lim}}(s,\pi'(s))$.

Corollary 4. *Policy improvement steps with bootstrapping based on the Bellman optimality operator guarantee monotonic improvement for V^{π}_{trans}, but do not for V^{π}_{lim}.*

See Appendix A for details on how these ideas connect to relevant off-policy or offline RL approaches and to the literature on catastrophic forgetting in continual RL.

3.8. Empirical Analysis of Mixing Behavior on Atari

We have established theoretically that scalable MDPs have polynomial mixing times and that polynomial mixing times present significant optimization difficulties for current approaches to RL. However, we still must demonstrate 1) that scalable MDPs are a useful construct for understanding the scaling process within modern continual RL benchmarks and 2) that large mixing times become a significant practical impediment to performing reliable policy evaluation in these domains. Towards this end, we consider empirical scaling of the mixing time with respect to the number of distinct tasks, $|\mathcal{Z}|$, and to the task duration, τ, which controls the bottleneck structure. For this purpose, we evaluate a set of high-quality pretrained policies. In both cases, we report empirical results via the dependence of the ϵ-return mixing time (equation 3.2) on the relative precision with which the reward rate is estimated. Presenting the relationship in this way ensures we isolate the contribution that myopic policy evaluation has on the performance relative to the optimum and that high mixing times are not inflated by spurious sources of distributional mismatch that do not hinder policy evaluation.

Domain of Focus. We perform experiments involving sequential interaction across 7 Atari environments: *Breakout, Pong, SpaceInvaders, BeamRider, Enduro, SeaQuest* and *Qbert*. This is a typical number of tasks explored in the continual RL literature [**39, 20, 42, 37, 35**] and high-performing, pretrained policies for them are publicly available [**33**]. We consider a sequential Atari setup where the individual environments can be seen as subregions of a larger environment that are stitched together (see Figure 3.5; also Appendix A for further details). We leverage task labels to use the pretrained model specific to each task as our behavior policy as appropriate when tasks change.

We also perform experiments involving mujoco environments: *HalfCheetah, Hopper, Walker2d, Ant, Swimmer* which have vector valued state spaces. We consider this domain because we are interested to see if the mixing times has any dependence on the state-space.

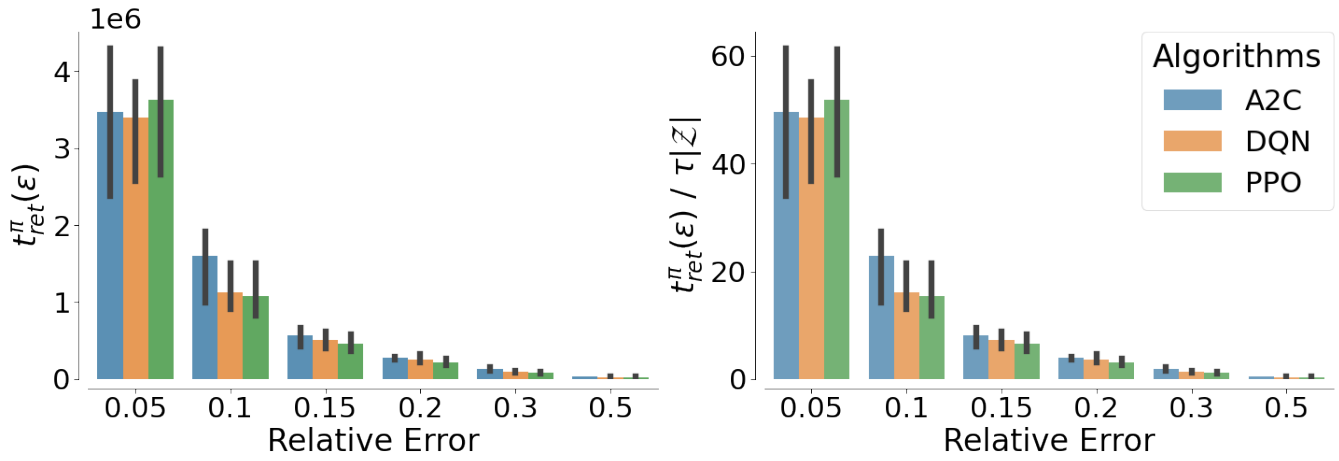


Fig. 3.7. Mixing time as a function of relative error for fixed scaling parameters. *Left*: Return mixing time (equation 3.2) as a function of relative error, $\epsilon/\rho(\pi)$, in reward rate estimation for 3 standard algorithms. *Right*: Same as left, here normalized by $\tau|\mathcal{Z}|$ where τ =10,000 and $|\mathcal{Z}| = 7$. Note the difference in range on the y-axis.

Empirical Mixing Time Estimation. In order to quantitatively estimate the ϵ-return mixing time $t^{\pi}_{\text{ret}}(\epsilon)$ in equation 3.2, we need to estimate two terms: the average true reward rate $\rho(\pi)$ (which is agnostic to the start state), and the h-step undiscounted return $\rho(h, s_0, \pi)$ from a start state s_0 for all $s \in \mathcal{S}$. To calculate $\rho(\pi)$ we unroll the policy π in the environment for a large number of time steps (more than a million steps) while accumulating the rewards that we received and finally compute the average over the total number of environment interactions. Estimating $\rho(h, s, \pi)$ for every start state $s \in \mathcal{S}$ in equation 3.2 is challenging because of the large state spaces in Atari. Hence we rely on approximations where we choose a subset of states from $\mathcal{S}$ to estimate $\rho(h, s, \pi)$. However, since the estimates could be biased based on the start states we choose, instead of randomly sampling a fixed set of start states, we leverage reservoir sampling [**48**] to ensure that the limited fraction of possible start states that we consider is unbiased according to the on-policy distribution. Conservative definitions the mixing time take the maximum over start states [**18**]. Here, since we are interested in the characteristic mixing time size, we report the average among start states weighted according to the on-policy distribution, which ensures that the mixing times reported are representative of those actually needed for evaluation over the course of on-policy training. Finally, in order to provide an intuitive interpretation for ϵ across domains, we parametrize ϵ using the *relative error*, $\epsilon/\rho(\pi)$, and calculate the ϵ-return mixing times for fixed relative error. See Appendix A for further details.

Setting of Interest. We focus our experiments on the following scalable MDP formulation of Atari that is broadly representative of the majority of work on continual RL [**39, 20, 42, 37, 35**]:

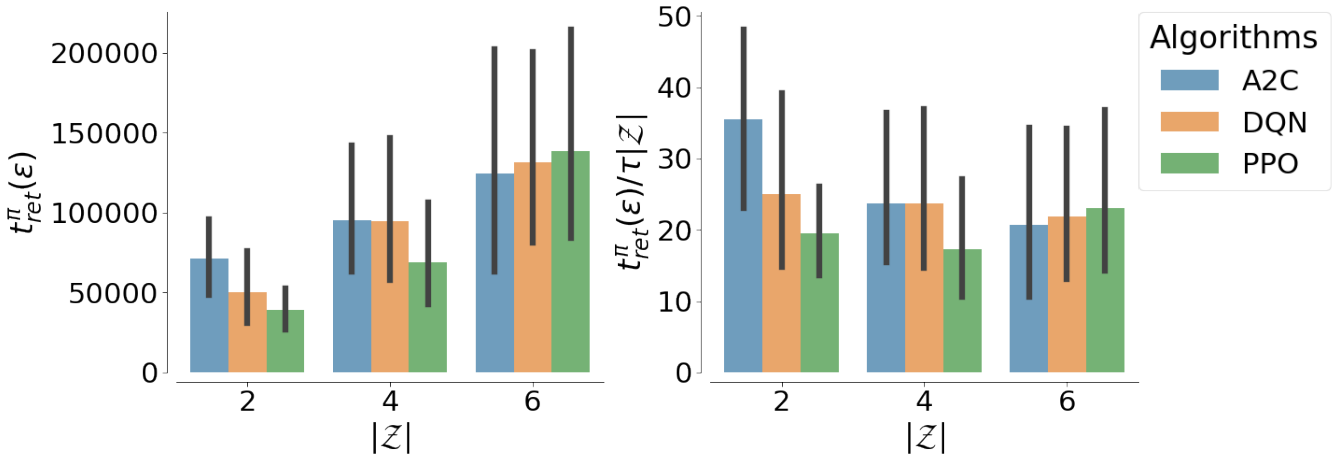


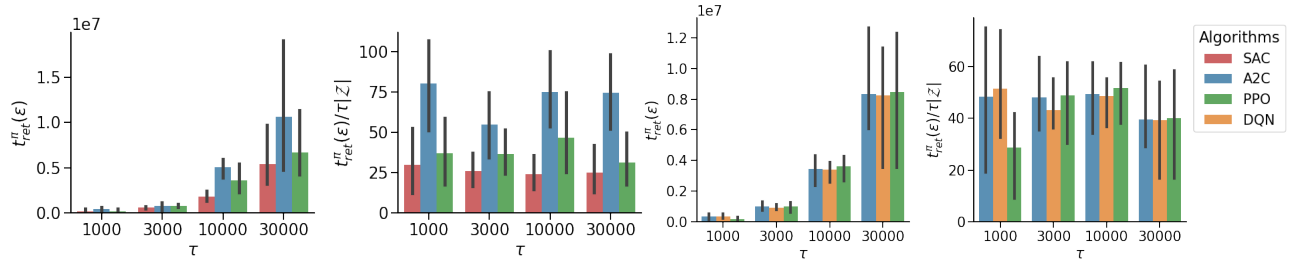
Fig. 3.8. Mixing time scaling with the number of tasks $|\mathcal{Z}|$. *Left*: ϵ-return mixing time across different tasks $\mathcal{Z}$ for different algorithms. *Right*: Same as left, here normalized by $\tau|\mathcal{Z}|$ where τ =1,000. Note the difference in range on the y-axis.

Example 1. *(Continual Learning with Passive Task Switching): Consider an environment with tasks $z \in \mathcal{Z}$ and within task states $x \in \mathcal{X}_z$. The residence time of the region encompassed by task z before moving to another region is fixed for all tasks at τ for any π such that $t^{\pi}_{\mathcal{X}_z} = \tau \ \forall z \in \mathcal{Z}$. Regardless of the way that tasks are connected, the diameter of such an MDP must scale as $D^* \in \Omega(\tau|\mathcal{Z}|)$ because the residence time bounds the minimum possible time to travel between two states in different tasks. Expected mixing times will thus grow at least linearly as either τ or $|\mathcal{Z}|$ are increased. Utilizing our formulation of scalable MDPs with initial MDP parameters $q_0 = (\tau, |\mathcal{Z}|)$, the space of possible proportional scaling functions on τ only, $|\mathcal{Z}|$ only, and both τ and $|\mathcal{Z}|$ simultaneously are $\sigma_{\tau}(q_0, \nu) = (\tau + a\nu, |\mathcal{Z}|)$, $\sigma_{|\mathcal{Z}|}(q_0, \nu) = (\tau, |\mathcal{Z}| + b\nu)$, and $\sigma_{\tau,|\mathcal{Z}|}(q_0, \nu) = (\tau + a\nu, |\mathcal{Z}| + b\nu)$, respectively, where $a, b \in \mathbb{R}$ span the respective spaces.*

3.9. Overview of Empirical Findings

Mixing as a Function of Relative Error. As the task connection structure following Example 1 can take any arbitrary form, we have focused our experiments on random transitions between tasks. In Figure 3.7 we plot both $t^{\pi}_{\text{ret}}(\epsilon)$ and the value normalized by the diameter lower bound $t^{\pi}_{\text{ret}}(\epsilon)/\tau|\mathcal{Z}|$ for τ =10,000 and $|\mathcal{Z}| = 7$ as a function of the relative error of $\rho(\pi, s, h)$ w.r.t $\rho(\pi)$. From this analysis it is clear that the lower bound on the diameter is a very conservative estimate of the mixing time where $t^{\pi}_{\text{ret}}(\epsilon)/\tau|\mathcal{Z}| < 1$ only for reward rates estimated to a poor tolerance of more than 30% error.

Scaling $|\mathcal{Z}|$. We now analyze Example 1 in the case where the state space and diameter grow as we increase the number of tasks $|\mathcal{Z}|$. Specifically we scale $|\mathcal{Z}|$ from 2 to 4 to 6 as τ is kept fixed at 1,000 and generate 10 random task combinations from the list of 7 for each value of $|\mathcal{Z}|$. In Figure 3.8 we plot the resulting mixing times for a precision of 10% error in



(a) Continual Mujoco $|\mathcal{Z}| = 5$ **(b)** Continual Atari $|\mathcal{Z}| = 7$

Fig. 3.9. Mixing time scaling with the task duration, τ. Left (a, b): ϵ-return mixing time across different τ values for different algorithms. Right (a, b): Same as left, here normalized by $\tau|\mathcal{Z}|$. Note the difference in range on the y-axis and the distinct subset of algorithms tested (see legend).

approximating the reward rate. We find that there is a linear trend in the expected mixing time and not just the theoretical lower bound.

Scaling τ. We finally analyze Example 1 in the case where the state space is held constant but the diameter grows with the increasingly severe bottleneck structure as we increase τ. Here we consider $|\mathcal{Z}| = 7$ and $\tau = 1{,}000$, $3{,}000$, $10{,}000$, and $30{,}000$, while calculating $\rho(h, s, \pi)$ and $\rho(\pi)$ by unrolling the policy π for $\tau \times 1{,}000$ environment steps each. In Figure 3.9 we once again plot both $t^{\pi}_{\text{ret}}(\epsilon)$ and the value normalized by the diameter lower bound $t^{\pi}_{\text{ret}}(\epsilon)/\tau|\mathcal{Z}|$ to find that empirical average mixing times scale linearly with increasing τ and not just the lower bound We also perform experiments sequential MuJoco with $|\mathcal{Z}| = 5$ in order to verify the generality of our results across different domains. As expected, we can see in Figure 3.9a that the empirical mixing time scales linearly with τ

In summary, we have observed from Figure 3.7 that, in the case of Atari, average mixing times reach a staggering high for lower relative errors, and shorter mixing times are associated with very high relative errors. From Figures 3.8 and 3.9 we observed that, for a broad class of algorithms, there seems to be a linear dependency of the expected mixing time with increasing $|\mathcal{Z}|$ and increasing τ.

3.10. Implications For Continual RL

Optimization Instability for Continual Gridworld

To demonstrate the implication of polynomial mixing times, we perform experiments on a 3D gridworld domain and analyze the performance as we scale the number of tasks and time spent in each task

Setup: We consider a simple 10x10x10 3-dimensional grid world environment with 6 actions corresponding to up and down actions in each dimension. In this environment, we

can define 1,000 different tasks corresponding to considering each unique state as a goal location.

Inputs: At each time-step the agent is sent a 60-dimensional concatenation of 6 10-dimensional one-hot vectors. The first three one-hot vectors correspond to the agent's x,y, and z coordinates and the second three correspond to the goal location.

Connection to Continual RL: As we have motivated in Section 2.2, this environment is made to demonstrate the difficulty associated with myopic optimization bias even when task boundaries and their relation are fully observable. We consider a continual episodic RL formulation following the description in Example 1 where task sequences arise from changing the goal location every τ steps (the set of goal locations defines the set of tasks, $\mathcal{Z}$). As such, the diameter of the MDP clearly increases as τ and the number of tasks $|\mathcal{Z}|$ are scaled up, leading to at least linear scaling. As a result, an agent that updates only based on the current episode experiences myopic bias with respect to the steady-state distribution (over which the tasks are balanced).

Training Procedure: The 60 dimensional sparse observation vector is processed by a multi layer perceptron with two 100 unit hidden layers and Relu activations. The agent performs optimization following episodic REINFORCE. We explore different configurations of this scalable MDP by varying $|\mathcal{Z}|$ from 1 to 1,000 and varying τ from 100 to 10,000. An important baseline to consider is the extreme of switching tasks after every episode as in multi-task RL (which is generally considered an upper bound on continual RL performance). The agent receives a reward of 1 every step it makes progress towards its goal and a reward of 0 otherwise. This implies that the reward rate of π^* is 1.0 and that the reward rate of a uniform stochastic policy over actions is 0.5 for every task regardless of the length and goal location.

Empirical Evidence: In the Figure 3.10, for each τ and $|\mathcal{Z}|$, we train the agent for 10,000$|\mathcal{Z}|$ total steps, switching between tasks every τ steps. We report the average reward rate at the end of training across the randomly selected task progressions as an average across 60 seeds. It is interesting to notice the episodic transition performance, which can near flawlessly learn a single task in 10,000 steps, but struggles significantly with interference in the multi-task setting. Performance goes down from 1 to 30 tasks and appears to rebound afterward, which is logical because the similarity between incoming tasks and old tasks is increasing and the total number of training steps across tasks is going up. When both τ and $|\mathcal{Z}|$ are high at the same time, learning performance is quite poor (not much better than a uniform policy). The starting location is always in the same corner, so some commonality can be exploited even by a policy experiencing significant intereference. Performance bottoms out with fewer tasks added when τ is larger, which makes sense as the myopic bias of optimization is greatest in that setting.

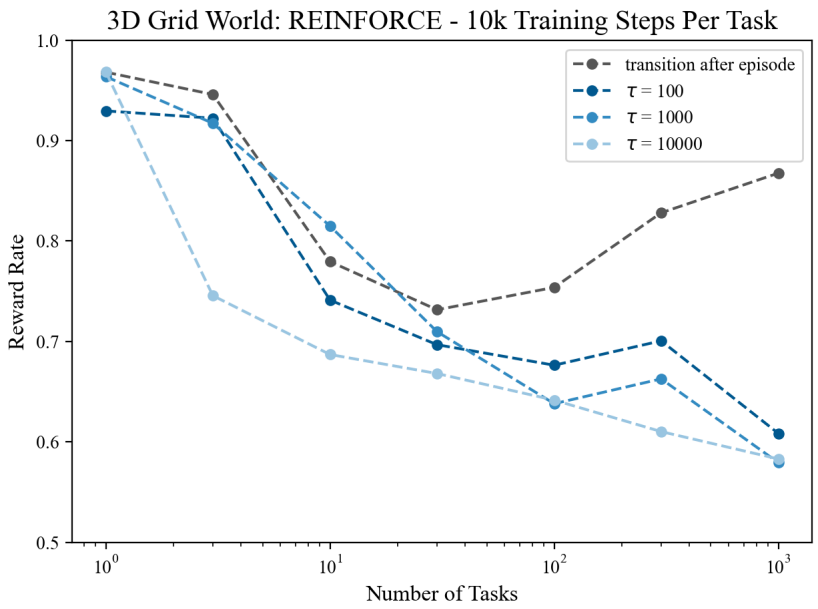


Fig. 3.10. Effect of scaling τ and $\mathcal{Z}$ on the performance: As we scale τ and $\mathcal{Z}$, the reward rate degrades.

Optimization Instability for Continual Atari. While we have empirically established that mixing times scale to be quite large in the Atari domain following Example 1 and established theoretically how RL approaches struggle when this is the case in Section 3.7, we would like to emphasize that the prior literature applying deep RL to continual learning on Atari has empirically come to the same conclusion. Indeed, [**20**] looked at continual learning across 10 Atari games with a randomized duration between task switches and found that all approaches were significantly outperformed by the single task baseline, reflecting significant instability in optimizing across tasks. Additionally, perhaps the most common setting in the literature considered by [**42**], [**37**], and [**1**] is to train on a set of tasks where $|\mathcal{Z}| = 6$ and $\tau = 50{,}000{,}000$. Note that extrapolating from our results would indicate average mixing times on the order of ≈ 15 billion steps for a precision of 5% error or ≈ 5 billion steps for a precision of 10% error. Across a number of RL approaches tried on these domains, as we would expect, all approaches have experienced significant instability in their learning curves. A very popular failure mode is approaches that display too much plasticity such that their performance for a task goes through big peaks and valleys resulting from myopic bias towards optimization on the current task. This phenomena is often known in the literature as catastrophic forgetting [**24**].

3.11. Algorithms for Polynomial Mixing Times

3.12. Policy Improvement With Efficient Scaling

The challenges faced by existing approaches highlighted in Corollaries 2 and 3 can only be adequately addressed by a method for policy evaluation that allows the agent to evaluate new policies with an amount of compute or environment interaction that scales independently of the mixing time. Here, we discuss potential approaches for addressing this policy evaluation problem and approaches for taking policy improvement steps on V^{π}_{lim} directly.

3.13. Policy Evaluation Independent of Mixing Time

A straightforward approach for evaluating a new policy when faced with limited environment interactions between updates is to leverage a model of the environment. Having built a model of the environment transition dynamics $\hat{T}(s'|s,a)$ and reward function $\hat{R}(s,a)$, we can follow equation 3.1 to approximate the average reward objective $\rho(\pi)$ as:

$$\hat{\rho}(\pi) = \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{R}(s,a) \ , \tag{3.21}$$

where the estimated steady-state distribution, $\hat{\mu}^{\pi}$, is obtained from $\hat{T}$ and $\hat{R}$. How to efficiently approximate μ^{π} is unclear, especially in the presence of polynomial mixing times as any sample based algorithm must be run for at least t^{π}_{mix} samples before getting any unbiased samples. In the general setting where the mixing time is not known in advance, the most efficient sample-based algorithm for estimating μ^{π} is based on the Coupling From The Past algorithm [**30, 31**]. This requires $O(|\mathcal{S}|t^{\pi}_{\text{mix}})$ sampling steps before generating a single unbiased sample from $\hat{\mu}^{\pi}$ [**52**]. The algorithm naively requires $O(|\mathcal{S}|^2 t^{\pi}_{\text{mix}})$ computations before generating a sample from $\hat{\mu}^{\pi}$, while an efficient implementation exists that only requires $\tilde{O}(|\mathcal{S}|t^{\pi}_{\text{mix}})$ computations [**52**]. Despite this improvement, any dependence of the computation per step on t^{π}_{mix} is clearly unacceptable in the presence of polynomial mixing times. Additionally, sample-based methods still suffer from high variance despite all of this redundant computation.

An obvious algorithm with computation independent of the mixing time performs k sweeps of approximate value iteration leveraging $\hat{T}$ and $\hat{R}$ at each step, which amounts to $O(k|\mathcal{S}|^2|\mathcal{A}|)$ computations per update. Despite computation independent of the mixing time, value iteration may suffer from slow convergence as it does not address the challenge highlighted in Corollary 4 and thus will not maximize improvements on the limiting distribution at each step.

We present two methods for estimating μ^{π} directly with computation that is independent of t^{π}_{mix}, making them efficient even in scalable MDPs. Both approaches rely on the following

recursive relationship involving $\hat{\mu}$ and $\hat{T}$:

$$\hat{\mu}^{\pi}(s') = \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) \;\; \forall s' \in \mathcal{S} \; . \tag{3.22}$$

Exact steady-state evaluation: The direct approach casts equation 3.22 as a system of equations, which can be solved exactly by performing matrix inversion of a $|\mathcal{S}| \times |\mathcal{S}|$ matrix. Popular libraries implementing Gauss–Jordan elimination execute this procedure in $O(|\mathcal{S}|^3)$, but theoretical algorithms of little practical use can achieve this in as little as $O(|\mathcal{S}|^{2.376})$ [**12**].

Iterative steady-state evaluation: A novel approach leverages the recursive form of Eq. 3.22, which bears many similarities to the Bellman equation. We can provide an estimate of μ^{π} in $O(k|\mathcal{S}|^2|\mathcal{A}|)$ computations as we assume k sweeps over S which is similar to value iteration.

Proposition 3. *Iteratively evaluating $\hat{\mu}^{\pi}$ at iteration k by bootstrapping off the estimate from the last iteration $\hat{\mu}^{\pi}_{k+1}(s') = \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}_{k}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) \;\; \forall s' \in \mathcal{S}$ converges to the true value of $\hat{\mu}^{\pi}$ in the limit as $k \to \infty$.*

Proof. We follow the same high-level proof methodology as used for the standard proof of the convergence of value iteration. First we will show that the steady-state operator $\mathbb{O}_{\infty}$ defined as $\mathbb{O}_{\infty}\hat{\mu}^{\pi}_{k}(s') = \sum_{s \in mathcalS} \hat{\mu}^{\pi}_{k}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) \; \forall s' \in \mathcal{S}$ is a contraction mapping. Then, we will demonstrate that if $\hat{\mu}^{\pi}(s') = \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a)$, there exists a fixed point and it is unique. It can be shown, using the contraction property and the existence and uniqueness of the fixed point, that the sequence of iterates converges to the unique value of $\hat{\mu}^{\pi}$ for any starting s in a closed subset of $\mathcal{S}$.

Contraction Mapping Property: We would now like to further demonstrate the contraction mapping property that:

$$\max_{s' \in \mathcal{S}} \left| \mathbb{O}_{\infty}\hat{\mu}^{\pi}_{1}(s') - \mathbb{O}_{\infty}\hat{\mu}^{\pi}_{2}(s') \right| < \max_{s \in \mathcal{S}} \left| \hat{\mu}^{\pi}_{1}(s) - \hat{\mu}^{\pi}_{2}(s) \right| \tag{3.23}$$

Based on the definition of $\mathbb{O}_{\infty}$ we have that:

$$\begin{aligned}
\max_{s' \in \mathcal{S}} \left| \mathbb{O}_{\infty}\hat{\mu}^{\pi}_{1}(s') - \mathbb{O}_{\infty}\hat{\mu}^{\pi}_{2}(s') \right| &= \max_{s' \in \mathcal{S}} \Big| \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}_{1}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) - \sum_{s \in \mathcal{S}} \hat{\mu}^{\pi}_{2}(s) \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) \Big| \\
&\leq \max_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|s) \hat{T}(s'|s,a) \Big| \hat{\mu}^{\pi}_{1}(s) - \hat{\mu}^{\pi}_{2}(s) \Big| \\
&< \max_{s \in \mathcal{S}} \left| \hat{\mu}^{\pi}_{1}(s) - \hat{\mu}^{\pi}_{2}(s) \right|
\end{aligned} \tag{3.24}$$

The first inequality follows from the property that $|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$. The last inequality being "$\leq$" follows from the fact that $\hat{T}(s'|s,a)$, $\hat{\mu}(s)$, and $\pi(a|s)$ are all non-negative and summing to one. We also can apply normalization

to 1 after each iteration on $\hat{\mu}_k$, which we find in our experiments helps with the speed of convergence. However, this normalization is not needed to ensure convergence as long as $\hat{\mu}(s)$ is non-negative for all $s \in \mathcal{S}$. The fact that we can guarantee a strict "$<$" follows from regularizing $\hat{T}$, via the modification that adds ϵ to all entries, which means every transition has a finite probability of happening for all $\epsilon > 0$.

Existence and Uniqueness of the Fixed Point: Because we regularize $\hat{T}$ by adding ϵ to each connection, this means that regardless of the composition of the actual MDP defined by T we can then guarantee that the approximate MDP defined by $\hat{T}$ is *communicating*. As such, $\hat{T}^{\pi}$ must be unichain and converge to a unique corresponding value of $\hat{\mu}^{\pi}$ following Assumption 1 and Corollary 1. As mentioned in the main text, the effect of the regularization on the objective of each stationary policy is $O(\epsilon)$ [**8**].

■

Similarly to recent work by [**16**] that estimates the accumulated successor state in the discounted reward setting, here we estimate the steady-state distribution in the average reward setting. While the computation is still comparable to approximate value iteration, this approach leaves much potential for further improvements without requiring full sweeps over the state space in analogy to asynchronous dynamic programming methods and approaches for RL. However, a detailed analysis of computationally efficient implementations inspired by approaches extending value and policy iteration would be a standalone contribution beyond the scope of the current work.

3.14. Policy Improvement On The Limiting Distribution

Finally, we now can consider how to leverage this dynamic model-based approximation of the average reward per step $\rho(\pi)$ in order to directly optimize for V^{π}_{lim}, addressing the issue highlighted on Corollary 4. We will begin by defining a new notion of greedy policy improvement by construction:

Theorem 3. *(Average Reward Policy Improvement): Given a stochastic policy π and a greedy policy π' that is then defined as $\pi'(s) = \text{argmax}_{a \in \mathcal{A}} \rho(\pi_m(a,s,\pi))$ with $\pi' = \pi$ at all other states, $\rho(\pi') \geq \rho(\pi) \quad \forall s \in \mathcal{S}$ and eventually converges to an optimal policy π^* with respect to ρ.*

Proof. Demonstrating the efficacy of this form of policy improvement is actually the simplest theoretical result to show in our paper. As in the standard policy improvement theorem, it is true by construction. Furthermore, because of its lack of dependence on a starting state there is no need to examine a sequence of on-policy updates as is typically done to establish the standard policy improvement theorem. We begin by demonstrating policy improvement for deterministic policies and then go on to show it for stochastic policies as well.

Deterministic Policies: Let us consider a deterministic policy $a = \pi(s)$ for all $s \in \mathcal{S}$ and a corresponding modified policy $\pi_m(a,s,\pi)$ which takes action a in state s and follows the original policy $a' = \pi(s')$ for all $s' \in \mathcal{S} - s$. We can then consider a greedy policy $\pi'(s)$ with respect to ρ, defined such that at a state $s \in \mathcal{S}$ the action taken is given by $\pi'(s) = \text{argmax}_{a \in \mathcal{A}} \rho(\pi_m(a,s,\pi))$. The following property then must hold:

$$\rho(\pi'(s)) \geq \rho(\pi(s)) \;\; \forall s \in \mathcal{S} \; . \tag{3.25}$$

The above inequality should be an equality in the case that $\pi(s) = \pi'(s)$. Otherwise if $\pi(s) \neq \pi'(s)$ it should be a strict inequality and monotonic improvements will be guaranteed over π following from the definition of π', assuming the action taken at s makes an impact on ρ and there are no ties.

Stochastic Policies: We now present a slightly modified version of this analysis for the stochastic case. We consider a stochastic policy $a \sim \pi(s)$ following the probability $\pi(a|s)$ for all $s \in \mathcal{S}$ and a corresponding modified policy $\pi_m(a,s,\pi)$ which deterministically takes action a in state s and follows the original stochastic policy $\pi(a'|s')$ for all $s' \in \mathcal{S} - s$. We can begin by defining the average reward of such a stochastic policy:

$$\rho(\pi(a|s)) = \sum_{a \in \mathcal{A}} \pi(a|s) \rho(\pi_m(a,s,\pi)) \;\; \forall s \in \mathcal{S} \tag{3.26}$$

We can then consider a greedy policy with respect to ρ that at a state $s \in \mathcal{S}$ decides to take modify π to take a new action following $\pi'(s) = \text{argmax}_{a \in \mathcal{A}} \rho(\pi_m(a,s,\pi))$. The following property then must hold:

$$\rho(\pi'(s)) \geq \rho(\pi(a|s)) \;\; \forall s \in \mathcal{S} \tag{3.27}$$

The above inequality should once again be an equality in the case that $\pi(s) = \pi'(s)$. On the other hand, if $\pi(s) \neq \pi'(s)$ it should be a strict inequality and monotonic improvements will be guaranteed over π following from the definition of π', assuming the action taken at s makes an impact on ρ and there are no ties. Moreover, it is not necessary that we move all the way to a deterministic greedy action, just that we make that action more likely for this inequality to hold. This can be seen clearly from equation 3.26.

Optimal Policy: As in the standard policy improvement analysis [**45**], when the inequalities converge to an equality i.e $\rho(\pi') = \rho(\pi) \;\; \forall s \in \mathcal{S}$ we have achieved optimality with respect to the ρ objective. Thus we can say we have achieved a *gain optimal policy* following the terminology from [**23**]. ■

Moreover, this new form of policy improvement can also be leveraged to develop a novel policy gradient algorithm [**44, 46**] that motivates how this idea may be applied in deep RL applications.

Theorem 4. *(Average Reward Policy Gradient): Given a stochastic policy π differentiable in its parameters θ with transition and reward functions T and R, the gradient of the expected average reward per step is $\rho(\pi)$:*

$$\begin{aligned}\nabla_\theta \rho(\pi) &= \sum_{s\in\mathcal{S}} \mu^\pi(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s) \rho(\pi_m(a,s,\pi)) \\ &= \mathbb{E}_{\mu^\pi(s,a)}\Big[\nabla_\theta \log \pi(a|s) \rho(\pi_m(a,s,\pi))\Big]\end{aligned} \tag{3.28}$$

where $\mu^\pi(s)$ is the steady-state distribution following T for policy π and $\mu^\pi(s,a) = \mu^\pi(s)\pi(a|s)$.

Proof. We first consider the policy gradient derived by [**44**], but in the undiscounted reward setting. We then modify this gradient for the average reward objective while also leveraging our new bootstrapping procedure that was proposed in Theorem 3. After this, we apply L'Hôpital's rule to achieve our final result.

We begin by building off the policy gradient result initially derived by [**44**]. However, we are not interested in leveraging the differential value function generally used for bias optimal learning in the average reward setting as explained by Proposition 5. As such, we somewhat counterintuitively begin with the discounted policy gradient and consider the limit as the discount factor $\gamma \to 1$. We split the sum over time steps in the value function into a transient, V_{trans}, and a limiting V_{lim} term using the mixing time, t^π_{mix}. In particular, we consider the H-step undiscounted objective $J^H_{\gamma=1}(s,\pi)$ in the limit as $H \to \infty$:

$$\begin{aligned}\lim_{H\to\infty} \nabla_\theta J^H_{\gamma=1}(s,\pi) &:= \lim_{H\to\infty} \sum_{t=0}^{H} \sum_{s_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} P(s_t|s,\pi) \nabla_\theta \pi(a_t|s_t) V^\pi(s_t,a_t) \\ &= \lim_{H\to\infty} \Bigg(\sum_{t=0}^{t^\pi_{\text{mix}}} \sum_{s_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} P(s_t|s,\pi) \nabla_\theta \pi(a_t|s_t) V^\pi(s_t,a_t) + \\ &\quad \sum_{t=t^\pi_{\text{mix}}}^{H} \sum_{s_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} \mu^\pi(s_t) \nabla_\theta \pi(a_t|s_t) V^\pi(s_t,a_t) \Bigg) \\ &\approx \lim_{H\to\infty} \sum_{t=t^\pi_{\text{mix}}}^{H} \sum_{s_t\in\mathcal{S}} \sum_{a_t\in\mathcal{A}} \mu^\pi(s_t) \nabla_\theta \pi(a_t|s_t) V^\pi(s_t,a_t) \\ &= \lim_{H\to\infty} \mathbb{E}_{\mu^\pi(s,a)}\Big[\nabla_\theta \log \pi(a|s) V^\pi(s,a)\Big]\end{aligned} \tag{3.29}$$

where $V^\pi(s,a)$ represents the undiscounted value function following π from start state s and action a. The second equality follows from a decomposition into the transient and limiting components of the long-term distribution. The first approximation follows from the fact that in the infinite horizon limit the reward on the limiting distribution will dominate the reward achieved on the transient distribution.

Following the policy improvement result from Theorem 3, we would like to update our bootstrapping strategy to better consider persistent changes to the policy in the limit of this type leveraging $\pi_m(a,s,\pi)$. As such instead of $V^\pi(s,a)$, we can consider $\lim_{H\to\infty} V^{\pi_m(a,s,\pi)}(s,a) = \lim_{H\to\infty} V^{\pi_m(a,s,\pi)}_{\text{trans}}(s,a) + (H - t^{\pi_m(a,s,\pi)}_{\text{mix}})\rho(\pi_m(a,s,\pi))$. Moreover, we can divide both sides by H to find $\nabla_\theta \rho(\pi)$:

$$\begin{aligned}\nabla_\theta \rho(\pi) &= \lim_{H\to\infty} \frac{1}{H} \nabla_\theta J^H_{\gamma=1}(s,\pi) \\ &= \lim_{H\to\infty} \mathbb{E}_{\mu^\pi(s,a)}\left[\frac{\nabla_\theta \log \pi(a|s)}{H}\left(V^{\pi_m(a,s,\pi)}_{\text{trans}}(s,a) + (H - t^{\pi_m(a,s,\pi)}_{\text{mix}})\rho(\pi_m(a,s,\pi))\right)\right] .\end{aligned} \tag{3.30}$$

Finally, applying L'Hôpital's rule once eliminates the dependence on the limit, yielding the desired result:

$$\begin{aligned}\nabla_\theta \rho(\pi) &= \mathbb{E}_{\mu^\pi(s,a)}\left[\nabla_\theta \log \pi(a|s)\rho(\pi_m(a,s,\pi))\right)\Big] \\ &= \sum_{s\in\mathcal{S}} \mu^\pi(s) \sum_{a\in\mathcal{A}} \nabla_\theta \pi(a|s)\rho(\pi_m(a,s,\pi)) .\end{aligned} \tag{3.31}$$

■

Now that we have established how to perform improvement of $\rho(\pi)$ directly, we would like to highlight advantages over standard approaches that only optimize on V^π_{trans}.

Proposition 4. *All finite improvements to the average reward per step $\rho(\pi)$ dominate all finite improvements in $V^\pi_{\text{trans}}(s)$ in their contribution to the undiscounted return.*

Proof. We first define the undiscounted return and what is meant by finite improvements to $V^\pi_{\text{trans}}(s)$ and $\rho(\pi)$. We then asses the ratio of these improvements and demonstrate that improvements to $\rho(\pi)$ dominate improvements to $V^\pi_{\text{trans}}(s)$ through the application of L'Hôpital's rule.

We begin by defining the infinite horizon undiscounted return of policy π, which we denote $V^\pi(s)$:

$$V^\pi(s) := V^\pi_{\text{trans}}(s) + V^\pi_{\text{lim}}(s) = \lim_{H\to\infty} V^\pi_{\text{trans}}(s) + (H - t^\pi_{\text{mix}})\rho(\pi) . \tag{3.32}$$

From this equation we can define a finite improvement c_{trans} to $V^\pi_{\text{trans}}(s)$ and denote it as $V_{+\text{trans}}{}^\pi(s)$:

$$V^\pi_{+\text{trans}}(s) := \lim_{H\to\infty} V^\pi_{\text{trans}}(s) + c_{\text{trans}} + (H - t^\pi_{\text{mix}})\rho(\pi) . \tag{3.33}$$

We then also proceed to define a finite improvement c_ρ to $\rho(\pi)$ and denote it as $V^\pi_{+\rho}(s)$:

$$V^{\pi}_{+\rho}(s) := \lim_{H\to\infty} V^{\pi}_{\text{trans}}(s) + (H - t^{\pi}_{\text{mix}})(\rho(\pi) + c_\rho) \tag{3.34}$$

Finally, we can demonstrate the dominating property discussed in the proposition by considering the ratio between the improved value functions $V_{+\text{trans}}{}^{\pi}(s)/V^{\pi}_{+\rho}(s)$:

$$\begin{aligned}\frac{V^{\pi}_{+\text{trans}}(s)}{V^{\pi}_{+\rho}(s)} &= \lim_{H\to\infty} \frac{V^{\pi}_{\text{trans}}(s) + c_{\text{trans}} + (H - t^{\pi}_{\text{mix}})\rho(\pi)}{V^{\pi}_{\text{trans}(s)} + (H - t^{\pi}_{\text{mix}})(\rho(\pi) + c_\rho)} \\ &= \frac{\rho(\pi)}{\rho(\pi) + c_\rho}\,.\end{aligned} \tag{3.35}$$

The second line follows from the application of L'Hôpital's rule. As we can see, the impact of c_ρ is still present in the limit, but c_{trans} is not and dominated by finite improvements to $\rho(\pi)$, thus proving the proposition.

■

Proposition 4 demonstrates how policy improvement towards the average reward results in faster changes in the undiscounted return. However, the question still remains of how much is lost by simply optimizing for ρ rather than the full undiscounted objective as advocated by [**41, 23, 46**]. For scalable MDPs of large complexity, the less stringent objective function is justified by the speedup in learning time.

Proposition 5. *For any scalable MDP $\mathbb{C}_\sigma$ the minimum learning time advantage needed to compensate for an algorithm that merely optimizes for $\rho(\pi)$ rather than the undiscounted return $\Delta t_{\text{learn}} \to 0$ as $|\mathcal{S}| \to \infty$.*

Proof. We begin the proof by defining the speed up in learning Δt_{learn} and comparing it to the regret resulting from learning a policy that merely optimizes for $\rho(\pi)$. This allows us to formalize the amount of speed up needed to justify this regret, which we demonstrate depends strongly on $|\mathcal{S}|$. Finally, we demonstrate that in the limit of $|\mathcal{S}| \to \infty$, the speed up needed to justify regret from only optimizing for the average reward effectively goes to zero.

Proof: The distinction highlighted in this proposition is between what are traditionally called *gain* and *bias* optimality in the average reward RL literature [**23**]. It is well known that both of these notions of optimality are optimal from the perspective of the average reward per step metric, but that bias optimality performs better on the transient distribution and thus achieves an additional constant factor of returns c through this improvement in performance [**23**]. In this proposition, we provoke the question of what slow down in learning Δt_{learn} is justified to ensure that we achieve this extra reward for the optimal policy of c. We will begin by defining Δt_{learn}:

$$\Delta t_{\text{learn}} = t_{\text{bias}} - t_{\text{gain}} \tag{3.36}$$

Here $t_{\text{bias}} \geq 0$ is the time taken to achieve the bias optimal policy and $t_{\text{gain}} \geq 0$ is the time taken to achieve the gain optimal policy. We further assume in this proposition that $t_{\text{bias}} \geq t_{\text{gain}}$ and therefor $\Delta t_{\text{learn}} \geq 0$. We can then consider the difference in regret of each of these algorithms, which in general should be greater than the lower bound on each algorithm's regret because regret must be non-negative based on the assumption that the algorithms accumulate regret at the same rate per step of sub-optimality in the general case:

$$\Delta\text{Regret} \geq \sqrt{|S|^{k+1}|A|t_{\text{bias}}} - \sqrt{|S|^{k+1}|A|t_{\text{gain}}} \tag{3.37}$$

Here the $|S|^k$ factor for some $k > 0$ comes from the definition of a polynomial mixing time. So if $\Delta\text{Regret} \geq c$, then the speed up in learning must justify the use of a gain optimality objective rather than a bias optimality objective as we do in our paper. Following from the previous equation, then the choice is also justified if this inequality holds:

$$\begin{aligned} \sqrt{|S|^{k+1}|A|t_{\text{bias}}} - \sqrt{|S|^{k+1}|A|t_{\text{gain}}} &\geq c \\ \sqrt{t_{\text{bias}}} - \sqrt{t_{\text{gain}}} &\geq \frac{c}{\sqrt{|S|^{k+1}|A|}} \end{aligned} \tag{3.38}$$

where the second line follows from dividing both sides by $\sqrt{|S|^{k+1}|A|}$. We then know that $\Delta t_{\text{learn}} \geq \sqrt{t_{\text{bias}}} - \sqrt{t_{\text{gain}}}$. Because both times are greater than or equal to zero. This then implies that, as indicated by Proposition 5, as $|\mathcal{S}| \to \infty$, the speed up in learning is justified if $\Delta t_{\text{learn}} \geq 0$. ■

3.15. Algorithms For Polynomial Mixing Times

With these new methods for policy improvement based on the average reward in mind, we now go on to propose three concrete algorithms to explore in our proof of concept experiments.

- We first propose **on-policy ρ-learning** detailed in Algorithm A.1.2 based on Theorem 3. The idea is similar to on-policy q-learning [**51**] in that optimal actions are considered as we enter each state based on our current approximation of the average reward per step resulting from each action.
- Next in Algorithm A.1.3 we propose **off-policy ρ-learning**. In this case our current policy π is used for interacting in the environment and random states are chosen for off-policy updates after each step.

3.16. Scalable Grid World Experiments

Apart from scaling τ and $\mathcal{Z}$, for the following experiments we also consider the scaling of the spatial dimensions $d$of the grid. We then empirically analyze the scaling behavior of our proposed models on these environment classes using accumulated lifelong regret per step and contrast this performance against relevant baselines. We perform an extensive hyperparameter search and pick the best hyperparameters for all models (including the baselines). For each experiment we report the mean and the standard deviation across 10 seeds. [6] Each example shares a region, $\mathcal{R}$, of size $d \times d$ with four directional actions.

Scaling the Spatial Dimensions: Here we consider scalable MDPs where the scaling factor is the spatial dimension d of a $d \times d$ gridworld.

Example 2. (Spatial dimensions, d): In this episodic task, the agent is placed at an arbitrary location in $\mathcal{R}$ and must reach a goal in an arbitrary location that is fixed across episodes. The agent is only rewarded upon reaching the goal location, which implies that the expected diameter of π^* is $\mathbb{E}[D^{\pi^*}] \in \Omega(d)$. Since $|\mathcal{S}| = d^2$ for this class, we have $\mathbb{E}[D^{\pi^*}] \in \Omega(|\mathcal{S}|^{\frac{1}{2}})$.

Grid Length d	*Steps*	*On-Policy Q-Learning*	*On-Policy ρ-Learning*	*Off-Policy Q-Learning*	*Dyna Q-Learning*	*Model-based n-step TD*	*Off-Policy ρ-Learning*
5	10k	$0.113_{\pm 0.028}$	$0.054_{\pm 0.003}$	$0.100_{\pm 0.058}$	$0.097_{\pm 0.046}$	$0.135_{\pm 0.029}$	$0.041_{\pm 0.003}$
	100k	$0.081_{\pm 0.028}$	$0.048_{\pm 0.004}$	$0.077_{\pm 0.028}$	$0.101_{\pm 0.011}$	$0.117_{\pm 0.024}$	$0.036_{\pm 0.003}$
25	10k	$0.062_{\pm 0.037}$	$0.033_{\pm 0.004}$	$0.058_{\pm 0.039}$	$0.061_{\pm 0.038}$	$0.060_{\pm 0.037}$	$0.057_{\pm 0.015}$
	100k	$0.057_{\pm 0.040}$	$0.016_{\pm 0.002}$	$0.056_{\pm 0.042}$	$0.060_{\pm 0.038}$	$0.059_{\pm 0.037}$	$0.019_{\pm 0.002}$

Table 3.1. Accumulated lifelong regret per step obtained by an agent in a scalable MDP featuring spatial scaling (Example 2).

The results for $d = 5{,}25$ are shown in Table 3.1. Our proposed algorithms consistently outperform baseline models in terms of lifelong regret.

Scaling $\mathcal{Z}$: As presented in Example 1, we consider scaling $\mathcal{Z}$ in a $\mathcal{Z} \times d \times d$ gridworld. Each grid has an arbitrary starting location when entering from the previous region and an arbitrary goal location serving as a bottleneck transporting the agent to the next region. The agent is only rewarded when it reaches the goal location of its current region, which implies that $\mathbb{E}[D^{\pi^*}_{\mathcal{R}_i}] \in \Omega(d)$ for all i. We consider three possibilities for how regions are connected.

(1) *Cycle Transitions:* the regions are accessed in a strict order with no repeats. We know that $\mathbb{E}[D^{\pi^*}] \in \Omega(d \times \mathcal{Z})$, so if $\mathcal{Z}$ is scaled with d fixed, $\mathbb{E}[D^{\pi^*}] \in \Omega(|\mathcal{S}|)$.

(2) *Random Transitions:* the regions are accessed randomly with repeats. We again know that $\mathbb{E}[D^{\pi^*}] \in \Omega(d \times \mathcal{Z})$, so again $\mathbb{E}[D^{\pi^*}] \in \Omega(|\mathcal{S}|)$.

(3) *Curricular Transitions:* the regions are accessed in a curricular fashion i.e. $\mathcal{R}_1$,$\mathcal{R}_1$,$\mathcal{R}_2$,$\mathcal{R}_1$,$\mathcal{R}_2$,$\mathcal{R}_3$, *etc.* We know $\mathbb{E}[D^{\pi^*}] \in \Omega(d \times \mathcal{Z}!)$, so if $\mathcal{Z}$ is scaled and d is fixed, $\mathbb{E}[D^{\pi^*}] \in \Omega(|\mathcal{S}|^{\mathcal{Z}-1})$.

[6] We also provide our code for easy reproduction of experiments https://github.com/SharathRaparthy/polynomial-mixing-times.

No. of Rooms $\mathcal{Z}$	*Task type*	*Steps*	*On-Policy Q-Learning*	*On-Policy ρ-Learning*	*Off-Policy Q-Learning*	*Dyna Q-Learning*	*Model-based n-step TD*	*Off-Policy ρ-Learning*
4	*Random*	10k	$0.367_{\pm 0.138}$	$0.162_{\pm 0.040}$	$0.396_{\pm 0.230}$	$0.300_{\pm 0.124}$	$0.274_{\pm 0.032}$	$0.180_{\pm 0.038}$
		100k	$0.279_{\pm 0.059}$	$0.152_{\pm 0.038}$	$0.242_{\pm 0.090}$	$0.222_{\pm 0.049}$	$0.201_{\pm 0.032}$	$0.153_{\pm 0.040}$
	Cycles	10k	$0.271_{\pm 0.115}$	$0.074_{\pm 0.030}$	$0.287_{\pm 0.161}$	$0.171_{\pm 0.070}$	$0.229_{\pm 0.075}$	$0.099_{\pm 0.033}$
		100k	$0.165_{\pm 0.056}$	$0.063_{\pm 0.029}$	$0.120_{\pm 0.045}$	$0.120_{\pm 0.045}$	$0.130_{\pm 0.042}$	$0.065_{\pm 0.029}$
16	*Random*	10k	$0.365_{\pm 0.230}$	$0.138_{\pm 0.014}$	$0.348_{\pm 0.178}$	$0.410_{\pm 0.087}$	$0.355_{\pm 0.105}$	$0.292_{\pm 0.030}$
		100k	$0.303_{\pm 0.092}$	$0.091_{\pm 0.009}$	$0.321_{\pm 0.095}$	$0.187_{\pm 0.017}$	$0.334_{\pm 0.030}$	$0.106_{\pm 0.010}$
	Cycles	10k	$0.338_{\pm 0.162}$	$0.100_{\pm 0.026}$	$0.303_{\pm 0.152}$	$0.364_{\pm 0.084}$	$0.412_{\pm 0.114}$	$0.287_{\pm 0.041}$
		100k	$0.243_{\pm 0.067}$	$0.062_{\pm 0.017}$	$0.363_{\pm 0.122}$	$0.144_{\pm 0.028}$	$0.181_{\pm 0.04}$	$0.083_{\pm 0.017}$
2	*Curricular*	10k	$0.452_{\pm 0.128}$	$0.343_{\pm 0.071}$	$0.379_{\pm 0.0.095}$	$0.400_{\pm 0.086}$	$0.408_{\pm 0.096}$	$0.354_{\pm 0.080}$
		100k	$0.424_{\pm 0.109}$	$0.340_{\pm 0.067}$	$0.362_{\pm 0.098}$	$0.366_{\pm 0.092}$	$0.384_{\pm 0.087}$	$0.340_{\pm 0.070}$
3	*Curricular*	10k	$0.359_{\pm 0.157}$	$0.281_{\pm 0.043}$	$0.389_{\pm 0.234}$	$0.322_{\pm 0.091}$	$0.331_{\pm 0.118}$	$0.260_{\pm 0.055}$
		100k	$0.306_{\pm 0.097}$	$0.259_{\pm 0.073}$	$0.306_{\pm 0.097}$	$0.300_{\pm 0.076}$	$0.285_{\pm 0.076}$	$0.250_{\pm 0.063}$
4	*Curricular*	10k	$0.283_{\pm 0.059}$	$0.180_{\pm 0.056}$	$0.311_{\pm 0.094}$	$0.246_{\pm 0.042}$	$0.286_{\pm 0.074}$	$0.195_{\pm 0.071}$
		100k	$0.245_{\pm 0.069}$	$0.161_{\pm 0.053}$	$0.207_{\pm 0.037}$	$0.210_{\pm 0.045}$	$0.209_{\pm 0.055}$	$0.183_{\pm 0.076}$

Table 3.2. Accumulated lifelong regret per step obtained by an agent in a scalable MDP featuring scaling $\mathcal{Z}$. The values shown are for the three room transition variants across different $\mathcal{Z}$ values with each room of size $d = 5$.

The results for $\mathcal{Z} = 4, 16$ and $d = 5$ are shown in Table 3.2. Our models consistently outperform baselines for both cyclic and random transitions regardless of the number of rooms. The curricular transition case is challenging since the diameter scales exponentially with N, making the diameter substantially higher. The corresponding convergence rates of all methods (Table 3.2) are lower as expected by regret bounds [**15**]. Nevertheless, our proposed models still outperform the baselines.

Scaling τ: In this case, we scale the amount of time spent in each room. We use cyclic transitions to highlight similarities to typical settings in continual RL [**19**]. Here, room transitions are passive: the agents' current policy has no direct effect on the room transitions which purely depend on τ. $\tau \geq 2d$ as otherwise some regions would be impossible to solve, so we can assume a form $\tau = cd^x \quad \forall c \geq 2, x \geq 1$. This implies that as we scale x keeping c, d, and N constant, $\mathbb{E}[D^{\pi^*}] \in \Omega(|\mathcal{S}|^{x/2})$. The results for $x = 2, 3$ and 4 keeping $d = 5$ and $N = 16$ fixed are shown in Table 3.3. As expected, our proposed methods achieve better sample efficiency.

Exponent x	*Steps*	*On-Policy Q-Learning*	*On-Policy ρ-Learning*	*Off-Policy Q-Learning*	*Dyna Q-Learning*	*Model-based n-step TD*	*Off-Policy ρ-Learning*
2	10k	$0.245_{\pm 0.016}$	$0.125_{\pm 0.008}$	$0.286_{\pm 0.024}$	$0.240_{\pm 0.018}$	$0.279_{\pm 0.025}$	$0.216_{\pm 0.010}$
	100k	$0.243_{\pm 0.013}$	$0.062_{\pm 0.002}$	$0.233_{\pm 0.020}$	$0.124_{\pm 0.012}$	$0.201_{\pm 0.012}$	$0.070_{\pm 0.002}$
3	10k	$0.233_{\pm 0.024}$	$0.089_{\pm 0.005}$	$0.253_{\pm 0.024}$	$0.262_{\pm 0.012}$	$0.262_{\pm 0.012}$	$0.252_{\pm 0.016}$
	100k	$0.197_{\pm 0.019}$	$0.049_{\pm 0.003}$	$0.194_{\pm 0.022}$	$0.121_{\pm 0.011}$	$0.168_{\pm 0.013}$	$0.072_{\pm 0.003}$
4	10k	$0.220_{\pm 0.036}$	$0.066_{\pm 0.007}$	$0.243_{\pm 0.046}$	$0.269_{\pm 0.085}$	$0.243_{\pm 0.051}$	$0.236_{\pm 0.034}$
	100k	$0.185_{\pm 0.014}$	$0.047_{\pm 0.003}$	$0.168_{\pm 0.014}$	$0.140_{\pm 0.012}$	$0.163_{\pm 0.015}$	$0.089_{\pm 0.007}$

Table 3.3. Accumulated lifelong regret per step obtained by an agent in a scalable MDP featuring scaling τ. The values shown are for the *cyclic* room transitions with $\mathcal{Z} = 16$ rooms.

Chapter 3

Conclusion

In this work, we have considered the implicit premise of the continual RL literature and analyzed the factors affecting learning in this setting. Specifically, we focused on *Mixing times* of a Markov chain induced by the policy π. We first formalized the continual RL setting with the help of the so-called *Scalable MDPs* in Section 3.5. We then formally introduced mixing times and their various variants. Although this quantity can be substantial in continual RL settings, we theoretically argued that the problems of long-term interest in continual RL are those whose mixing times scale at least by a polynomial factor with the state-space size (Definition 15) and proved that the scalable MDPs indeed have polynomial mixing times (Theorem 2). We validated our theoretical claims by measuring the mixing times of pretrained policies in Atari and MuJoco environments. The results from currently manageable small-scale experiments are indicative of performance for the large-scale aspirational use cases of the future. In particular, we have highlighted that mixing times will scale significantly as the problems we deal with are scaled and how traditional approaches to RL are ill-suited to deal with this.

In the context of commonly considered problems that follow the high-level structure of Example 1, we have showcased the deep connection between catastrophic forgetting and myopic bias in the presence of very large mixing times. Moreover, this perspective naturally points towards approaches that directly reason over the steady-state distribution as a natural solution to the catastrophic forgetting problem. To verify this, we consider simple tabular RL experiments across 3 example classes of scalable MDPs (spatial scaling, bottleneck scaling and scaling τ). Our analysis demonstrates that on-policy and off-policy versions of Q-Learning that perform policy evaluation using an estimate of the steady-state distribution derived from matrix inversion of a tabular environment model consistently outperform standard model-based and model-free baselines in terms of lifelong regret as environments are scaled up. Even though estimating the steady-state distribution would clearly be quite

challenging in complex state and observation spaces, recent approaches have made great strides towards developing practical and scalable approaches leveraging buffers.

Finally, to conclude, the focus of our work is on highlighting the difficulty of the polynomial mixing time problem and proposing some small-scale solutions methods. We believe that our analysis provides significant insights that the community will be able to draw on to build better and more theoretically grounded methods for continual RL.

References

[1] AGI-Labs. Continual rl project. `https://github.com/AGI-Labs/continual_rl`, 2021.

[2] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew E. Taylor. Online multi-task learning for policy gradient methods. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1206–II–1214. JMLR.org, 2014.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[5] Richard Bellman. *Dynamic Programming*. Dover Publications, 1957.

[6] Emmanuel Bengio, Joelle Pineau, and Doina Precup. Correcting momentum in temporal difference learning. *arXiv preprint arXiv:2106.03955*, 2021.

[7] Glen Berseth, Cheng Xie, Paul Cernek, and Michiel van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. *CoRR*, abs/1802.04765, 2018.

[8] Dimitri P Bertsekas. A new value iteration method for the average cost dynamic programming problem. *SIAM journal on control and optimization*, 36(2):742–759, 1998.

[9] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[10] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[12] Gene H Golub and Charles F Van Loan. Matrix computations. johns hopkins studies in the mathematical sciences, 1996.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

[16] Michael Janner, Igor Mordatch, and Sergey Levine. gamma-models: Generative temporal difference learning for infinite-horizon prediction. *arXiv preprint arXiv:2010.14496*, 2020.

[17] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *CoRR*, abs/1902.00255, 2019.

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

[19] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526, 2017.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[22] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[23] Sridhar Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1):159–195, 1996.

[24] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[26] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[27] Abhishek Naik, Roshan Shariff, Niko Yasui, Hengshuai Yao, and Richard S Sutton. Discounted reinforcement learning is not an optimization problem. *arXiv preprint arXiv:1910.02140*, 2019.

[28] Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? *CoRR*, abs/1906.07073, 2019.

[29] Ian Osband and Benjamin Van Roy. On lower bounds for regret in reinforcement learning. *arXiv preprint arXiv:1608.02732*, 2016.

[30] James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.

[31] James Gary Propp and David Bruce Wilson. How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

[32] ML Puterman. Markov decision processes. 1994. *Jhon Wiley & Sons, New Jersey*, 1994.

[33] Antonin Raffin. Rl baselines3 zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

[34] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *CoRR*, abs/1810.11910, 2018.

[35] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[36] Matthew Riemer, Tim Klinger, Djallel Bouneffouf, and Michele Franceschini. Scalable recollections for continual lifelong learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1352–1359, 2019.

[37] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32:350–360, 2019.

[38] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[39] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[41] Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, volume 298, pages 298–305, 1993.

[42] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537, 2018.

[43] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[44] Richard Sutton. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural information Processing Systems*, 12:1057–1063, 2000.

[45] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[46] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[48] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[49] Yi Wan, Abhishek Naik, and Richard S Sutton. Learning and planning in average-reward markov decision processes. *arXiv preprint arXiv:2006.16318*, 2020.

[50] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

[51] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[52] Tom Zahavy, Alon Cohen, Haim Kaplan, and Yishay Mansour. Unknown mixing times in apprenticeship and reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 430–439. PMLR, 2020.

Appendix A

Appendix

A.1. Proposed Algorithms

Estimating The Mixing Time. In this section, we begin by providing pseudo-code for mixing time estimation as outlined in the main text. For the detailed algorithm please refer Algorithm A.1.1. Regarding the hyperparameters, for all our experiments we considered $\mathcal{H} = \tau \times 10^3$. For the experiments reported in Figure 3 and 5, we report the mean and standard deviation across 3 seeds. For Figure 4 experiments, in order to remove the task bias, we randomly shuffle the list of environments we consider and report the mean and standard deviation across 10 seeds.

Algorithm A.1.1 ϵ- Return Mixing Time Estimation

procedure MIXINGTIMEESTIMATION(env,ϵ, π, $|\hat{\mathcal{S}}|_{\max}$) `// Calculate the asymptotic` h-`step return of the policy` π

$s_h \leftarrow env.reset()$
$G(\pi) \leftarrow 0$
Initialize $\mathcal{H}$ `// Horizon length for` $\rho(\pi)$ `calculation`
for h *in* $1, 2, \ldots \mathcal{H}$ **do**
  $a_h \leftarrow \pi(s_h)$
  $s_{h+1}, r_h \leftarrow env.step(a_h)$
  $G(\pi) = G(\pi) + r_h$
end
$\rho(\pi) = G(\pi)/\mathcal{H}$ `// Asymptotic h-step return`
`// Store the reward history for each state`
StateRewardHistory $\leftarrow$ dict()
$s_h \leftarrow env.reset()$
for h *in* $1, 2, \ldots \mathcal{H}$ **do**
  $a_h \leftarrow \pi(s_h)$
  $s_{h+1}, r_h \leftarrow env.step(a_h)$
  $s_{idx} \leftarrow random.randint(1, h)$
  if $s_{idx} < |\hat{\mathcal{S}}|_{max}$ **then**
    StateRewardHistory[s_{idx}] $\leftarrow$ [ ]
  else
    for s_i *in StateRewardHistory* **do**
      StateRewardHistory[s_i].append(r_h)
    end
  end
end
`// Calculate the mixing time using` $\rho(\pi)$ `and StateRewardHistory`
$t^{\pi}_{\text{ret}}(\epsilon) \leftarrow$ [ ]
for *state in StateRewardHistory* **do**
  rewards $\leftarrow$ StateRewardHistory[state]
  $t_{\text{mix}} \leftarrow$ [ ]
  $h \leftarrow 0$
  $G(h, state, \pi) \leftarrow 0$
  for r *in rewards* **do**
    $h \leftarrow h + 1$
    $G(h, state, \pi) \leftarrow G(h, state, \pi) + r$
    $\rho(h, state, \pi) \leftarrow G(h, state, \pi)/h$
    if $|\rho(h, state, \pi) - \rho(\pi)| < \epsilon$ **then**
      $t_{\text{mix}}.append(h)$
    else
      $t_{\text{mix}} \leftarrow$ [ ]
    end
  end
  $t^{\pi}_{\text{ret}}(\epsilon).append(\min(t_{\text{mix}}))$ `// Add` $\min(t_{\text{mix}})$ `to` $t^{\pi}_{\text{ret}}(\epsilon)$
end
return Mean($t^{\pi}_{\text{ret}}(\epsilon)$)
end procedure

We now provide pseudo-code for our proposed RL algorithms discussed in the main text. First, we highlight On-Policy ρ-Learning in Algorithm A.1.2.

Algorithm A.1.2 On-Policy ρ-Learning

procedure OnPolicyρLearning(env,ϵ)
  Initialize π, $\hat{T}$ *and* $\hat{R}$
  $s_t \leftarrow env.reset()$
  while *not done* **do**
    $p \sim uniform([0,1])$
    if $p \geq \epsilon$**:**
      for $a \in \mathcal{A}$**:**
        Solve for $\hat{\mu}^{\pi_m(s_t,a,\pi)}$ (equation 3.22)
        Solve for $\hat{\rho}(\pi_m(s_t,a,\pi))$ (equation 3.21)
      $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{\rho}(\pi_m(a,s_t,\pi))$
      Update to greedy policy: $\pi = \pi_m(a_t,s_t,\pi)$
    else:
      Random exploration: $a_t \sim uniform(a \in \mathcal{A})$
    $s_{t+1}, r_t \leftarrow env.step(s_t, a_t)$
    Update $\hat{T}$ *on* (s_t,a_t,s_{t+1}) *and* $\hat{R}$ *on* (s_t,a_t,r_t)
    Update to next state: $s_t = s_{t+1}$
  end
  return π,$\hat{T}$,$\hat{R}$
end procedure

We now detail Off-Policy ρ-Learning in Algorithm A.1.3.

Algorithm A.1.3 Off-Policy ρ-Learning

procedure OFFPOLICYρLEARNING(env,ϵ,B)
  Initialize π, $\hat{T}$ *and* $\hat{R}$
  $s_t \leftarrow env.reset()$
  while *not done* **do**
    $p \sim uniform([0,1])$
    if $p \geq \epsilon$**:**
      Sample an action: $a_t \sim \pi(s_t)$
    else:
      Random exploration: $a_t \sim uniform(a \in \mathcal{A})$
    $s_{t+1}, r_t \leftarrow env.step(s_t, a_t)$
    Update $\hat{T}$ *on* (s_t,a_t,s_{t+1}) *and* $\hat{R}$ *on* (s_t,a_t,r_t)
    Update to next state: $s_t = s_{t+1}$
    for $i \in [0, ..., B-1]$:
      $s_i \sim uniform(s \in \mathcal{S})$
      for $a \in \mathcal{A}$**:**
        Solve for $\hat{\mu}^{\pi_m(s_i,a,\pi)}$ (equation 3.22)
        Solve for $\hat{\rho}(\pi_m(s_i,a,\pi))$ (equation 3.21)
      $a_i = \text{argmax}_{a \in \mathcal{A}} \hat{\rho}(\pi_m(a,s_i,\pi))$
      Update to greedy policy: $\pi = \pi_m(a_i,s_i,\pi)$
  end
  return π,$\hat{T}$,$\hat{R}$
end procedure