Unknown mixing times in apprenticeship and reinforcement learning

Tom Zahavy[1,2], **Alon Cohen**[1,2], **Haim Kaplan**[1,3] **and Yishay Mansour**[1,3*]
{*tomzahavy*, *aloncohen*, *haimk*, *mansour*}@*google.com*
[1] Google AI, Tel Aviv
[2] Technion, Israel Institute of Technology
[3] Tel Aviv University

Abstract

We derive and analyze learning algorithms for apprenticeship learning, policy evaluation, and policy gradient for average reward criteria. Existing algorithms explicitly require an upper bound on the mixing time. In contrast, we build on ideas from Markov chain theory and derive sampling algorithms that do not require such an upper bound. For these algorithms, we provide theoretical bounds on their sample-complexity and running time.

1 INTRODUCTION

Reinforcement Learning (RL) is an area of machine learning concerned with how agents learn long-term interactions with their environment (Sutton & Barto, 1998). The agent and the environment are modeled as a Markov Decision Process (MDP). The agent's goal is to determine a policy that maximizes her cumulative reward. Much of the research in RL focuses on episodic or finite-horizon tasks. When studying infinite-horizon tasks, the standard approach is to discount future rewards. Discounting serves two purposes: first, it makes the cumulative reward bounded; second, in some domains, such as economics, discounting is used to represent "interest" earned on rewards. Thus an action that generates an immediate reward is preferable over one that generates the same reward in the future. Nevertheless, discounting is unsuitable in general domains. Alternatively, it is common to maximize the expected reward received in the steady-state of the Markov chain defined by the agent's policy. This is the case in many control problems: elevators, drones, climate control, etc. (Bertsekas et al., 2005) as well as many sequential decision-making problems such as inventory-management (Arrow et al., 1958) and queuing (Kelly, 1975).

Blackwell (1962) pioneered the study of MDPs with average-reward criteria. He showed that the optimal policy for the average reward is the limit of the sequence of optimal policies for discounted reward as the discount factor converges to 1. However, it has been established that it is computationally hard to find the optimal policy when the discount factor is close to 1. For these reasons, Dynamic Programming (DP) algorithms were developed for average-reward criteria (see Mahadevan, 1996; Puterman, 1984; for detailed surveys). Howard (1960) introduced the policy-iteration algorithm. Value iteration was later proposed by White (1963). However, these algorithms require knowledge of the state transition probabilities and are also computationally intractable.

The main challenge in deriving RL algorithms for average-reward MDPs is calculating the stationary distribution of the Markov chain induced by a given policy. This is a necessary step in evaluating the average reward of the policy. When the transition probabilities are known, the stationary distribution can be obtained by solving a system of linear equations. In the reinforcement learning setup, however, the dynamics are unknown, and practitioners tend to "run the simulation for a sufficiently long time to obtain a good estimate" (Gosavi, 2014). This implicitly implies that the learner knows a bound on the mixing time T_{mix} of the Markov chain. Indeed, model-free algorithms for the average reward with theoretical guarantees (e.g., Wang, 2017; Chen et al., 2018) require an upper bound on the mixing time as an input.

We will later see that **not knowing such a bound on the mixing time comes with an additional cost** and requires $O(T_{\text{mix}}|S|)$ samples to get a single sample from the stationary distribution. Instead, one might consider learning the transition probability matrix and use it for

*Supported in part by grant from the Israel Science Foundation

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

computing the stationary distribution (i.e., model-based RL). Not surprisingly, model-based algorithms assume that the mixing time, or an upper bound on it, are known explicitly (Kearns et al., 2000; Brafman & Tennenholtz, 2002).[1] Moreover, even if we estimate the transition probabilities, it is not clear how to use it to obtain the average reward. In particular, the stationary distribution in the estimated model is not guaranteed to be close to the stationary distribution of the true model; an equivalent of the simulation lemma (Kearns & Singh, 2002) for this setup does not exist. To illustrate the difficulty, consider a periodic Markov chain with states ordered in a (deterministic) cycle. Also consider a similar Markov chain, but with a probability of ε to remain in each state. Even though the two models are "close" to each other, the latter chain is ergodic and does have a stationary distribution while the former chain is periodic and therefore does not have a stationary distribution **at all**.

Alternatively, one may consider estimating the mixing time (or an upper bound on it) directly, in order to use it to get samples from the stationary distribution. There are two drawbacks to this approach. First, the sample complexity for estimating the mixing time is quite significant. For an arbitrary ergodic Markov chain, it is possible to estimate an upper and a lower bound on the mixing time by approximating the pseudo-spectral gap and the minimal stationary probability $\pi_\star$ (Levin et al., 2017, Theorems 12.3 and 12.4), and estimating these quantities to within a relative error of ε requires $O\left(T_{\text{mix}}^2 \max\{T_{\text{mix}}, |S|/\pi_\star\}/\varepsilon^2\pi_\star\right)$ samples (Wolfer & Kontorovich, 2019). Second, these techniques can be used to get an upper bound on the mixing time of a single policy, and not on the maximum of all the deterministic policies in an MDP. For these reasons we focus on algorithms that avoid estimating the mixing time directly.

In this work, we build on Coupling From the Past (CFTP) – a technique from Markov chain theory that obtains unbiased samples from a Markov chain's stationary distribution (Propp & Wilson, 1996, 1998). These samples are generated *without any prior knowledge on the mixing time of the Markov chain*. Intuitively, CFTP starts $|S|$ parallel simulations of the Markov chain, one from each state, at minus infinity. When two simulations reach the same state, they continue together as one simulation. The simulations coalesce at time zero to a single sample state, which is distributed exactly as the stationary distribution. CFTP, rather than starting at minus infinity, starts at zero and generates suffixes of increasing length of this infinite simulation until it can identify the state at which all simulations coalesce. The simulations are shown to coalesce, in expectation, after $O\left(|S|T_{\text{mix}}\right)$-time. In Section 2, we provide an alternative, simple proof of the coalescence-time of the CFTP procedure and a matching lower bound. Additionally, we analyze the time it takes for two simulations to coalesce and show how to use this process to estimate differences of Q-values.

We further describe sampling-based RL algorithms for the average-reward criteria that utilize these ideas. The main advantage of our algorithms is that they do not require a bound on the associated Markov chain's mixing time. In Section 3, we consider apprenticeship learning and propose two sampling mechanisms to evaluate the game matrix and analyze their sample complexity. These are: using the CFTP protocol to estimate the game matrix directly at the beginning of the algorithm; querying the expert, at each step, for two trajectories to provide an unbiased estimate of the game matrix. We also include an unbiased estimator of the policy gradient (under average reward criteria) and analyze its sample complexity. Finally, in the supplementary material (Section A), we use CFTP to propose a sample-efficient data structure that allows us to get an unbiased sample from the stationary distribution of any policy in an MDP.

1.1 PRELIMINARIES

In this section, we provide background on RL with **average reward criteria** (based on Puterman, 1984), as well as on the **CFTP** algorithm (Propp & Wilson, 1996, 1998) for getting unbiased samples from a stationary distribution of a Markov chain without knowing its mixing time. Background on apprenticeship learning is provided in the relevant section.

A Markov Decision Process (MDP) consists of a set of states S, and a set of actions A. We assume that S and A are finite. Associated with each action $a \in A$ is a state transition matrix P^a, where $P^a(x, y)$ represents the probability of moving from state x to y under action a. There is also a *stochastic reward* function $R : S \times A \mapsto \mathbb{R}$ where $r(x, a) = \mathbb{E}[R(s, a)]$ is the expected reward when performing action a in state x. A stationary deterministic policy is a mapping $\pi : S \mapsto A$ from states to actions. Any policy induces a state transition matrix P^π, where $P^\pi(x, y) = P^{\pi(x)}(x, y)$. Thus, any policy yields a Markov chain (S, P^π). The **stationary distribution** μ of a Markov chain with transition matrix P is defined to be the probability distribution satisfying $\mu^\top = \mu^\top P$.

We specifically study *ergodic MDPs* in which any pol-

[1] UCRL2 (Jaksch et al., 2010) avoids using the mixing time but instead assumes knowledge of the MDP diameter (which is implicitly related to the mixing time) to guarantee an ε–optimal policy. We emphasize that there is no need to know a bound on the diameter in the regret setting, but only when the goal is to learn an ε-optimal policy. In this case, the learner has to know a bound on the diameter in order to bound the sample complexity.

icy induces an ergodic Markov chain. That is, a Markov chain which is irreducible and aperiodic (Levin et al., 2017). Such a Markov chain converges to a unique stationary distribution independent of the starting state (for generalizations to unichain MDPs, see Puterman, 1984). The **average reward** $\rho(\pi)$ ssociated with a particular policy π is defined as $\rho(\pi) = \mathbb{E}_{x\sim\mu(\pi)}\, r(x, \pi(x))$ where $\mu(\pi)$ is the stationary distribution of the Markov chain induced by π. The *optimal policy* is one that maximizes the average reward. The ***Q*-value** of a state-action pair given a policy π is defined as

$$Q^{\pi}(s,a) = \sum_{t=0}^{\infty} \mathbb{E}\left\{r_t - \rho(\pi) \mid s_0 = s, a_0 = a, \pi\right\}. \qquad (1)$$

We define the **total-variation distance** for two probability measures P and Q on a sample space Ω to be $\mathrm{TV}[P,Q] = \sup_{A\subseteq\Omega}|P(A) - Q(A)|$ (which is equivalent to the $L1$ distance). Informally, this is the largest possible difference between the probabilities that the two distributions assign to the same event.

The **mixing time** of an ergodic Markov chain with a stationary distribution μ is the smallest t such that $\forall x_0,\ \mathrm{TV}[\Pr_t(\cdot|x_0), \mu] \leq 1/8$, where $\Pr_t(\cdot|x_0)$ is the distribution over states after t steps, starting from x_0. For MDP M, let T^{π}_{mix} be the mixing time of the Markov chain which π induces in M, i.e., (S, P^{π}). The **MDP mixing time**, $\bar{T}_{\mathrm{mix}} = \max_{\pi\in\Pi} T^{\pi}_{\mathrm{mix}}$ is the maximal mixing time for any deterministic policy.

The algorithms presented in this paper rely on access to a **generative model** (Kearns & Singh, 2002); an oracle that accepts a state-action pair (s, a) and outputs a state s' that is drawn from the next-state distribution $P^a(s,\cdot)$, and a sample from the reward distribution $R(s,a)$. We further assume that a sample is generated in unit time, and measure the *sample complexity* of an algorithm by the number of calls it makes to the generative model.

Algorithm 1 Coupling from the past

$F_0 \leftarrow$ Identity Map, $t \leftarrow (-1)$
repeat
  $t \leftarrow t + 1$
  $f_{-(t+1)} \leftarrow$ RandomMap(P)
  $F_{-(t+1)} \leftarrow F_{-t} \circ f_{-(t+1)}$
until $F_{-(t+1)}$ is constant
Return the value into which $F_{-(t+1)}$ coalesces

Coupling From the Past (CFTP) is a method for sampling from the stationary distribution of a Markov chain (Propp & Wilson, 1996, 1998). Contrary to many Markov Chain Monte-Carlo algorithms, Coupling from the past gives a perfect sample from the stationary distribution. Intuitively, CFTP starts $|S|$ parallel simulations of the Markov chain, one from each state, at minus infinity. When two simulations reach the same state, they continue together as one simulation. The simulations coalesce at time zero to a single sample state, which is distributed exactly as the stationary distribution. CFTP rather than starting at minus infinity starts at zero and generates suffixes of increasing length of this infinite simulation until it can identify the state at which all simulations coalesce (Häggström, 2002).

Consider a finite state ergodic Markov chain M with state space S, a transition probability matrix P. CFTP generates a sequence of mappings $F_0, F_{-1}, F_{-2}, \ldots$ each from S to S, until the first of these mappings, say F_{-t} is constant, sending all states into the same one. In other words, F_{-t} defines simulations from every starting state, that *coalesce* into a single state after t steps. Initially $F_0(s) = s$ for every $s \in S$. Then we generate $F_{-(t+1)}$ by drawing a random map $f_{-(t+1)} : S \mapsto S$ (denoted by RandomMap(P)) where we pick $f_{-(t+1)}(s)$ from the next state distribution $P(s,\cdot)$ (e.g., the Markov chain dynamics) for every s, and compose $f_{-(t+1)}(s)$ with F_{-t}.

Theorem 1. *With probability 1, the CFTP protocol returns a value, which is distributed according to the Markov chain's stationary distribution.*

See Propp & Wilson (1996) for proof. Additionally, Theorem 5 in Propp & Wilson (1998) states that the expected value of t when F_{-t} coalesces is $O\big(T_{\mathrm{mix}}|S|\big)$. The straightforward implementation of Algorithm 1 takes $O(|S|)$ time per step for a total of $O\big(T_{\mathrm{mix}}|S|^2\big)$ time. Propp & Wilson (1996) also give a cleverer implementation that takes $O\big(T_{\mathrm{mix}}|S|\log|S|\big)$ time. It uses the fact that coalescence occurs gradually and reduces the number of independent simulations as time progresses.

1.2 EXAMPLE

We finish this section with a motivating simulation, where we compare the CFTP procedure with a common practice of "guessing" the mixing time and running the chain for that time. While CFTP does not suffer from bias at all, the baseline methods do suffer from bias and are shown to produce errors in estimating the average reward. If the guess is too large, then these methods are highly sample-inefficient compared to CFTP.

Consider the Markov reward process in Fig. 1. The initial state distribution μ_0 is given by $\mu_0 = (0, 1)$ (starts from the right state). The stationary distribution is $(\frac{2}{3}, \frac{1}{3})$, the average reward is $\frac{2}{3}$, the expected coalescence time is 2, and the mixing time is 4. For this chain, simulating from time 0 forward until all chains coalesce gives a biased sample, as coalescence can only occur in the left state.

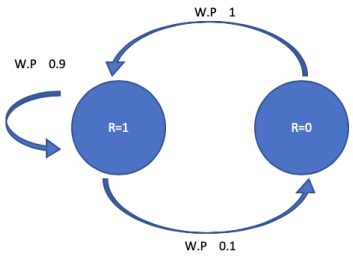

Figure 1: Markov chain

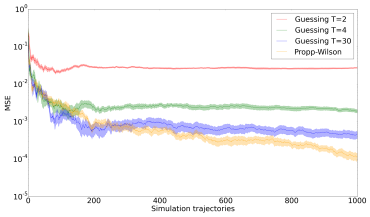

Figure 2: MSE vs. runs

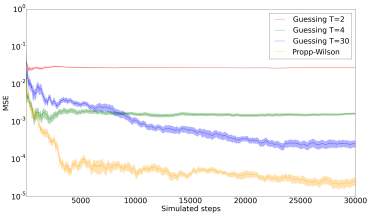

Figure 3: MSE vs. steps

We implemented the CFTP procedure[2] and compared it with a baseline that "guesses" the mixing time, T_{guess}. This baseline operates as follows. First, it samples an initial state from μ_0. Then, it simulates the chain for T_{guess} steps. Finally, it returns the reward at the resulting state.

We run each algorithm to obtain a sample from the stationary distribution, and use the average of these samples to estimate the average reward. For each algorithm, we report the average (over 10 runs, reported alongside error bars) Mean Squared Error (MSE) with respect to the average reward as a function of the number of runs taken (Fig. 2). We can see that underestimating the mixing time $T_{\text{guess}} = 2$ (red), and using precisely the mixing time $T_{\text{guess}} = 4$ (green) leads to bias in the estimation of the average reward. The latter is due to the fact that by the definition of the mixing time, we are not guaranteed to sample exactly from the stationary distribution, but only from a distribution that is close it in the total variation distance. When we overestimate the mixing time, e.g., for $T_{\text{guess}} = 30$ (blue), the bias decreases significantly. Similarly, we can see that CFTP (orange) produces unbiased samples as expected.

The advantage of CFTP (orange) becomes clearer when inspecting Fig. 3. We can see the MSE as a function of the number of simulation steps. Overestimating the mixing time $T_{\text{guess}} = 30$ (blue), still gives unbiased estimates but uses too many simulation steps to achieve a single sample of the reward. As a result, CFTP (orange) yields much lower MSE for the same amount of samples.

2 SAMPLING FROM A STATIONARY DISTRIBUTION WITH UNKNOWN MIXING TIME

2.1 COALESCENCE FROM TWO STATES

We begin this section by analyzing a simple scenario: for a Markov chain with $|S|$ states, we simultaneously start two simulations from two different states. At each time step, each simulation proceeds according to the next state distribution of the Markov chain. We are interested in bounding the time that it takes for **two** simulations to reach the same state. As we will see, this takes $O(T_{\text{mix}}|S|)$ time in expectation. We begin with the following two lemmas.

Lemma 2. *Let P and Q be distributions on $\{1, \ldots, |S|\}$ such that $TV[P, Q] \leq \frac{1}{4}$. Draw x from P and y from Q independently. Then* $\Pr[x = y] \geq \frac{1}{2|S|}$.

Proof. Let $B = \{i \in \{1, \ldots, |S|\} : P(i) > Q(i)\}$ and note that, by definition of the total variation distance, $P(B) - Q(B) = \text{TV}[P, Q] \leq 1/4$. We have that

$$\Pr[x = y] = \sum_{i=1}^{|S|} P(i)Q(i) \geq \sum_{i \in B} Q(i)^2 + \sum_{i \in B^c} P(i)^2$$
$$\geq \frac{\big(Q(B) + P(B^c)\big)^2}{|S|}. \quad \text{(Cauchy-Schwartz)}$$

The proof is completed by noticing that $Q(B) + P(B^c) = 1 - \big(P(B) - Q(B)\big) \geq 3/4$. □

Lemma 3. *Let i and j be two states of an ergodic Markov chain of $|S|$ states. Let x be the state reached after making T_{mix} steps starting from i, and let y be the state reached after making T_{mix} steps starting from j. Then* $\Pr[x = y] \geq \frac{1}{2|S|}$.

Proof. Let P be the distribution on states after making T_{mix} steps starting from i, and let Q be the distribution on states after making T_{mix} steps starting from j. Let μ be the stationary distribution of the Markov chain. Then by the definition of T_{mix}, we have that $\text{TV}[P, \mu] \leq 1/8$ and $\text{TV}[Q, \mu] \leq 1/8$. Therefore, $\text{TV}[P, Q] \leq 1/4$, and by Lemma 2 we have $\Pr[x = y] \geq \frac{1}{2|S|}$. □

By repeating the argument of the previous Lemma, we arrive at the following conclusion.

[2] Accompanying code can be found in the supplementary material.

Theorem 4. *Let i and j be two states of an ergodic Markov chain on $|S|$ states. Suppose that two chains are run simultaneously; one starting from i and the other from j. Let T_c be the first time in which the chains coalesce. Then $T_c \leq 2|S|T_{mix}\log(1/\delta)$ with probability at least $1-\delta$. Moreover, $\mathbb{E}[T_c] \leq 2|S|T_{mix}$.*

Proof. Let us start by sketching the proof idea. We break time into multiples of T_{mix}. We show that the probability that the chains do not coalesce after ℓ such time-multiples is at most $(1-\frac{1}{2|S|})^{\ell}$.

Denote by x_t and y_t be the states of the two chains at time t. For any t and two states i',j', denote by $E_{t,i',j'}$ be the event that $x_t = i'$ and $y_t = j'$. By the Markov property and by Lemma 3,

$$\Pr[x_{t+T_{\text{mix}}} = y_{t+T_{\text{mix}}} \mid E_{t,i',j'}] \geq \frac{1}{2|S|} .$$

Using the Markov property again, for any $\ell \geq 1$:

$$\begin{aligned}
&\Pr\left[T_c > \ell \cdot T_{\text{mix}} \mid T_c > (\ell-1)T_{\text{mix}}\right] \\
&\leq \Pr[x_{\ell \cdot T_{\text{mix}}} \neq y_{\ell \cdot T_{\text{mix}}} \mid T_c > (\ell-1)T_{\text{mix}}] \\
&= \sum_{i' \neq j'} \Pr\Big[x_{\ell \cdot T_{\text{mix}}} \neq y_{\ell \cdot T_{\text{mix}}} \mid T_c > (\ell-1)T_{\text{mix}}, \\
&\qquad\qquad E_{(\ell-1)T_{\text{mix}},i',j'}\Big] \cdot \Pr[E_{(\ell-1)T_{\text{mix}},i',j'}] \\
&= \sum_{i' \neq j'} \Pr[x_{\ell \cdot T_{\text{mix}}} \neq y_{\ell \cdot T_{\text{mix}}} \mid E_{(\ell-1)T_{\text{mix}},i',j'}] \\
&\qquad\qquad \cdot \Pr[E_{(\ell-1)T_{\text{mix}},i',j'}] \\
&\leq \sum_{i' \neq j'} \left(1 - \frac{1}{2|S|}\right) \cdot \Pr[E_{(\ell-1)T_{\text{mix}},i',j'}] \leq 1 - \frac{1}{2|S|} .
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Pr[T_c > \ell \cdot T_{\text{mix}}] \\
&\quad = \Pr[T_c > \ell \cdot T_{\text{mix}} \mid T_c > (\ell-1)T_{\text{mix}}] \\
&\qquad \cdot \Pr[T_c > (\ell-1)T_{\text{mix}}] \\
&\quad \leq \left(1 - \frac{1}{2|S|}\right) \Pr[T_c > (\ell-1)T_{\text{mix}}] ,
\end{aligned}$$

and inductively $\Pr[T_c > \ell \cdot T_{\text{mix}}] \leq \left(1-\frac{1}{2|S|}\right)^{\ell}$. The high probability bound immediately implies a bound on the expected coalescence time as follows:

$$\begin{aligned}
\mathbb{E}[T_c] &= \sum_{t=0}^{\infty} \Pr[T_c > t] \leq T_{\text{mix}}(1 + \sum_{\ell=1}^{\infty} \Pr[T_c > \ell T_{\text{mix}}]) \\
&\leq T_{\text{mix}} + T_{\text{mix}} \sum_{\ell=1}^{\infty} \left(1 - \frac{1}{2|S|}\right)^{\ell} = 2|S|T_{\text{mix}} .\square
\end{aligned}$$

We finish this subsection by showing that the upper bound in Theorem 4 is tight.

Theorem 5. *There exists an ergodic Markov chain on $|S|$ states and two states i,j such that the coalescence time T_c of two chains running simultaneously, one starting from i and the other from j, satisfies $\mathbb{E}[T_c] \geq \frac{1}{6}T_{mix} \cdot |S|$.*

Proof. Let $\varepsilon \in (0,1)$ and consider a Markov chain with $|S|$ states that for each state s, stays at s with probability $1-\varepsilon$, and with probability of ε choose the next state uniformly at random. Then, $\Pr[s' \mid s] = (1-\varepsilon)\mathbf{1}_{s=s'} + \frac{\varepsilon}{|S|}$. It is easy to see that the stationary distribution of this chain is uniform. This means that, starting the chain at state s_0, with probability ε the distribution at any time $t \geq 1$ is uniform. Therefore, $\Pr[s_t \mid s_0] = (1-\varepsilon)^t \mathbf{1}_{s_t=s_0} + (1-(1-\varepsilon)^t)\frac{1}{|S|}$.

Denote $\bar{u}$ the uniform distribution. We get that

$$\begin{aligned}
\text{TV}[\Pr_t[\,\cdot \mid s_0], \bar{u}] &= \frac{1}{2}\sum_s \left|\Pr[s_t = s \mid s_0] - \frac{1}{|S|}\right| \\
&= \frac{1}{2}\sum_s (1-\varepsilon)^t \left|\mathbf{1}_{s=s_0} - \frac{1}{|S|}\right| \\
&= \frac{|S|-1}{|S|} \cdot (1-\varepsilon)^t \leq e^{-\varepsilon t} .
\end{aligned}$$

This implies that $T_{\text{mix}} \leq \frac{3}{\varepsilon}$.

Next, notice that for coalescence to happen, one of the states i or j must transition to a state held by the other chain, which happens with probability at most $\frac{2\varepsilon}{|S|}$ via a union bound. Thus, in expectation, the time it takes for them to coalesce is $\mathbb{E}[T_c] \geq |S|/2\varepsilon \geq \frac{1}{6}|S|T_{\text{mix}}$. $\square$

2.2 COALESCENCE FROM $|S|$ STATES

In the supplementary material, we provide an *alternative, simple proof of the coalescence-time of the CFTP procedure*. The main ingredient of the proof is a generalization of the argument for bounding the coalescence-time of two chains to that of $|S|$ chains. The following theorem formalizes this.

Theorem 6. *Let μ be the stationary distribution of an ergodic Markov chain with $|S|$ states. We run $|S|$ simulations of the chain each starting at a different state. When two or more simulations coalesce, we merge them into a single simulation. With probability at least $1-\delta$, all $|S|$ chains are merged after at most $512|S|T_{mix}\log(1/\delta)$ iterations.*

2.3 ESTIMATING THE DIFFERENCE IN AVERAGE REWARD OF TWO POLICIES

Denote the difference in average reward between two policies by $\Delta\rho(\pi, \pi') \coloneqq \rho(\pi') - \rho(\pi)$. As seen in Section 2, we can sample a state from the stationary distribution of π and thereby get an unbiased estimate of $\rho(\pi)$ and similarly for π'. The difference between these estimates is an unbiased estimate of $\Delta\rho(\pi, \pi')$. However, we can also get an unbiased estimate of $\Delta\rho(\pi, \pi')$ by sampling the stationary distribution of only one of π and π', as we will now show. This property is useful when the sampling mechanism from one of the policies is restricted by real world constraints, e.g., in apprenticeship learning (see Section 3.3 for a concrete example). Our result builds on the following fundamental lemma regarding the average reward criteria (see, for example, Even-Dar et al., 2009, Lemma 5):

Lemma 7. $\forall \pi, \pi' \in \Pi : \Delta\rho(\pi, \pi') \coloneqq \rho(\pi') - \rho(\pi) = \mathbb{E}_{s\sim\mu(\pi')} \left\{ Q^{\pi}(s, \pi'(s)) - Q^{\pi}(s, \pi(s)) \right\}$.

Recall that Q^{π} is defined as in Eq. (1). Lemma 7 suggests a mechanism to estimate $\Delta\rho(\pi, \pi')$ using Theorem 4. We first sample a state, s_0 from the stationary distribution of π'. Then, we initiate two trajectories from s_0, the first trajectory follows π from s_0 and the second trajectory takes the first action (at s_0) according to π' and follows π thereafter. We accumulate the reward achieved by each trajectory until they coalesce. The difference between these sums makes an unbiased estimate of $\Delta\rho(\pi, \pi')$.

3 APPRENTICESHIP LEARNING

Consider learning in an MDP for which the reward function is not given explicitly, but we can observe an expert demonstrating the task that we want to learn. We think of the expert as trying to maximize the average reward function that is expressible as a linear combination of known features. This is the Apprenticeship Learning (AL) problem (Abbeel & Ng, 2004). We focus on extending the Multiplicative Weights Apprenticeship Learning (MWAL) algorithm (Syed & Schapire, 2008) that was developed for the discounted reward to the average-reward criteria. Our ideas may apply to other AL algorithms as well.

3.1 BACKGROUND

In AL, we are given an MDP dynamics M that is comprised of known states S, actions A, and transition matrices $(P^a)_{a\in A}$, yet the reward function is unknown. We further assume the existence of an *expert policy*, denoted by π^E, such that we are able to observe its execution in M. Following Syed & Schapire (2008), our goal is to find a policy π such that $\rho(\pi) \geq \rho(\pi^E) - \varepsilon$, **for any reward function**. To simplify the learning process, we follow Syed & Schapire (2008) in representing each state s by a low-dimensional vector of features $\phi(s) \in [0, 1]^k$. We consider reward functions that are linear in these features; i.e., $r(s) = w \cdot \phi(s)$, for some $w \in \Delta^k$ where Δ^k is the $(k-1)$-dimensional probability simplex. For compatibility with previous work, we decided to follow Syed & Schapire (2008) in assuming that the reward is in the probability simplex – in other AL papers (e.g. Abbeel & Ng, 2004; Zahavy et al., 2020), the L2 ball was considered instead. Having the set as the simplex, combined with the use of the Hedge algorithm (see below), allowed Syed & Schapire (2008) to improve the complexity of the algorithm to depend logarithmically on the dimension of the features rather than polynomially as in Abbeel & Ng (2004).

With this feature representation, the average reward of a policy π may be written as $\rho(\pi) = w \cdot \Phi(\pi)$ where $\Phi(\pi)$ is the expected accumulated feature vector associated with π, defined as $\Phi(\pi) = \lim_{N\to\infty} \mathbb{E}_{\pi} \sum_{t=0}^{N-1} \phi(x_t)/N$. Notice that similar to the average reward, this limit is not a function of the initial state when the MDP is ergodic.

We also require the notion of a *mixed policy* which is a distribution over stationary deterministic policies. Our algorithms return a mixed policy, and our analysis is with respect to this mixed policy. We denote by Ψ the set of all mixed policies in M and by Π the set of all deterministic stationary policies in M. For a mixed policy $\psi \in \Psi$ and a deterministic policy $\pi \in \Pi$, we denote by $\psi(\pi)$ the probability assigned by ψ to π. A mixed policy ψ is executed by randomly selecting the policy $\pi \in \Pi$ at time 0 with probability $\psi(\pi)$, and following π after that. The definition of Φ extends naturally to mixed policies. In terms of average reward, mixed policies cannot achieve higher average reward than deterministic policies.

We think of AL as a zero-sum game between two players, defined by the following $k \times |\Pi|$ matrix:

$$G(i, \pi) = \Phi(\pi)[i] - \Phi(\pi^E)[i], \tag{2}$$

where $\Phi(\pi)[i]$ is the i-th component of feature expectations vector $\Phi(\pi)$ for the deterministic policy π. Both players play a mixed policy. The row player selects a vector $w \in \Delta^k$, which is a probability distribution over the k features, and the column player chooses a policy $\psi \in \Psi$. Then, the value of the game is defined as

$$\begin{aligned} v^{\star} &= \max_{\psi\in\Psi} \min_{w\in\Delta^k} \left[w \cdot \Phi(\psi) - w \cdot \Phi(\pi^E) \right] \\ &= \max_{\psi\in\Psi} \min_{w\in\Delta^k} w^{\top} G \psi \,. \end{aligned} \tag{3}$$

In Sections 3.2 and 3.3 we propose and analyze two algorithms for apprenticeship learning with the average re-

ward criteria, based on the MWAL algorithm (Syed & Schapire, 2008). Specifically, these algorithms learn a mixed policy $\bar{\psi}$ that approximately achieves the max-min value v^* (defined in Eq. (3)) against any $w \in \Delta^k$. As in previous work, we assume that the dynamics are known, yet we have access to the expert policy via an expert generative model E. Given a state s, the expert generative model E provides a sample from $\pi^E(s)$. In Section 3.2 we propose an algorithm that uses Coupling From The Past (CFTP) to estimate the feature expectations of the expert Φ^E. In Section 3.3 we propose an algorithm that queries the expert, at each step, for two trajectories to compute an unbiased estimate $\tilde{g}_t$ of the column of the game matrix (Eq. (2)) corresponding to $\pi^{(t)}$ based on Lemma 7. Both algorithms update the strategies of the min (row) and max (column) players using standard RL methods as follows.

(i) Given a min player strategy w, find $\arg\max_{\pi\in\Pi} G(w,\pi) = \sum_{i=1}^{k} w(i)G(i,\pi)$. This step is equivalent to finding the optimal policy in an MDP with a known reward and can be solved for example with Value Iteration or Policy Iteration. **(ii)** Given a max player strategy π, the min player maintains a probability vector $w \in \Delta^k$ giving a weight to each row (feature). To update the weights, we estimate $G(i,\pi)$ for each $i \in \{1,\ldots,k\}$ and the policy π of the max player. The algorithms in Sections 3.2 and 3.3 differ in the way they estimate these $G(i,\pi)$'s. In Section 3.2, we estimate the features expectations of the expert once before we start. Then, in each iteration, we evaluate the feature expectations of π by solving a system of linear equations using the known dynamics. Then we estimate $G(i,\pi)$ by subtracting the features expectations of π from the estimates of the features expectations of π^E. We note that we can also handle the case where this step (and the PI step) is inaccurate. In this case, the representation error would appear in the bounds of the theorems below. Importantly, the complexity of both steps in our algorithms grows with the size of the MDP, but not with the size of the game matrix. In Section 3.3 we take a different approach and estimate the difference directly by generating two trajectories of the expert from two particular states.

3.2 ESTIMATING THE FEATURE EXPECTATIONS OF THE EXPERT

The algorithm of this section uses CFTP to obtain samples from the expert's stationary distribution and uses them to estimate $\widetilde{\Phi}^E$–the expert's feature expectations. See Algorithm 2, line 3. Obtaining each of these samples requires $\Theta(|S|T^{\pi^E}_{\text{mix}})$ calls to the generative model (Theorem 6), totaling at $O(|S|T^{\pi^E}_{\text{mix}} \cdot m)$ calls overall. The number of samples m is taken to be large enough so that the estimate $\widetilde{\Phi}^E$ is ε-accurate. The following theorem describes the sample complexity of Algorithm 2.

Theorem 8. *Assume we run Algorithm 2 for* $T = \frac{144}{\varepsilon^2}\log k$ *iterations, using* $m = \frac{18}{\varepsilon^2}\log(2k/\delta)$ *samples from* $\mu(\pi^E)$. *Let* $\bar{\psi}$ *be the mixed policy returned by the algorithm. Let* $v^\star$ *be the game value as in Eq.* (3). *Then, we have that* $\rho(\bar{\psi}) - \rho(\pi^E) \geq v^\star - \varepsilon$ *with probability at least* $1-\delta$, *where* ρ *is any average reward of the form* $r(s) = w \cdot \phi(s)$ *where* $w \in \Delta_k$.

Note that Theorem 8 is similar to Theorem 2 of Syed & Schapire (2008). The main difference is that our result applies to the average-reward criteria, and we evaluate the expert using samples of its stationary distribution instead of using trajectories of finite length (which are biased). This simplifies the analysis and gives tighter bounds. As a comparison, the iteration complexity of MWAL is $T = O(\frac{\log(k)}{\varepsilon^2(1-\gamma)^2})$, which is also logarithmic in k and linear in $1/\varepsilon^2$ but depends in the discount factor. In the discounted case, a complete trajectory is required in order to have a single unbiased estimate of the feature expectations. In the average reward case, on the other hand, a single sample from the stationary distribution suffices to create an unbiased estimate of the feature expectations, and therefore the iteration complexity does not depend on the trajectory length. More details can be found in the proof (Section C.2).

Algorithm 2 MWAL for average reward criteria

1: **Given**: MDP dynamics M; generative model of the expert policy E; feature dimension k; number of iterations T; m the number of samples from $\mu(\pi^E)$.
2: Let $\beta = \sqrt{\frac{\log k}{T}}$ (learning rate)
3: **Sampling**: Use the CFTP protocol with E and M, to obtain m samples $\{\phi(s_i)\}_{i=1}^m$ s.t. s_i are i.i.d random variables and $s_i \sim \mu(\pi^E)$. Let $\widetilde{\Phi}^E = \frac{1}{m}\sum_{i=1}^m \phi(s_i)$.
4: Initialize $W^{(1)}(i) = 1$, for $i = 1,\ldots,k$.
5: **for** $t = 1,\ldots,T$ **do**
6: Set $w^{(t)}(i) = \frac{W^{(t)}(i)}{\sum_{i=1}^k W^{(t)}(i)}$, for $i = 1,\ldots,k$.
7: Compute an optimal policy $\pi^{(t)}$ for M with respect to reward function $r^{(t)}(s) = w^{(t)} \cdot \phi(s)$.
8: **for** $i = 1,\ldots,k$ **do**
9: Set $\tilde{g}_t(i) = \left(\Phi(\pi^{(t)})[i] - \widetilde{\Phi}^E[i] + 1\right)/2$.
10: $W^{(t+1)}(i) = W^{(t)}(i) \cdot \exp\left(-\beta\tilde{g}_t(i)\right)$.
11: **end for**
12: **end for**
13: Post-processing: Return the mixed policy $\bar{\psi}$ that assigns probability $\frac{1}{T}$ to $\pi^{(t)}$, for all $t \in \{1,\ldots,T\}$.

Remark. Recall that the expert policy may be stochastic. At first glance, it may be tempting to try to estimate

the expert policy directly. However, note that $\phi(\pi^E)$ is an expectation over the expert's stationary distribution. Even if we do manage to estimate the expert's policy to ε-accuracy in each state, the small error in the estimated policy may entail a significant error in its stationary distribution. In fact, this error might be as large as $\Omega(T_{\text{mix}}^{\pi^E}\varepsilon)$. In particular, there is no sample size which is oblivious to $T_{\text{mix}}^{\pi^E}$ and guarantees an ε bounded error.

3.3 ESTIMATING THE GAME MATRIX DIRECTLY

In the previous section, we introduced an algorithm that uses the CFTP protocol to sample the expert's stationary distribution without any knowledge of the corresponding Markov chain's mixing time. However, this mechanism required to query the expert for a long trajectory starting from **every** state to obtain a single sample from the stationary distribution. This may be tedious for the expert in practice, in particular in domains with large state spaces.

To relax this requirement, Algorithm 3 uses a different sampling mechanism that is **not** estimating $\Phi(\pi^E)$ at the beginning of the algorithm. Instead, Algorithm 3 queries the expert for two trajectories **at each step** to generate an unbiased estimate g_t of a particular column of the game matrix. To obtain the estimate g_t (Algorithm 3, line 7), we use the sampling mechanism developed in Section 2 for evaluating the difference in the average reward of two policies $\Delta\rho(\pi, \pi')$ (Lemma 7). Specifically, we start by sampling a state s_0 from the stationary distribution of $\pi^{(t)}$. Since $\pi^{(t)}$ and the dynamics are known, the stationary distribution of $\pi^{(t)}$ can be computed by solving a system of linear equations. Next, we initiate two trajectories from s_0; the first trajectory follows the expert policy from s_0 and the second takes the first action (at s_0) according to $\pi^{(t)}$ and follows the expert after that. We accumulate the features $\phi(s)$ along the trajectories until they coalesce. The difference between these sums gives the unbiased estimate g_t of $G(\cdot, \pi^{(t)}) = \Phi(\pi^{(t)}) - \Phi(\pi^E)$ (the column of the game matrix G corresponding to $\pi^{(t)}$).

Theorem 9 below presents the sample complexity of this approach as a function of b: a parameter that bounds the estimates g_t with high probability. Concretely, we assume that for any $\ell > 0$, $\Pr[\|g_t\|_\infty > \ell \cdot b] \leq e^{-\ell}$. In view of Theorem 4, b is always upper bounded by $|S|T_{\text{mix}}^{\pi^E}$. But, we believe that it can be much smaller in practice and that there exists many cases where b can be known a-priori due to the structure of the reward function. For example, consider an MDP with a p–sparse reward function, i.e., the reward (and the feature vector in these states) is not zero in at most p states. While it might take a long time for two trajectories to coalesce, the difference in the reward between these trajectories can be upper bounded using the sparsity degree p of the reward. Concretely, consider an MDP with the following dynamics: $P(s_i, s_{i+1}) = 1, \forall i \in [1, ..n-1], P(s_n, s_n) = 1-\varepsilon, P(s_n, s_1 = \varepsilon)$, and a p–sparse reward function. For $\varepsilon \ll 1/n$, the trajectories will coalesce at s_n (with high probability), and we have that for any $\ell > 0$, $\Pr[\|g_t\|_\infty > \ell \cdot p] \leq e^{-\ell}$.

Algorithm 3 MWAL with generative differences

1: **Given**: MDP dynamics M; generative model of the expert policy E; feature dimension k; number of iterations T; parameter δ; parameter b.
2: Let $\beta = \sqrt{\frac{\log k}{T}}$, $B = b\log(2Tk/\delta)$
3: Initialize $W^{(1)}(i) = 1$, for $i = 1, \ldots, k$.
4: **for** $t = 1, \ldots, T$ **do**
5: Set $w^{(t)}(i) = \frac{W^{(t)}(i)}{\sum_{i=1}^{k} W^{(t)}(i)}$, for $i = 1, \ldots, k$.
6: Compute an optimal policy $\pi^{(t)}$ for M w.r.t reward function $r^{(t)}(s) = w^{(t)} \cdot \phi(s)$.
7: **Sample** g_t s.t. $\mathbb{E}[g_t(i)] = G(i, \pi^{(t)}), \forall i = 1, \ldots, k$
8: **for** $i = 1, \ldots, k$ **do**
9: Set $\tilde{g}_t(i) = (g_t(i) + B)/2B$.
10: $W^{(t+1)}(i) = W^{(t)}(i) \cdot \exp\left(-\beta\tilde{g}_t(i)\right)$.
11: **end for**
12: **end for**
13: Post-processing: Return the mixed policy $\bar{\psi}$ that assigns probability $\frac{1}{T}$ to $\pi^{(t)}$, for all $t \in \{1, \ldots, T\}$.

Theorem 9. *Assume we run Algorithm 3 for T iterations, and there exists a parameter b, such that for any ℓ, $\Pr(\|g_t\|_\infty \geq \ell \cdot b) \leq e^{-\ell}$. Let $\bar{\psi}$ be the mixed policy returned by the algorithm. Let $v^\star$ be the game value as in Eq.* (3)*. Then, there exists a constant c such that for $T \geq cB\log^2 B$ where $B = \frac{b^2}{\varepsilon^2}\log^3 k \log^2(1/\delta)$, we have that $\rho(\bar{\psi}) - \rho(\pi^E) \geq v^\star - \varepsilon$ with probability at least $1-\delta$, where ρ is the average of any reward of the form $r(s) = w \cdot \phi(s)$ where $w \in \Delta_k$.*

The key difference from the proof of Theorem 8 is in refining the original analysis to incorporate the variance of the estimates g_t into the algorithm's iteration complexity. The proof is found in Section C.3.

4 POLICY GRADIENT

Consider the problem of finding the best policy in an MDP from the set of all policies that are parameterized by a vector θ. Sutton et al. (2000) proposed a variant of Policy Iteration that uses the unbiased estimate of the policy gradient and guaranteed that it converges to a locally optimal policy. We now describe a sampling mechanism that achieves such an unbiased sample, resulting in a much simpler algorithm than the biased policy gradients algorithm of (Baxter & Bartlett, 2001; Marbach &

Tsitsiklis, 2001).

The Policy Gradient Theorem (Sutton et al., 2000), states that for the average-reward criteria,

$$\frac{\partial \rho}{\partial \theta} = \mathbb{E}_{s\sim\mu(\pi)}\mathbb{E}_{a\sim\pi(s)}\frac{\partial \log \pi(s,a)}{\partial \theta}Q^{\pi}(s,a)\ ,$$

where $Q^{\pi}(s,a) = \sum_{t=1}^{\infty}\mathbb{E}(r_t - \rho(\pi)|s_0 = s, a_0 = a, \pi)$. We produce an unbiased estimate of the policy gradient similarly to evaluating the reward difference between policies described in Section 2. Specifically, we do the following: (1) use the CFTP method to get unbiased sample $s \sim \mu(\pi)$ from the stationary distribution of π; (2) sample $a' \sim \pi(s)$; (3) initiate two trajectories from s. The first trajectory starts by taking action a (the action we want to estimate the Q function at), and the second starts by taking a'. Even if these actions are the same, they would not necessarily lead to the same state as the environment is stochastic. After the first action is taken, both trajectories follow π until coalescence. The difference of cumulative rewards between the two trajectories forms an unbiased estimate of the Q-value. To construct the estimate of the gradient, we multiply the Q-value estimate by the derivative of $\log \pi(s,a)$ at θ.

5 DISCUSSION

We derived and analyzed reinforcement learning algorithms for average reward criteria. Existing algorithms explicitly require an upper bound on the mixing time. In contrast, we leveraged the CFTP protocol and derived sampling algorithms that **do not require such an upper bound**. For these algorithms, we provided theoretical bounds on their sample-complexity and running time. Finally, we offered an alternative, simpler proof for the correctness of CFTP. As CFTP is a twenty-year-old protocol, we hope that our proof will make it more accessible to the RL community.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Arrow, K. J., Karlin, S., Scarf, H. E., et al. *Studies in the mathematical theory of inventory and production*. Stanford University Press, 1958.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., and Bertsekas, D. P. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

Blackwell, D. Discrete dynamic programming. *The Annals of Mathematical Statistics*, 1962.

Brafman, R. I. and Tennenholtz, M. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

Chen, Y., Li, L., and Wang, M. Scalable bilinear learning using state and action features. In *International Conference on Machine Learning*, pp. 833–842, 2018.

Even-Dar, E., Kakade, S. M., and Mansour, Y. Online markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Gosavi, A. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. Springer US, 2014.

Häggström, O. *Finite Markov chains and algorithmic applications*, volume 52. Cambridge University Press, 2002.

Howard, R. A. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

Kearns, M. J., Mansour, Y., and Ng, A. Y. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems*, pp. 1001–1007, 2000.

Kelly, F. P. Networks of queues with customers of different types. *Journal of applied probability*, 12(3):542–554, 1975.

Levin, D., Peres, Y., and Wilmer, E. *Markov Chains and Mixing Times*. American Mathematical Society, 2017.

Mahadevan, S. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning*, 22(1-3):159–195, 1996.

Marbach, P. and Tsitsiklis, J. N. Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control*, 46(2):191–209, 2001.

Propp, J. G. and Wilson, D. B. Exact sampling with coupled markov chains and applications to statistical

mechanics. *Random Structures & Algorithms*, 9(1-2): 223–252, 1996.

Propp, J. G. and Wilson, D. B. 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), 1998.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1984.

Sutton, R. S. and Barto, A. G. *Reinforcement learning - an introduction*. MIT Press, 1998.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Syed, U. and Schapire, R. E. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, 2008.

Wang, M. Primal-dual π learning: Sample complexity and sublinear run time for ergodic markov decision problems. *arXiv:1710.06100*, 2017.

White, D. J. Dynamic programming, markov chains, and the method of successive approximations. *Journal of Mathematical Analysis and Applications*, 6(3), 1963.

Wolfer, G. and Kontorovich, A. Estimating the mixing time of ergodic markov chains. *arXiv:1902.01224*, 2019.

Zahavy, T., Cohen, A., Kaplan, H., and Mansour, Y. Apprenticeship learning via frank-wolfe. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.