Bridging Reinforcement Learning Theory and Practice with the Effective Horizon

Cassidy Laidlaw **Stuart Russell** **Anca Dragan**
University of California, Berkeley
{cassidy_laidlaw,russell,anca}@cs.berkeley.edu

Abstract

Deep reinforcement learning (RL) works impressively in some environments and fails catastrophically in others. Ideally, RL theory should be able to provide an understanding of why this is, i.e. bounds predictive of practical performance. Unfortunately, current theory does not quite have this ability. We compare standard deep RL algorithms to prior sample complexity bounds by introducing a new dataset, BRIDGE. It consists of 155 deterministic MDPs from common deep RL benchmarks, along with their corresponding tabular representations, which enables us to exactly compute instance-dependent bounds. We choose to focus on deterministic environments because they share many interesting properties of stochastic environments, but are easier to analyze. Using BRIDGE, we find that prior bounds do not correlate well with when deep RL succeeds vs. fails, but discover a surprising property that does. When actions with the highest Q-values under the *random* policy also have the highest Q-values under the *optimal* policy (i.e. when it is optimal to be greedy on the random policy's Q function), deep RL tends to succeed; when they don't, deep RL tends to fail. We generalize this property into a new complexity measure of an MDP that we call the *effective horizon*, which roughly corresponds to how many steps of lookahead search would be needed in that MDP in order to identify the next optimal action, when leaf nodes are evaluated with random rollouts. Using BRIDGE, we show that the effective horizon-based bounds are more closely reflective of the empirical performance of PPO and DQN than prior sample complexity bounds across four metrics. We also find that, unlike existing bounds, the effective horizon can predict the effects of using reward shaping or a pre-trained exploration policy. Our code and data are available at `https://github.com/cassidylaidlaw/effective-horizon`.

1 Introduction

Deep reinforcement learning (RL) has produced impressive results in robotics [1], strategic games [2], and control [3]. However, the same deep RL algorithms that achieve superhuman performance in some environments completely fail to learn in others. Sometimes, using techniques like reward shaping or pre-training help RL, and in other cases they don't. Our goal is to provide a theoretical understanding of why this is—a theoretical analysis that is *predictive* of practical RL performance.

Unfortunately, there is a large gap between the current theory and practice of RL. Despite RL theorists often focusing on algorithms using strategic exploration (e.g., UCB exploration bonuses; Azar et al. [4], Jin et al. [5]), the most commonly-used deep RL algorithms, which explore randomly, resist such analysis. In fact, theory suggests that RL with random exploration is exponentially hard in the worst case [6], but this is not predictive of practical performance. Some theoretical research has explored instance-dependent bounds, identifying properties of MDPs when random exploration should perform better than this worst case [7, 8]. However, it is not clear whether these properties correlate with when RL algorithms work vs. fail—and our results will reveal that they tend not to.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

If the current theory literature cannot explain the empirical performance of deep RL, what can? Ideally, a theory of RL should provably show why deep RL succeeds while using random exploration. It should also be able to predict which environments are harder or easier to solve empirically. Finally, it should be able to explain when and why tools like reward shaping or initializing with a pre-trained policy help make RL perform better.

We present a new theoretical complexity measure for MDPs called the *effective horizon* that satisfies all of the above criteria. Intuitively, the effective horizon measures approximately how far ahead an algorithm must exhaustively plan in an environment before evaluating leaf nodes with random rollouts.

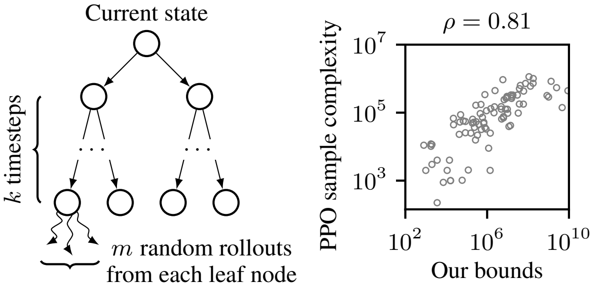


Figure 1: We introduce the effective horizon, a property of MDPs that controls how difficult RL is. Our analysis is motivated by Greedy Over Random Policy (GORP), a simple Monte Carlo planning algorithm (left) that exhaustively explores action sequences of length k and then uses m random rollouts to evaluate each leaf node. The effective horizon combines both k and m into a single measure. We prove sample complexity bounds based on the effective horizon that correlate closely with the real performance of PPO, a deep RL algorithm, on our BRIDGE dataset of 155 deterministic MDPs (right).

In order to assess previous bounds and eventually arrive at such a property, we start by creating a new dataset, BRIDGE, of deterministic MDPs from common deep RL benchmarks. A major difficulty with evaluating instance-dependent bounds is that they can't be calculated without tabular representations, so prior work work has typically relied on small toy environments for justification. To get a more realistic picture, we choose 155 MDPs across different benchmarks and compute their tabular representations—some with over 100 million states which must be exhaustively explored and stored. This is a massive engineeering challenge, but it enables connecting theoretical and empirical results at an unprecedented scale. We focus on deterministic MDPs in BRIDGE and in this paper because they are simpler to analyze but still have many of the interesting properties of stochastic MDPs, like reward sparsity and credit assignment challenges. Many deep RL benchmarks are (nearly) deterministic, so we believe our analysis is highly relevent to practical RL.

Our journey to the effective horizon began with identifying a surprising property that holds in many of the environments in BRIDGE: one can learn to act *optimally* by acting *randomly*. More specifically, actions with the highest Q-values under the uniformly *random* policy *also* have the highest Q-values under the *optimal* policy. The random policy is about as far as one can get from the optimal policy, so this property may seem unlikely to hold. However, about two-thirds of the environments in BRIDGE satisfy the property. This proportion rises to four-fifths among environments that PPO [9], a popular deep RL algorithm, can solve efficiently (Table 1). Conversely, when this property does not hold, PPO is more likely to fail than succeed—and when it does succeed, so does simply applying a few steps of lookahead on the Q-function of the random policy (Figure 6). We found it remarkable that, at least in the environments in BRIDGE, modern algorithms seem to boil down to not much more than acting greedily on the random policy Q-values.

The property that it is optimal to act greedily with respect to the random policy's Q-function has important implications for RL theory and practice. Practically, it suggests that very simple algorithms designed to estimate the random policy's Q-function could efficiently find an optimal policy. We introduce such an algorithm, Greedy Over Random Policy (GORP), which also works in the case where one may need to apply a few steps of value iteration to the random policy's Q-function before acting greedily. Empirically, GORP finds an optimal policy in fewer timesteps than DQN (another deep RL algorithm) in more than half the environments in BRIDGE. Theoretically, it is simple to analyze GORP, which consists almost entirely of estimating the random policy's Q-function via a sample average over i.i.d. random rollouts. Since GORP works well empirically and can be easily understood theoretically, we thoroughly analyze it in the hopes of finding sample complexity bounds that can explain the performance of deep RL.

Our analysis of Greedy Over Random Policy leads to a single metric, the effective horizon, that measures the complexity of model-free RL in an MDP. As shown in Figure 1, GORP is an adaptation of a Monte Carlo planning algorithm to the reinforcement learning setup (where the transitions are unknown): it mimics exhaustively planning ahead k steps and then sampling m random rollouts from each leaf node. The effective horizon H combines the depth k and number of rollouts m. We call it the *effective* horizon because worst-case sample complexity bounds for random exploration are exponential in the horizon T, while we prove sample complexity bounds exponential only in the effective horizon H. For most BRIDGE environments, $H \ll T$, explaining the efficiency of RL in these MDPs.

In the environments in BRIDGE, we find that the effective horizon-based sample complexity bounds satisfy all our desiderata above for a theory of RL. They are more predictive of the empirical sample complexities of PPO and DQN than several other bounds from the literature across four metrics, including measures of correlation, tightness, and accuracy (Table 2). Furthermore, the effective horizon can predict the effects of both reward shaping (Table 3a) and initializing using a pre-trained policy learned from human data or transferred from a similar environment (Table 3b). In contrast, none of the existing bounds we compare to depend on both the reward function and initial policy; thus, they are unable to explain why reward shaping, human data, and transfer learning can help RL. Although our results focus on deterministic MDPs, we plan to extend our work to stochastic environments in the future and already have some promising results in that direction.

2 Preliminaries

We begin by presenting the reinforcement learning (RL) setting we consider. An RL algorithm acts in a deterministic, tabular, episodic Markov decision process (MDP) with finite horizon. The MDP comprises a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a horizon $T \in \mathbb{N}$ and optional discount factor $\gamma \in [0, 1]$, a start state s_1, transition function $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Throughout the paper we use $\gamma = 1$ but all our theory applies equally when $\gamma < 1$.

An RL agent interacts with the MDP for a number of episodes, starting at a fixed start state s_1. At each step $t \in [T]$ of an episode (using the notation $[n] = \{1, \ldots, n\}$), the agent observes the state s_t, picks an action a_t, receives reward $R(s_t, a_t)$, and transitions to the next state $s_{t+1} = f(s_t, a_t)$. A policy π is a set of functions $\pi_1, \ldots, \pi_t : \mathcal{S} \to \Delta(\mathcal{A})$, which defines for each state and timestep a distribution $\pi_t(a \mid s)$ over actions. If a policy is deterministic at some state, then with slight abuse of notation we denote $a = \pi_t(s)$ to be the action taken by π_t in state s.

We denote a policy's Q-function $Q_t^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and value function $V_t^\pi : \mathcal{S} \to \mathbb{R}$ for each $t \in [T]$. In this paper, we also use a Q-function which is generalized to *sequences* of actions. We use the shorthand $a_{t:t+k}$ to denote the sequence $a_t, \ldots, a_{t+k}$, and define the action-sequence Q-function as

$$Q_t^\pi(s_t, a_{t:t+k}) = \mathbb{E}_\pi \left[\sum_{t'=t}^{T} \gamma^{t'-t} \, R(s_{t'}, a_{t'}) \mid s_t, a_{t:t+k} \right].$$

The objective of an RL algorithm is to find an optimal policy π^*, which maximizes $J(\pi) = V_1^\pi(s_1)$, the expected discounted sum of rewards over an episode, also known as the return of the policy π.

Generally, an RL algorithm can be run for any number of timesteps n (i.e., counting one episode as T timesteps), returning a policy π^n. We define the *sample complexity* N of an RL algorithm as the minimum number of timesteps needed such that the algorithm has at least a 50-50 chance of returning an optimal policy:

$$N = \min \left\{ n \in \mathbb{N} \mid \mathbb{P}\left(J(\pi^n) = J^*\right) \geq 1/2 \right\}.$$

Here, the probability is with respect to any randomness in the algorithm itself. One can estimate the sample complexity N empirically by running an algorithm several times, calculating the number of samples n needed to reach the optimal policy during each run, and then taking the median.

The following simple theorem gives upper and lower bounds for the worst-case sample complexity in a deterministic MDP, depending on A and T.

Theorem 2.1. *There is an RL algorithm which can solve any deterministic MDP with sample complexity $N \leq T\lceil A^T/2 \rceil$. Conversely, for any RL algorithm and any values of T and A, there must be some deterministic MDP for which its sample complexity $N \geq T(\lceil A^T/2 \rceil - 1)$.*

All proofs are deferred to Appendix A. In this case, the idea of the proof is quite simple, and will later be useful to motivate our idea of the effective horizon: in an MDP where exactly one sequence of actions leads to a reward, an RL algorithm may have to try almost every sequence of actions to find the optimal policy; there are A^T such sequences. As we develop sample complexity bounds based on the effective horizon in Section 5, we can compare them to the worst-case bounds in Theorem 2.1.

Why deterministic MDPs? We focus on deterministic (as opposed to stochastic) MDPs in this study for several reasons. First, analyzing deterministic MDPs avoids the need to consider generalization within RL algorithms. In common stochastic MDPs, one often needs neural-network based policies, whereas in a deterministic MDP one can simply learn a sequence of actions. Since neural network generalization is not well understood even in supervised learning, analyzing generalization in RL is an especially difficult task. Second, deterministic MDPs still display many of the interesting properties of stochastic MDPs. For instance, deterministic MDPs have worst case exponential sample complexity when using naive exploration; environments with dense rewards are easier to solve empirically than those with sparse rewards; credit assignment can be challenging; and there is a wide range of how tractable environments are for deep RL, even for environments with similar horizons, state spaces, and action spaces.

Finally, many common RL benchmark environments are deterministic or nearly-deterministic. For instance, the ALE Atari games used to evaluate DQN [3], Rainbow [10], and MuZero [11] are all deterministic after the first state, which is selected randomly from one of only 30 start states. The widely used DeepMind Control Suite [12] is based on the MuJoCo simulator [13], which is also deterministic given the initial state (some of the environments do use a randomized start state). MiniGrid environments [14], which are commonly used for evaluating exploration [15], environment design [16], and language understanding [17], are also deterministic after the initial state. Thus, our investigation of deterministic environments is highly relevant to common deep RL practice.

3 Related Work

Before delving into our contributions, we briefly summarize existing work in theoretical RL and prior sample complexity bounds. Our novel bounds are contrasted with existing ones in Sections 5.1 and 6; for a detailed comparison with full definitions and proofs, please see Appendix D.

Recent RL theoretical results largely focus on strategic exploration using techniques like UCB exploration bonuses [18, 4, 19, 5, 20, 21, 22]. Such bounds suggest RL is tractable for smaller or low-dimensional state spaces. In deterministic MDPs, the UCB-based R-MAX algorithm [23, 18] has sample complexity bounded by SAT.

Some prior work has focused on sample complexity bounds for random exploration. Liu and Brunskill [7] give bounds based on the covering length L of an MDP, which is the number of episodes needed to visit all state-action pairs at least once with probability at least $1/2$ while taking actions at random. This yields a sample complexity bound of TL for deterministic MDPs. Other work suggests that it may not be necessary to consider rewards all the way to the end of the episode to select an optimal action [24, 25, 8]. One can define a "effective planning window" of W timesteps ahead that must be considered, resulting in a sample complexity bound of $T^2 A^W$ for deterministic MDPs. Finally, Dann et al. [6] define a "myopic exploration gap" that controls the sample complexity of using ϵ-greedy exploration, a form of naive exploration. However, in Appendix D.4, we demonstrate why their bounds are impractical and often vacuous.

There have been a few prior attempts to bridge the RL theory-practice gap. bsuite [26], MDP playground [27], and SEGAR [28] are collections of environments that are designed to empirically evaluate deep RL algorithms across various axes of environment difficulty. However, they do not provide theoretical explanations for why environments with varying properties are actually easier or harder. Furthermore, their environments are artificially constructed to have understandable properties. In contrast, we aim to find the mathematical reasons that deep RL succeeds or fails in "in-the-wild" environments like Atari and Procgen. Conserva and Rauber [29] calculate two regret bounds and compare the bounds to the empirical performance of RL algorithms. However, they consider tabular RL algorithms in simple artificial environments with less than a thousand states, while we experiment with deep RL algorithms on real benchmark environments with tens of millions of states.

Our GORP algorithm and the effective horizon are inspired by rollout and Monte Carlo planning algorithms, which have a long history [30, 31, 32, 33, 24, 34, 35, 36]. These algorithms were used effectively in Backgammon [32], Go [37], and real-time strategy games [38] before the start of deep RL. GORP and related Monte Carlo rollout algorithms are sometimes referred to as "one-step" or "multi-step" lookahead. Bertsekas [39, 40] suggests that one-step lookahead, possibly after a few steps of value iteration, often leads to fast convergence to optimal policies because it is equivalent to a step of Newton's method for finding a fixed-point of the Bellman equation [41, 42]. Our analysis suggests the success of deep RL is due to similar properties. However, we go beyond previous work by introducing GORP, which approximates a step of policy iteration in model-free RL—a setting where on-line planning approaches are not applicable. Furthermore, unlike previous work in Monte-Carlo planning, we combine our theoretical contributions with extensive empirical analysis to verify that our assumptions hold in common environments.

4 The BRIDGE Dataset

In order to assess how well existing bounds predict practical performance, and gain insight about novel properties of MDPs that could be predictive, we constructed BRIDGE (Bridging the RL Interdisciplinary Divide with Grounded Environments), a dataset of 155 popular deep RL benchmark environments with full tabular representations. One might assume that the ability to calculate the instance-dependent bounds we just presented in Section 3 already exists; however, it turns out that for many real environments even the number of states S is unknown! This is because a significant engineering effort is required to analyze large-scale environments and calculate their properties.

Acting greedily with respect to $Q^{\pi^{\text{rand}}}$ is optimal?	PPO finds optimal policy in $\leq 5\text{M}$ timesteps?	
	Yes	No
Yes	**80 MDPs**	24 MDPs
No	15 MDPs	**36 MDPs**

Table 1: The distribution of the MDPs in our BRIDGE dataset according to two criteria: first, whether PPO empirically converges to an optimal policy in 5 million timesteps, and second, whether acting greedily with respect to the Q-function of the random policy is optimal. We find that a surprising number of environments satisfy the latter property, especially when only considering those where PPO succeeds.

In BRIDGE, we tackle this problem by computing tabular representations for all the environments using a program that exhaustively enumerates all states, calculating the reward and transition functions at every state-action pair. We do this for 67 Atari games from the Arcade Learning Enivornment [43], 55 levels from the Procgen Benchmark [44], and 33 gridworlds from MiniGrid [14] (Figure 5). The MDPs have state space sizes S ranging across 7 orders of magnitude from tens to tens of millions, 3 to 18 discrete actions, and horizons T ranging from 10 to 200, which are limited in some cases to avoid the state space becoming too large. See Appendix E for the full details of the BRIDGE dataset.

A surprisingly common property To motivate the effective horizon, which is introduced in the next section, we describe a property that we find holds in many of the MDPs in BRIDGE. Consider the random policy π^{rand}, which assigns equal probability to every action in every state, i.e., $\pi_t^{\text{rand}}(a \mid s) = 1/A$. We can use dynamic programming on a tabular MDP to calculate the random policy's Q-function $Q^{\pi^{\text{rand}}}$. We denote by $\Pi(Q^{\pi^{\text{rand}}})$ the set of policies which act greedily with respect to this Q-function; that is,

$$\Pi\left(Q^{\pi^{\text{rand}}}\right) = \left\{\pi \mid \forall s, t \quad \pi_t(s) \in \arg\max_{a \in \mathcal{A}} Q_t^{\pi^{\text{rand}}}(s, a)\right\}.$$

Perhaps surprisingly, we find that all the policies in $\Pi(Q^{\pi^{\text{rand}}})$ are *optimal* in about two-thirds of the MDPs in BRIDGE. This proportion is even higher when considering only the environments where PPO empirically succeeds in finding an optimal policy (Table 1). Thus, it seems that this property may be the key to what makes many of these environments tractable for deep RL.

5 The Effective Horizon

We now theoretically analyze why RL should be tractable in environments where, as we observe in BRIDGE, it is optimal to act greedily with respect to the random policy's Q-function. This leads to a more general measure of an environment's complexity for model-free RL: the effective horizon.

Our analysis centers around a simple algorithm, GORP (Greedy Over Random Policy), shown in Algorithm 1. GORP constructs an optimal policy iteratively; each iteration i aims to calculate an optimal policy π_i for timestep $t = i$. In the case where we set $k = 1$ and $\pi^{\text{expl}} = \pi^{\text{rand}}$, GORP can solve environments which have the property we observe in BRIDGE. It does this at each iteration i by simulating m random rollouts for each action from the state reached at timestep $t = i$. Then, it averages the m rollouts' returns to obtain a Monte Carlo estimate of $Q^{\pi^{\text{rand}}}$ for each action. Finally, it greedily picks the action with the highest estimated Q-value.

Besides taking advantage of the surprising property we found in BRIDGE, GORP has other properties which help us bridge the theory-practice gap in RL. It explores randomly, like common deep RL algorithms, meaning that it can give us insight into why random exploration works much better than the worst-case given in Theorem 2.1. Also, unlike other RL algorithms, it has cleanly separated *exploration* and *learning* stages, making it much easier to analyze than algorithms in which exploration and learning are entangled.

Algorithm 1 The Greedy Over Random Policy (GORP) algorithm, used to motivate the effective horizon.

```
procedure GORP($k, m, \pi^{\text{expl}}$)
    for $i = 1, \ldots, T$ do
        for $a_{i:i+k-1} \in \mathcal{A}^k$ do
            sample $m$ episodes following $\pi_1, \ldots, \pi_{i-1}$,
                then actions $a_{i:i+k-1}$, and finally $\pi^{\text{expl}}$.
            $\hat{Q}_i(s_i, a_{i:i+k-1}) \leftarrow$
                $\frac{1}{m} \sum_{j=1}^{m} \sum_{t=i}^{T} \gamma^{t-i} R(s_t^j, a_t^j)$.
        end for
        $\pi_i(s_i) \leftarrow \arg\max_{a_i \in \mathcal{A}}$
            $\max_{a_{i+1:i+k-1} \in \mathcal{A}^{k-1}} \hat{Q}_i(s_i, a_i, a_{i+1:i+k-1})$.
    end for
    return $\pi$
end procedure
```

Furthermore, GORP is extensible beyond environments satisfying the property we found in BRIDGE. First, it can solve MDPs where one may have to apply a few steps of value iteration to the random policy's Q-function before acting randomly. Second, it can use an "exploration policy" π^{expl} different from the random policy π^{rand}. These two generalizations are captured in the following definition. In the definition, we use the notation that a step of Q-value iteration transforms a Q-function Q to $Q' = \text{QVI}(Q)$, where

$$Q'_t(s,a) = R_t(s,a) + \max_{a' \in \mathcal{A}} Q_{t+1}\left(f(s,a), a'\right).$$

Definition 5.1 (k-QVI-solvable). *Given an exploration policy π^{expl} ($\pi^{expl} = \pi^{rand}$ unless otherwise noted), let $Q^1 = Q^{\pi^{expl}}$ and $Q^{i+1} = QVI(Q^i)$ for $i = 1, \ldots, T-1$. We say an MDP is k-*QVI-solvable *for some $k \in [T]$ if every policy in $\Pi(Q^k)$ is optimal.*

We will see that running GORP with $k > 1$ will allow it to find an optimal policy in MDPs that are k-QVI-solvable. Although the sample complexity of GORP scales with A^k, we find that nearly all of the environments in BRIDGE are k-QVI-solvable for very small values of k (Figure 6).

We now use GORP to define the effective horizon of an MDP. Note that the total number of timesteps sampled by GORP with parameters k and m is $T^2 A^k m = T^2 A^{k + \log_A m}$. Thus, analogously to how the horizon T appears in the exponent of the worst-case sample complexity bound $O(TA^T)$, we define the *effective* horizon as the exponent of A in the sample complexity of GORP:

Definition 5.2 (Effective horizon). *Given $k \in [T]$, let $H_k = k + \log_A m_k$, where m_k is the minimum value of m needed for Algorithm 1 with parameter k to return the optimal policy with probability at least $1/2$, or ∞ if no value of m suffices. The* effective horizon *is $H = \min_k H_k$.*

By definition, the sample complexity of GORP can be given using the effective horizon:

Lemma 5.3. *The sample complexity of GORP with optimal choices of k and m is $T^2 A^H$.*

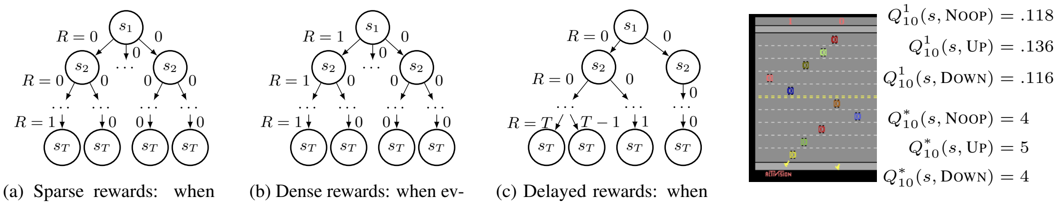
As we noted in the introduction, when $H \ll T$, as we find is often true in practice, this is far better than the worst-case bound given in Theorem 2.1 which scales with A^T.

Definition 5.2 does not give a method for actually calculating the effective horizon. It turns out we can bound the effective horizon using a generalized gap notion like those found throughout the RL theory literature. We denote by Δ_t^k the gap of the Q-function Q^k from Definition 5.1, where

$$\Delta_t^k(s) = \max_{a \in \mathcal{A}} Q_t^k(s,a) - \max_{a' \notin \arg\max_a Q_t^k(s,a)} Q_t^k(s,a').$$

The following theorem gives bounds on the effective horizon in terms of this gap.

Theorem 5.4. *Suppose that an MDP is k-QVI-solvable and that all rewards are nonnegative, i.e. $R(s,a) \geq 0$ for all s, a. Let Δ^k denote the gap of the Q-function Q^k as defined in Definition 5.1.*


s_1
$R=0$
0
s_2
$\cdots$
$R=1$
$R=T$
$T-1$
1
s_T
$Q^1_{10}(s, \text{Noop}) = .118$
$Q^1_{10}(s, \text{Up}) = .136$
$Q^1_{10}(s, \text{Down}) = .116$
$Q^*_{10}(s, \text{Noop}) = 4$
$Q^*_{10}(s, \text{Up}) = 5$
$Q^*_{10}(s, \text{Down}) = 4$
ACTIVISION

(a) Sparse rewards: when only one sequence of actions gives a reward of 1 and all others give 0, the effective horizon $H = \tilde{O}(T)$.

(b) Dense rewards: when every optimal action gives a reward of 1 and suboptimal actions give no reward, the effective horizon $H = \tilde{O}(1)$.

(c) Delayed rewards: when the rewards in (b) are all delayed to the end of the episode, the effective horizon remains $\tilde{O}(1)$.

(d) For the first 50 timesteps of the Atari game Freeway, we can bound $H \leq 10.2$, which is much lower than the horizon $T = 50$.

Figure 2: Examples of calculating the effective horizon H using Theorem 5.4; see Section 5.1 for the details.

Then

$$H_k \leq k + \max_{t \in [T], s \in \mathcal{S}_i^{opt}, a \in \mathcal{A}} \log_A \left(\frac{Q_t^k(s,a) V_t^*(s)}{\Delta_t^k(s)^2} \right) + \log_A 6 \log \left(2TA^k \right),$$

where $\mathcal{S}_i^{opt}$ is the set of states visited by some optimal policy at timestep i and $V_t^*(s) = \max_\pi V_t^\pi(s)$ is the optimal value function.

A full proof of Theorem 5.4 is given in Appendix A. Intuitively, the smaller the gap $\Delta_t^k(s)$, the more precisely we must estimate the Q-values in GORP in order to pick an optimal action.

The GORP algorithm is very amenable to theoretical analysis because it reduces the problem of finding an optimal policy to the problem of estimating several k-step Q-values, each of which is a simple mean of i.i.d. random variables. There are endless tail bounds that can be applied to analysis of GORP; we use some of these to obtain even tighter bounds on the effective horizon in Appendix C.

Why should the effective horizon, which is defined in terms of our GORP algorithm, also explain the performance of deep RL algorithms like PPO and DQN which are very different from GORP? In Appendix B, we present two algorithms, PG-GORP and FQI-GORP, which are more similar to PPO and DQN but whose sample complexities can still be bounded with the effective horizon. We also give additional bounds on the effective horizon and lower bounds on sample complexity.

5.1 Examples of the effective horizon

To gain some intuition for the bound in Theorem 5.4, consider the examples in Figure 2. MDP (a) has extremely sparse rewards, with a reward of 1 only given for a single optimal action sequence. However, note that this MDP is still 1-QVI-solvable by Definition 5.1. The maximum of the bound in Theorem 5.4 is at $t = 1$ with the optimal action, where $Q_1^1(s,a) = 1/A^{T-1}$, $V_1^*(s) = 1$, and $\Delta_1^1(s) = 1/A^{T-1}$. Ignoring logarithmic factors and constants gives $H \lesssim 1 + \log_A A^{T-1} = T$. That is, in the case of MDP (a), the effective horizon is no better than the horizon.

Next, consider MDP (b), which has dense rewards of 1 for every optimal action. Again, this MDP is 1-QVI-solvable. The maximum in the bound is obtained at $t = 1$ and the optimal action with $Q_1^1(s,a) \leq 2$, $V^*(s) = T$, and the gap $\Delta_1^1(s,a) \geq 1$. Again ignoring logarithmic factors gives in this case $H \lesssim 1 + \log_A T = \tilde{O}(1)$. In this case, the effective horizon is much shorter than the horizon, and barely depends on it! This again reflects our intuition that in this case, finding the optimal policy via RL should be much easier.

MDP (c) is similar to MDP (b) except that all rewards are delayed to the end of the episode. In this case, the Q function is the same as in MDP (b) so the effective horizon remains $\tilde{O}(1)$. This may seem counterintuitive since one needs to consider rewards T timesteps ahead to act optimally. However, the way GORP uses random rollouts to evaluate leaf nodes means that it can implicitly consider rewards quite far in the future even without needing to exhaustively plan that many timesteps ahead.

Finally, consider MDP (d), the first 50 timesteps of the Atari game Freeway, which is included in the BRIDGE dataset. This MDP is also 1-QVI-solvable and the maximum in the bound is obtained in the state shown in Figure 2d at timestep $t = 10$. Plugging in the Q values shown in the figure gives $H \leq 10.2$, which is far lower than the horizon $T = 50$. The low effective horizon reflects how this MDP is much easier than the worst case in practice. Both PPO and DQN are able to solve it with a sample complexity of less than 1.5 million timesteps, while the worst case bound would suggest a sample complexity greater than $50 \times 3^{50}/2 \approx 10^{25}$ timesteps!

Comparison to other bounds Intuitively, why might the effective horizon give better sample complexity bounds than previous works presented in Section 3? The MDP in Figure 2b presents a problem for the covering length and UCB-based bounds, both of which are $\Omega(A^T)$. The exponential sample complexity arises because these bounds depend on visiting every state in the MDP during training. In contrast, GORP doesn't need to visit every state to find an optimal policy. The effective horizon of $\tilde{O}(1)$ for MDP (b) reflects this, showing that our effective horizon-based bounds can actually be much smaller than the state space size, which is on the order of A^T for MDP (b).

The effective planning window (EPW) does manage to capture the same intuition as the effective horizon in the MDP in Figure 2b: in this case, $W = 1$. However, the analysis based on the EPW is unsatisfactory because it entirely ignores rewards beyond the planning window. Thus, in MDP (c) the EPW $W = T$, making EPW-based bounds no better than the worst case. In contrast, the effective horizon-based bound remains the same between MDPs (b) and (c), showing that it can account for the ability of RL algorithms to use rewards beyond the window where exhaustive planning is possible.

6 Experiments

We now show that sample complexity bounds based on the effective horizon predict the empirical performance of deep RL algorithms far better than other bounds in the literature. For each MDP in the BRIDGE dataset, we run deep RL algorithms to determine their empirical sample complexity. We also use the tabular representations of the MDPs to calculate the effective horizon and other sample complexity bounds for comparison.

Deep RL algorithms We run both PPO and DQN for five million timesteps for each MDP in BRIDGE, and record the empirical sample complexity (see Appendix F for hyperparameters and experiment details). PPO converges to the optimal policy in 95 of the 155 MDPs, and DQN does in 117 of 155. At least one of the two finds the optimal policy in 119 MDPs.

Sample complexity bounds We also compute sample complexity bounds for each MDP in BRIDGE. These include the worst-case bound of TA^T from Theorem 2.1, the effective-horizon-based bound of T^2A^H from Lemma 5.3, as well as three other bounds from the literature, introduced in Section 3 and proved in Appendix D: the UCB-based bound SAT, the covering-length-based bound TL, and the effective planning window (EPW)-based bound of T^2A^W.

Evaluation metrics To determine which sample complexity bounds best reflect the empirical performance of PPO and DQN, we compute a few summary metrics for each bound. First, we measure the *Spearman (rank) correlation* between the sample complexity bounds and the empirical sample complexity over environments where the algorithm converged to the optimal policy. The correlation (higher is better) is a useful for measuring how well the bounds can rank the relative difficulty of RL in different MDPs.

Second, we compute the *median ratio* between the sample complexity bound and the empirical sample complexity for environments where the algorithm converged. The ratio between the bound N_{bound} and empirical value N_{emp} is calculated as $\max\{N_{\text{bound}}/N_{\text{emp}}, N_{\text{emp}}/N_{\text{bound}}\}$. For instance, a median ratio of 10 indicates that half the sample complexity bounds were within a factor of 10 of the empirical sample complexity. Lower values indicate a better bound; this metric is useful for determining whether the sample complexity bounds are vacuous or tight.

Finally, we consider the binary classification task of predicting whether the algorithm will converge at all within five million steps using the sample complexity bounds. That is, we consider simply thresholding each sample complexity bound and predicting that only environments with bounds below the threshold will converge. We compute the *area under the ROC curve (AUROC)* for this prediction task as well as the *accuracy* with the optimal threshold. Higher AUROC and accuracy both indicate a better bound.

Results The results of our experiments are shown in Table 2. The effective horizon-based bounds have higher correlation with the empirical sample complexity than the other bounds for both PPO and DQN. While the EPW-based bounds are also reasonably correlated, they are significantly off in absolute terms: the typical bound based on the EPW is 3-4 orders of magnitude off, while the effective horizon yields bounds that are typically within 1-2 orders of magnitude. The UCB-based

	PPO				DQN			
Bound	Correl.	Median ratio	AUROC	Acc.	Correl.	Median ratio	AUROC	Acc.
Worst-case ($T\lceil A^T/2\rceil$)	0.24	7.2×10^{10}	0.57	0.63	0.15	5.5×10^{10}	0.67	0.76
Covering length (TL)	0.35	6.3×10^6	0.78	0.72	0.27	3.9×10^6	0.86	0.85
EPW (T^2A^W)	0.69	1.1×10^5	0.78	0.75	0.58	8.0×10^4	0.88	0.85
UCB (SAT)	0.26	**20**	0.68	0.67	0.31	**31**	0.67	0.77
Effective horizon (T^2A^H)	**0.81**	31	**0.92**	**0.86**	**0.74**	67	**0.92**	**0.86**
Other deep RL algorithm	0.77	2.3	0.84	0.85	0.77	2.3	0.86	0.99
GORP empirical	0.79	7.3	0.77	0.82	0.65	11	0.80	0.94

Table 2: Effective horizon-based sample complexity bounds are the most predictive of the real performance of PPO and DQN according to the four metrics we describe in Section 6. The effective horizon bounds are about as good at predicting the sample complexity of PPO and DQN as one algorithm's sample complexity is at predicting the other's.

	PPO		DQN	
Bound	Correl.	Ratio	Correl.	Ratio
EPW	0.20	2.1	**0.70**	12
Effective horizon	**0.48**	2.4	0.35	12
Other bounds	0.00	**1.3**	0.00	**1.9**

(a) Reward shaping.

	PPO	
Bound	Correl.	Ratio
Covering length	-0.36	2.5×10^4
Effective horizon	**0.57**	2.7
Other bounds	0.00	**2.2**

(b) Initializing with a policy trained on human data or transferred from similar environments.

Table 3: The effective horizon explains the effects of reward shaping and initializing with a pretrained policy by accurately predicting their effects on the empirical sample complexity of PPO and DQN. Correlation and median ratio are measured between the predicted change in sample complexity and the empirical change. See Section 6 for further discussion.

bounds are somewhat closer to the empirical sample complexity, but are not well correlated; this makes sense since the UCB bounds depend on strategic exploration, while PPO and DQN use random exploration. Finally, the effective horizon bounds are able to more accurately predict whether PPO or DQN will find an optimal policy, as evidenced by the AUROC and accuracy metrics.

As an additional baseline, we also calculate the four evaluation metrics when using the empirical sample complexity of PPO to predict the empirical sample complexity of DQN, or vice-versa, and when using the empirical sample complexity of GORP to predict PPO or DQN's performance (bottom two rows of Table 2). While these are not provable bounds, they provide another point of comparison for each metric. The effective horizon-based bounds correlate about as well with PPO and DQN's sample complexities as they do with each other's. The empirical performance of GORP is typically even closer to that of PPO and DQN than the effective horizon-based bounds.

Reward shaping, human data, and transfer learning In order for RL theory to be useful practically, it should help practitioners make decisions about which techniques to apply in order to improve their algorithms' performance. We show how the effective horizon can be used to explain the effect of three such techniques: reward shaping, using human data, and transfer learning.

Potential-based reward shaping [45] is a classic technique which can speed up the convergence of RL algorithms. It is generally used to transform a sparse-reward MDP like the one in Figure 2a to a dense-reward MDP like in Figure 2b without changing the optimal policy. If the effective horizon is a good measure of the difficulty of RL in an environment, then it should be able to predict whether (and by how much) the sample complexity of practical algorithms changes when reward shaping is applied. We develop 77 reward-shaped versions of the original 33 Minigrid environments and run PPO and DQN. Results in Table 3a show the effective horizon accurately captures the change in sample complexity from using reward shaping. We use similar metrics to those in Table 2: the correlation between the predicted and empirical ratio of the shaped sample complexity to the unshaped sample complexity, and the median ratio between the predicted change and the actual change.

Out of the five bounds we consider, three—worst case, covering length, and UCB—don't even depend on the reward function. The EPW does depend on the reward function and captures some of the effect of reward shaping for DQN, but does worse at predicting the effect on PPO. In comparison, the effective horizon does well for both algorithms, showing that it can accurately capture how reward shaping affects RL performance.

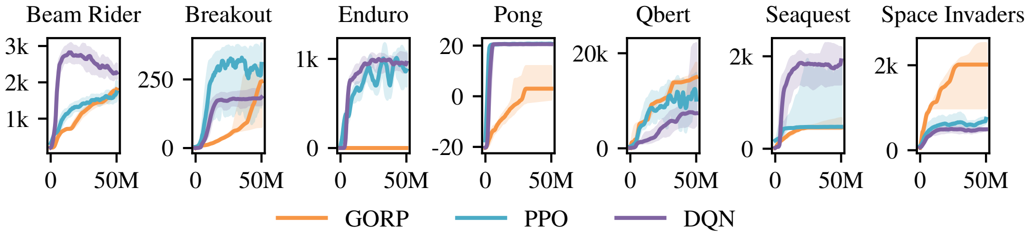


Figure 3: Learning curves for PPO, DQN, and GORP on full-horizon Atari games. We use 5 random seeds for all algorithms. The solid line shows the median return throughout training while the shaded region shows the range of returns over random seeds.

Another tool used to speed up RL is initializing with a pre-trained policy, which is used practically to make RL work on otherwise intractable tasks. Can the effective horizon also predict whether initializing RL with a pre-trained policy will help? We initialize PPO with pre-trained policies for 82 of the MDPs in BRIDGE, then calculate new sample complexity bounds based on using the pre-trained policies as an exploration policy π^{expl}. Table 3b shows that the effective horizon accurately predicts the change in sample complexity due to using a pre-trained policy. Again, three bounds—worst case, EPW, and UCB—do not depend on the exploration policy at all, while the covering length gives wildly inaccurate predictions for its effect. In contrast, the effective horizon is accurate at predicting the changes in sample complexities when using pre-trained policies.

Long-horizon environments We also perform experiments on full-length Atari games to evaluate the predictive power of the effective horizon in longer-horizon environments. It is intractable to construct tabular representations of these environments and thus we cannot compute instance-dependent sample complexity bounds. However, it is still possible to compare the empirical performance of PPO, DQN, and GORP. If the performance of GORP is close to that of PPO and DQN, then this suggests that the effective horizon, which is defined in terms of GORP, can explain RL performance in these environments as well. Figure 3 compares the learning curves of PPO, DQN, and GORP in deterministic versions of the seven Atari games from Mnih et al. [46] (see Appendix F.1 for details). GORP performs better than both PPO and DQN in two games and better than at least one deep RL algorithm in an additional three games. This provides evidence that the effective horizon is also predictive of RL performance in long-horizon environments.

7 Discussion

Overall, our results suggest the effective horizon is a key measure of the difficulty of solving an MDP via reinforcement learning. The intuition behind the effective horizon presented in Section 5 and the empirical evidence in Section 6 both support its importance for better understanding RL.

Limitations While we have presented a thorough theoretical and empirical justification of the effective horizon, there are still some limitations to our analysis. First, we focus on deterministic MDPs with discrete action spaces, leaving the extension to stochastic environments and those with continuous action spaces an open question. Furthermore, the effective horizon is not easily calculable without full access to the MDP's tabular representation. Despite this, it serves as a useful perspective for understanding RL's effectiveness and potential improvement areas. An additional limitation is that the effective horizon cannot capture the performance effects of generalization—the ability to use actions that work well at some states for other similar states. For an example where the effective horizon fails to predict generalization, see Appendix G.1. However, the effective horizon is still quite predictive of deep RL performance even without modeling generalization.

Implications and future work We hope that this paper helps to bring the theoretical and empirical RL communities closer together in pursuit of shared understanding. Theorists can extend our analysis of the effective horizon to new algorithms or explore related properties, using our BRIDGE dataset to ground their investigations by testing assumptions in real environments. Empirical RL researchers can use the effective horizon as a foundation for new algorithms. For instance, Brandfonbrener et al. [47] present an offline RL algorithm similar to GORP with $k = 1$; our theoretical understanding of GORP might provide insights for improving it or developing related algorithms.

Acknowledgments and Disclosure of Funding

We would like to thank Jacob Steinhardt, Kush Bhatia, Lawrence Chan, Paria Rashidinejad, Ruiqi Zhong, and Yaodong Yu for feedback on drafts as well as Jiantao Jiao and Banghua Zhu for helpful discussions.

This work was partially supported by the Open Philanthropy Project, the Office of Naval Researcher's Young Investigator Program (YIP), and Weill Neurohub. Cassidy Laidlaw is supported by an Open Philanthropy AI Fellowship and a National Defense Science and Engineering Graduate (NDSEG) Fellowship. Stuart Russell is supported by an AI 2050 Senior Fellowship from Schmidt Futures.

References

[1] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, January 2016. ISSN 1532-4435.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL `https://www.nature.com/articles/nature16961`. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 7587 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computational science;Computer science;Reward Subject_term_id: computational-science;computer-science;reward.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL `https://www.nature.com/articles/nature14236`. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 7540 Primary_atype: Research Publisher: Nature Publishing Group Subject_term: Computer science Subject_term_id: computer-science.

[4] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax Regret Bounds for Reinforcement Learning. *arXiv:1703.05449 [cs, stat]*, July 2017. URL `http://arxiv.org/abs/1703.05449`. arXiv: 1703.05449.

[5] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I. Jordan. Is Q-learning Provably Efficient? *arXiv:1807.03765 [cs, math, stat]*, July 2018. URL `http://arxiv.org/abs/1807.03765`. arXiv: 1807.03765.

[6] Chris Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for Epsilon-Greedy Reinforcement Learning with Function Approximation. In *Proceedings of the 39th International Conference on Machine Learning*, pages 4666–4689. PMLR, June 2022. URL `https://proceedings.mlr.press/v162/dann22a.html`. ISSN: 2640-3498.

[7] Yao Liu and Emma Brunskill. When Simple Exploration is Sample Efficient: Identifying Sufficient Conditions for Random Exploration to Yield PAC RL Algorithms, April 2019. URL `http://arxiv.org/abs/1805.09045`. arXiv:1805.09045 [cs, stat].

[8] Dhruv Malik, Aldo Pacchiano, Vishwak Srinivasan, and Yuanzhi Li. Sample Efficient Reinforcement Learning In Continuous State Spaces: A Perspective Beyond Linearity. In *Proceedings of the 38th International Conference on Machine Learning*, pages 7412–7422. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/malik21c.html`. ISSN: 2640-3498.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, August 2017. URL `http://arxiv.org/abs/1707.06347`. arXiv: 1707.06347.

[10] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning, October 2017. URL `http://arxiv.org/abs/1710.02298`. arXiv:1710.02298 [cs].

[11] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 588(7839):604–609, December 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-03051-4. URL `http://arxiv.org/abs/1911.08265`. arXiv:1911.08265 [cs, stat].

[12] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite, January 2018. URL `http://arxiv.org/abs/1801.00690`. arXiv:1801.00690 [cs].

[13] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, October 2012. doi: 10.1109/IROS.2012.6386109. ISSN: 2153-0866.

[14] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic Gridworld Environment for OpenAI Gym, 2018. URL `https://github.com/maximecb/gym-minigrid`. Publication Title: GitHub repository.

[15] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State Entropy Maximization with Random Encoders for Efficient Exploration. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9443–9454. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/seo21a.html`. ISSN: 2640-3498.

[16] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent Complexity and Zero-shot Transfer via Unsupervised Environment Design. In *Advances in Neural Information Processing Systems*, volume 33, pages 13049–13061. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/985e9a46e10005356bbaf194249f6856-Abstract.html`.

[17] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A Platform to Study the Sample Efficiency of Grounded Language Learning, December 2019. URL `http://arxiv.org/abs/1810.08272`. arXiv:1810.08272 [cs].

[18] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, University of London, University College London (United Kingdom), 2003.

[19] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual Decision Processes with low Bellman rank are PAC-Learnable. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1704–1713. PMLR, July 2017. URL `https://proceedings.mlr.press/v70/jiang17c.html`. ISSN: 2640-3498.

[20] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I. Jordan. Provably Efficient Reinforcement Learning with Linear Function Approximation. *arXiv:1907.05388 [cs, math, stat]*, August 2019. URL `http://arxiv.org/abs/1907.05388`. arXiv: 1907.05388.

[21] Simon Du, Sham Kakade, Jason Lee, Shachar Lovett, Gaurav Mahajan, Wen Sun, and Ruosong Wang. Bilinear Classes: A Structural Framework for Provable Generalization in RL. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2826–2836. PMLR, July 2021. URL `https://proceedings.mlr.press/v139/du21a.html`. ISSN: 2640-3498.

[22] Chi Jin, Qinghua Liu, and Sobhan Miryoosefi. Bellman Eluder Dimension: New Rich Classes of RL Problems, and Sample-Efficient Algorithms. In *Advances in Neural Information Processing Systems*, volume 34, pages 13406–13418. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/6f5e4e86a87220e5d361ad82f1ebc335-Abstract.html`.

[23] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002. ISSN ISSN 1533-7928. URL https://www.jmlr.org/papers/v3/brafman02a.html.

[24] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes. *Machine Learning*, 49(2):193–208, November 2002. ISSN 1573-0565. doi: 10.1023/A:1017932429737. URL https://doi.org/10.1023/A:1017932429737.

[25] Nan Jiang, Satinder Singh, and Ambuj Tewari. On structural properties of MDPs that bound loss due to shallow planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 1640–1647, New York, New York, USA, July 2016. AAAI Press. ISBN 978-1-57735-770-4.

[26] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado Van Hasselt. Behaviour Suite for Reinforcement Learning, February 2020. URL http://arxiv.org/abs/1908.03568. arXiv:1908.03568 [cs, stat].

[27] Raghu Rajan, Jessica Lizeth Borja Diaz, Suresh Guttikonda, Fabio Ferreira, André Biedenkapp, Jan Ole von Hartz, and Frank Hutter. MDP Playground: A Design and Debug Testbed for Reinforcement Learning, June 2021. URL http://arxiv.org/abs/1909.07750. arXiv:1909.07750 [cs, stat].

[28] R. Devon Hjelm, Bogdan Mazoure, Florian Golemo, Felipe Frujeri, Mihai Jalobeanu, and Andrey Kolobov. The Sandbox Environment for Generalizable Agent Research (SEGAR), March 2022. URL http://arxiv.org/abs/2203.10351. arXiv:2203.10351 [cs].

[29] Michelangelo Conserva and Paulo Rauber. Hardness in Markov Decision Processes: Theory and Practice, October 2022. URL http://arxiv.org/abs/2210.13075. arXiv:2210.13075 [cs].

[30] B. Abramson. Expected-outcome: a general model of static evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):182–193, February 1990. ISSN 1939-3539. doi: 10.1109/34.44404. URL https://ieeexplore.ieee.org/abstract/document/44404?casa_token=uj3nSMaKLJcAAAAA:d7LSY9mmrlNL8qhkLN6WEKgrZpmv1pRnvwj0cAolitRmNtbTUFegNNstXuXCEwBnrPayNKBO. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[31] Bernd Brügmann. Monte Carlo Go. 1993. URL https://www.semanticscholar.org/paper/Monte-Carlo-Go-Max-Planck/5f8b18be86be69077e66377e03198d21d06833f3.

[32] Gerald Tesauro and Gregory Galperin. On-line Policy Improvement using Monte-Carlo Search. In *Advances in Neural Information Processing Systems*, volume 9. MIT Press, 1996. URL https://proceedings.neurips.cc/paper/1996/hash/996009f2374006606f4c0b0fda878af1-Abstract.html.

[33] Dimitri P. Bertsekas, John N. Tsitsiklis, and Cynara Wu. Rollout Algorithms for Combinatorial Optimization. *Journal of Heuristics*, 3(3):245–262, December 1997. ISSN 1572-9397. doi: 10.1023/A:1009635226865. URL https://doi.org/10.1023/A:1009635226865.

[34] Hyeong Soo Chang, Michael C. Fu, Jiaqiao Hu, and Steven I. Marcus. An Adaptive Sampling Algorithm for Solving Markov Decision Processes. *Operations Research*, 53(1):126–139, February 2005. ISSN 0030-364X. doi: 10.1287/opre.1040.0145. URL https://pubsonline.informs.org/doi/10.1287/opre.1040.0145. Publisher: INFORMS.

[35] Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, pages 282–293, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-46056-5. doi: 10.1007/11871842_29.

[36] Rémi Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, Lecture Notes in Computer Science, pages 72–83, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-75538-8. doi: 10.1007/978-3-540-75538-8_7.

[37] B. Bouzy and B. Helmstetter. Monte-Carlo Go Developments. In H. Jaap Van Den Herik, Hiroyuki Iida, and Ernst A. Heinz, editors, *Advances in Computer Games: Many Games, Many Challenges*, IFIP — The International Federation for Information Processing, pages 159–174. Springer US, Boston, MA, 2004. ISBN 978-0-387-35706-5. doi: 10.1007/978-0-387-35706-5_11. URL https://doi.org/10.1007/978-0-387-35706-5_11.

[38] Michael Chung, Michael Buro, and Jonathan Schaeffer. Monte Carlo Planning in RTS Games. January 2005.

[39] Dimitri Bertsekas. *Rollout, Policy Iteration, and Distributed Reinforcement Learning*. Athena Scientific, Belmont, Massachusetts, first edition edition, August 2020. ISBN 978-1-886529-07-6.

[40] Dimitri P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, March 2022. ISBN 978-1-886529-17-5.

[41] D. Kleinman. On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1):114–115, February 1968. ISSN 1558-2523. doi: 10.1109/TAC.1968.1098829. URL https://ieeexplore.ieee.org/abstract/document/1098829?casa_token=tKFm8Om5gI4AAAAA:E3oaouksJy-5g-HuySnACwNdxyHpRDk8IbPTssBOB-PK0qZng_-v9Fsk7qzcyYyYVudFUrI8. Conference Name: IEEE Transactions on Automatic Control.

[42] Martin L. Puterman and Shelby L. Brumelle. On the Convergence of Policy Iteration in Stationary Dynamic Programming. *Mathematics of Operations Research*, 4(1):60–69, 1979. ISSN 0364-765X. URL https://www.jstor.org/stable/3689239. Publisher: INFORMS.

[43] 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, June 2013. ISSN 1076-9757. doi: 10.1613/jair.3912. URL https://www.jair.org/index.php/jair/article/view/10819.

[44] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning, July 2020. URL http://arxiv.org/abs/1912.01588. arXiv:1912.01588 [cs, stat].

[45] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *In Proceedings of the Sixteenth International Conference on Machine Learning*, pages 278–287. Morgan Kaufmann, 1999.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning, December 2013. URL http://arxiv.org/abs/1312.5602. arXiv:1312.5602 [cs].

[47] David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL Without Off-Policy Evaluation, December 2021. URL http://arxiv.org/abs/2106.08909. arXiv:2106.08909 [cs, stat].

[48] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In Richard S. Sutton, editor, *Reinforcement Learning*, The Springer International Series in Engineering and Computer Science, pages 5–32. Springer US, Boston, MA, 1992. ISBN 978-1-4615-3618-5. doi: 10.1007/978-1-4615-3618-5_2. URL https://doi.org/10.1007/978-1-4615-3618-5_2.

[49] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(18):503–556, 2005. ISSN 1533-7928. URL http://jmlr.org/papers/v6/ernst05a.html.

[50] Martin Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *Machine Learning: ECML 2005*, Lecture Notes in Computer Science, pages 317–328, Berlin, Heidelberg, 2005. Springer. ISBN 978-3-540-31692-3. doi: 10.1007/11564096_32.

[51] Irina Shevtsova. On the absolute constants in the Berry-Esseen type inequalities for identically distributed summands, November 2011. URL `http://arxiv.org/abs/1111.6554`. arXiv:1111.6554 [math].

[52] Eyal Even-Dar and Yishay Mansour. Learning Rates for Q-learning. *Journal of Machine Learning Research*, 5(Dec):1–25, 2003. ISSN ISSN 1533-7928. URL `https://www.jmlr.org/papers/v5/evendar03a.html`.

[53] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. The Dependence of Effective Planning Horizon on Model Accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1181–1189, Richland, SC, May 2015. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-3413-6.

[54] Alex Braylan, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen. Frame skip is a powerful parameter for learning to play Atari. In *Workshops at the twenty-ninth AAAI conference on artificial intelligence*, 2015.

[55] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for Datasets, December 2021. URL `http://arxiv.org/abs/1803.09010`. arXiv:1803.09010 [cs].

[56] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, June 2016. URL `http://arxiv.org/abs/1606.01540`. arXiv:1606.01540 [cs].

[57] Ruohan Zhang, Calen Walshe, Zhuode Liu, Lin Guan, Karl S. Muller, Jake A. Whritner, Luxin Zhang, Mary M. Hayhoe, and Dana H. Ballard. Atari-HEAD: Atari Human Eye-Tracking and Demonstration Dataset, September 2019. URL `http://arxiv.org/abs/1903.06754`. arXiv:1903.06754 [cs, stat].

[58] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1):12348–12355, 2021. URL `https://dl.acm.org/doi/abs/10.5555/3546258.3546526`. Publisher: JMLRORG.

[59] Antonin Raffin. RL Baselines3 Zoo, 2020. URL `https://github.com/DLR-RM/rl-baselines3-zoo`. Publication Title: GitHub repository.

Appendix

A Proofs of main results

A.1 Proof of Theorem 2.1

Theorem 2.1. *There is an RL algorithm which can solve any deterministic MDP with sample complexity $N \leq T\lceil A^T/2 \rceil$. Conversely, for any RL algorithm and any values of T and A, there must be some deterministic MDP for which its sample complexity $N \geq T(\lceil A^T/2 \rceil - 1)$.*

Proof. Consider the following RL algorithm:

```
procedure ExhaustiveSearch
    $\mathcal{T} \leftarrow$ Shuffle($\mathcal{A}^T$)
    $\mathcal{J}$ is an array of size $\lceil A^T/2 \rceil$
    for $i = 1, \ldots, \lceil A^T/2 \rceil$ do
        run one episode, taking the actions in $\mathcal{T}[i]$
        $\mathcal{J}[i] \leftarrow \sum_{t=1}^T \gamma^t R(s_t, a_t)$
    end for
    $i^* \leftarrow \arg\max_i \mathcal{J}[i]$
    return the policy which takes the actions in $\mathcal{T}[i^*]$
end procedure
```

Since the MDP is deterministic, an RL algorithm only needs to find an optimal sequence of actions. Clearly, there is at least a $1/2$ chance that some optimal sequence of actions is in the first $\lceil A^T/2 \rceil$ elements of $\mathcal{T}$. If this is the case, then ExhaustiveSearch will return an optimal policy corresponding to that optimal sequence. Since the number of environment timesteps taken by ExhaustiveSearch is equal to $T\lceil A^T/2 \rceil$, we have that $N \leq T\lceil A^T/2 \rceil$.

For the converse, fix A and T along with any RL algorithm. Consider a set of states indexed by sequences of actions of length 0 to T:

$$\mathcal{S} = \{s_{a_{1:\ell}} \mid \ell \in 0, \ldots, T, a_{1:\ell} \in \mathcal{A}^\ell\}.$$

Then, define a transition function

$$f(s_{a_{1:\ell}}, a) = s_{a_{1:\ell}, a}.$$

Now consider A^T different MDPs which share the state space $\mathcal{S}$ and transition function f, differing only in their reward functions:

$$\mathbb{M} = \{\mathcal{M}_{a_{1:T}} \mid a_{1:T} \in \mathcal{A}^T\} \quad \text{where the MDP } \mathcal{M}_{a_{1:T}} \text{ has } R(s_{a_{1:\ell}}, a) = \begin{cases} 1 & a_{1:\ell}, a = a_{1:T} \\ 0 & \text{otherwise.} \end{cases}$$

That is, each MDP has a single optimal sequence of actions that gives reward 1 on the final timestep; all other rewards are 0.

Let the RL algorithm in question take in some source of randomness z and output a policy $\pi_t^n(a \mid s; z, \mathcal{M})$ after n timesteps in MDP $\mathcal{M}$. Now suppose by way of contradiction the sample complexity of the algorithm is less than $T(\lceil A^T/2 \rceil - 1)$ in all MDPs in $\mathbb{M}$. By our definition of sample complexity, this means that

$$\forall \mathcal{M} \in \mathbb{M} \qquad \mathbb{P}_z \left(J \left(\pi_t^{T(\lceil A^T/2 \rceil - 2)}(\cdot; z, \mathcal{M}) \right) = 1 \right) \geq 1/2. \tag{1}$$

Clearly a policy can only be optimal in these MDPs if it is deterministic, so let $\tau(z, \mathcal{M})$ be the sequence of actions that $\pi_t^{T(\lceil A^T/2 \rceil - 2)}(a \mid s; z, \mathcal{M})$ takes if it is optimal, and let it be any suboptimal sequence of actions if the policy is suboptimal. We can rewrite (1) as

$$\forall \mathcal{M}_{a_{1:T}} \in \mathbb{M} \qquad \mathbb{P}_z \left(\tau(z, \mathcal{M}_{a_{1:T}}) = a_{1:T} \right) \geq 1/2.$$

Letting Unif($\mathcal{A}^T$) define a uniform distribution over action sequences of length T, this implies

$$\mathbb{P}_{a_{1:T} \sim \text{Unif}(\mathcal{A}^T), z} \Big(\tau(z, \mathcal{M}_{a_{1:T}}) = a_{1:T} \Big) \geq 1/2$$

which means that there must be some particular z such that

$$\mathbb{P}_{a_{1:T} \sim \text{Unif}(\mathcal{A}^T)} \Big(\tau(z, \mathcal{M}_{a_{1:T}}) = a_{1:T} \Big) \geq 1/2$$

$$\Big| \big\{ a_{1:T} \in \mathcal{A}^T \mid \tau(z, \mathcal{M}_{a_{1:T}}) = a_{1:T} \big\} \Big| \geq A^T/2 \tag{2}$$

Given that the RL algorithm is now deterministic due to the fixed z, we will prove that the LHS of (2) must be less than or equal to $\lceil A^T/2 \rceil - 1$. This can be shown via induction. For the first episode, the algorithm must take the same actions in every MDP since all MDPs give zero reward until the final action (so there is no way to distinguish them). After the first episode, only one MDP can be distinguished from the others: the one corresponding to the action sequence taken in the first episode, which has reward 1 instead of 0. Thus in the remaining $A^T - 1$ MDPs the algorithm must take the same actions in the second episode. Continuing this argument shows that after episode $\lceil A^T/2 \rceil - 2$, the algorithm must still be unable to distinguish between $A^T - \big(\lceil A^T/2 \rceil - 2\big) = \lfloor A^T/2 \rfloor + 2$ of the MDPs, and so $\tau(z, \mathcal{M})$ must be the same for all MDPs in this set. Since all of these MDPs have different optimal action sequences, $\tau(z, \mathcal{M})$ can only be optimal in one of them. Thus $\tau(z, \mathcal{M})$ must be suboptimal in at least $\lfloor A^T/2 \rfloor + 1$ MDPs, which means the LHS of (2) must be at most $A^T - (\lfloor A^T/2 \rfloor + 1) = \lceil A^T/2 \rceil - 1$.

Combining this with (2) gives $\lceil A^T/2 \rceil - 1 \geq A^T/2$, which is a contradiction. Thus, the sample complexity of the RL algorithm must be at least $T(\lceil A^T/2 \rceil - 1)$. ■

A.2 Proof that $k = T$ in the worst case

Lemma A.1. *Let $Q^1 = Q^{\pi^{rand}}$, $Q^{i+1} = QVI(Q^i)$ for $i = 2, \dots, T$, and $\Pi(Q^i)$ be defined as in Section 4. Then for any horizon T and number of actions $A \geq 2$ there is an MDP such that no policy in $\Pi(Q^i)$ is optimal for $i < T$.*

Proof. As in the proof of Theorem 2.1, define an MDP where every action sequence leads to a different state. Pick an arbitrary action sequence $a_{1:T}$, and let the reward of taking the final action in that sequence be 1:

$$R(s_{a_{1:T-1}}, a_T) = 1.$$

Now take some action $a_1' \neq a_1$, and let the reward for taking that action at the beginning of an episode be $3/4$:

$$R(s_1, a_1') = 3/4.$$

Let the rewards for all other state-action pairs be 0. We will show by induction that

$$Q_t^i(s_{a_{1:t-1}}, a_t) = \frac{1}{A^{\max\{T-i-t+1,0\}}} \qquad \text{and} \qquad Q_1^i(s_1, a_1') = 3/4. \tag{3}$$

The left half of (3) is clearly true for $i = 1$, since the random policy will take all action sequences following the $t-1$-th optimal action with probability $1/A^{T-t}$, and exactly one of those gives reward 1. The right half is also clear since following a_1' gives immediate reward of 3/4 and then no reward afterwards.

For the inductive step, we begin with the left half of (3); assume it holds for some i. By the definition of Q-value iteration, we have for $t < T$

$$\begin{aligned} &Q_t^{i+1}(s_{a_{1:t-1}}, a_t) \\ &\overset{\text{(i)}}{=} R(s_{a_{1:t-1}}, a_t) + \max_a Q_{t+1}^i(s_{a_{1:t}}, a) \\ &= 0 + \frac{1}{A^{\max\{T-i-(t+1)+1,0\}}} \\ &= \frac{1}{A^{\max\{T-(i+1)-t+1,0\}}} \end{aligned}$$

and for $t = T$,

$$Q_T^{i+1}(s_{a_{1:T-1}}, a_T) = R(s_{a_{1:T-1}}, a_T) = 1 = \frac{1}{A^0} = \frac{1}{A^{\max\{T-(i+1)-t+1,0\}}}.$$

(i) is because only one action at timestep $t+1$ can have a Q-value greater than 0, since only one action sequence leads to reward after taking a_1. For the right half of (3), we have

$$Q_1^{i+1}(s_1, a_1') = R(s_1, a_1') + \max_a Q_2^i(s_{a_1'}, a) = 3/4 + 0 = 3/4.$$

which is due to no reward being possible after taking a_1' at the first timestep.

Given that (3) holds for $i = 1, \dots, T$, it is easy to see that for $i < T$, any $\pi \in \Pi(Q^i)$ will take a_1' on the first timestep, since

$$Q_1^i(s_1, a_1) = \frac{1}{A^{\max\{T-i,0\}}} = \frac{1}{A^{T-i}} \leq \frac{1}{A} \leq \frac{1}{2} \leq \frac{3}{4} = Q_1^i(s_1, a_1').$$

This means that π will be suboptimal, since a_1 is the only optimal action at $t = 1$. ■

A.3 Proof of Lemma 5.3

Lemma 5.3. *The sample complexity of GORP with optimal choices of k and m is $T^2 A^H$.*

Proof. Recall the definition of effective horizon:

Definition 5.2 (Effective horizon)**.** *Given $k \in [T]$, let $H_k = k + \log_A m_k$, where m_k is the minimum value of m needed for Algorithm 1 with parameter k to return the optimal policy with probability at least $1/2$, or ∞ if no value of m suffices. The* effective horizon *is $H = \min_k H_k$.*

Let $k \in \arg\min_k H_k$. Then by Definition 5.2, GORP (Algorithm 1) with parameters k, m_k will converge to an optimal policy with probability at least $1/2$. Clearly, Algorithm 1 interacts with the environment for T iterations, each of which require evaluating A^k action sequences with m_k episodes of T timesteps each, for a total of

$$T^2 A^k m_k = T^2 A^{H_k} = T^2 A^H$$

timesteps. Thus the sample complexity of GORP satisfies $N_{\text{GORP}} \leq T^2 A^H$.

Now, suppose by way of contradiction that $N_{\text{GORP}} < T^2 A^H$. Then this must mean that there are parameters k', m' such running GORP with these parameters converges to an optimal policy with probability at least $1/2$, and

$$k' + \log_A m' < k + \log_A m_k.$$

By Definition 5.2, this means that $m_{k'} \leq m'$, and thus

$$H = \min_k H \leq k' + \log_A m_{k'} \leq k' + \log_A m_{\prime} < k + \log_A m_k = H$$

which is clearly a contradiction. Thus, it must be that $N_{\text{GORP}} = T^2 A^H$. ■

A.4 Proof of Theorem 5.4

Theorem 5.4. *Suppose that an MDP is k-QVI-solvable and that all rewards are nonnegative, i.e. $R(s,a) \geq 0$ for all s, a. Let Δ^k denote the gap of the Q-function Q^k as defined in Definition 5.1. Then*

$$H_k \leq k + \max_{t \in [T], s \in \mathcal{S}_i^{opt}, a \in \mathcal{A}} \log_A \left(\frac{Q_t^k(s,a) V_t^*(s)}{\Delta_t^k(s)^2} \right) + \log_A 6 \log\left(2TA^k\right),$$

where $\mathcal{S}_i^{opt}$ is the set of states visited by some optimal policy at timestep i and $V_t^*(s) = \max_\pi V_t^\pi(s)$ is the optimal value function.

Proof. Let

$$m = \log\left(2TA^k\right) \max_{t \in [T], s \in \mathcal{S}_t^{\text{opt}}, a \in \mathcal{A}} \frac{6 Q_t^k(s,a) V_t^*(s)}{\Delta_t^k(s)^2}.$$

We will show that GORP (Algorithm 1) converges to the optimal policy with probability at least $1/2$ given parameters k and m. By Definition 5.2, this means the effective horizon must be at most $k + \log_A m$, which gives the bound in the theorem. More precisely, we will show that GORP converges to a policy in $\Pi(Q^k)$ with probability at least $1/2$, which must be optimal because of the assumption that the MDP is k-QVI-solvable.

First, we will show the following relationship between the k-action Q^1 values and Q^k:

$$Q_i^k(s_i, a_i) = \max_{a_{i+1:i+k-1} \in \mathcal{A}^{k-1}} Q_i^1(s_i, a_i, a_{i+1:i+k-1}). \tag{4}$$

We prove that (4) holds inductively. For $k = 1$, (4) is obviously true. Supposing it holds for k, then

$$\begin{aligned} Q_i^{k+1}(s_i, a_i) &= \text{QVI}(Q_i^k)(s_i, a_i) \\ &= R(s_i, a_i) + \max_{a_{i+1} \in \mathcal{A}} Q_{i+1}^k(f(s_i, a_i), a_{i+1}) \\ &\overset{\text{(i)}}{=} R(s_i, a_i) + \max_{a_{i+1:i+k} \in \mathcal{A}^k} Q_{i+1}^1(f(s_i, a_i), a_{i+1:i+k}) \\ &= \max_{a_{i+1:i+k} \in \mathcal{A}^k} Q_i^1(s_i, a_i, a_{i+1:i+k}), \end{aligned}$$

which shows (4) holds for $k+1$. (i) holds due to the inductive hypothesis.

Recall that in Algorithm 1, we use $\hat{Q}_i(s_i, a_{i:i+k-1})$ to denote the estimated Q-value of the k-action sequence $a_{i:i+k-1}$. Analogously to (4), define

$$\hat{Q}_i^k(s_i, a_i) = \max_{a_{i+1:i+k-1} \in \mathcal{A}^{k-1}} \hat{Q}_i(s_i, a_i, a_{i+1:i+k-1})$$

to be the maximum estimated Q-value of any action sequence that starts with a_i. We can rewrite line 7 of Algorithm 1 as

$$\pi_i(s_i) \leftarrow \arg\max_{a_i \in \mathcal{A}} \hat{Q}_i^k(s_i, a_i).$$

That is, the action that GORP selects for timestep i is chosen from those with the highest values of $\hat{Q}_i^k(s_i, a_i)$. Suppose we can show that

$$\mathbb{P}\left(\arg\max_{a_i \in \mathcal{A}} \hat{Q}_i^k(s_i, a_i) \subseteq \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)\right) \geq 1 - \frac{1}{2T} \tag{5}$$

holds for each $i \in [T]$. Then by a union bound,

$$\mathbb{P}\left(\forall i \in [T] \quad \pi_i(s_i) \in \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)\right) \geq \frac{1}{2}. \tag{6}$$

This implies that $\pi \in \Pi(Q^k)$, which is the desired result.

It remains to show that (5) holds. We will actually prove the bound assuming that $s_i \in \mathcal{S}_i^{\text{opt}}$. This is still sufficient to imply (6) since one can inductively assume that previous actions are optimal. We can write (5) equivalently as

$$\mathbb{P}\left(\exists a_i \in \arg\max_{a_i \in \mathcal{A}} \hat{Q}_i^k(s_i, a_i) \;:\; a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)\right) \leq \frac{1}{2T}. \tag{7}$$

Let $a^*_{i:i+k-1} \in \arg\max_{a_{i:i+k-1} \in \mathcal{A}^k} Q_i^1(s_i, a_{i:i+k-1})$ be chosen arbitrarily. By (4), this implies that $a_i^* \in \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)$. Then

$$\begin{aligned}
&\mathbb{P}\left(\exists a_i \in \arg\max_{a_i \in \mathcal{A}} \hat{Q}_i^k(s_i, a_i) \;:\; a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)\right) \\
&\quad \leq \mathbb{P}\left(\exists a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i) \;:\; \hat{Q}_i^k(s_i, a_i) \geq \hat{Q}_i^k(s_i, a_i^*)\right) \\
&\quad \leq \mathbb{P}\left(\exists a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i), a_{i+1:i+k-1} \in \mathcal{A}^{k-1} \;:\; \hat{Q}_i(s_i, a_i, a_{i+1:k-1}) \geq \hat{Q}_i(s_i, a^*_{i:i+k-1})\right) \\
&\quad \leq \sum_{a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i), a_{i+1:i+k-1} \in \mathcal{A}^{k-1}} \mathbb{P}\left(\hat{Q}_i(s_i, a_i, a_{i+1:k-1}) \geq \hat{Q}_i(s_i, a^*_{i:i+k-1})\right). \qquad (8)
\end{aligned}$$

Consider a single term of the sum in (8). By the definition of the gap and the fact that $a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)$, we know that

$$Q_i^k(s_i, a_i) \leq Q_i^k(s_i, a_i^*) - \Delta_i^k(s_i).$$

Combining this with (4) implies that

$$\begin{aligned}
Q_i^1(s_i, a_{i:i+k-1}) &\leq Q_i^1(s_i, a^*_{i:i+k-1}) - \Delta_i^k(s_i) \\
Q_i^1(s_i, a^*_{i:i+k-1}) - Q_i^1(s_i, a_{i:i+k-1}) &\geq \Delta_i^k(s_i). \qquad (9)
\end{aligned}$$

Now, consider the random variables $\hat{Q}_i(s_i, a_i, a_{i+1:k-1})$ and $\hat{Q}_i(s_i, a^*_{i:i+k-1})$. Let $X_j = \sum_{t=i}^T \gamma^{t-i} R(s_t^j, a_t^j)$ be the discounted reward from timestep i in the jth episode used to estimate $\hat{Q}_i(s_i, a_i, a_{i+1:i+k-1})$; let Y_j be the equivalent for estimating $\hat{Q}_i(s_i a^*_{i:i+k-1})$. Then we can write

$$\hat{Q}_i(s_i, a_i, a_{i+1:i+k-1}) = \frac{1}{m} \sum_{j=1}^m X_j \qquad \hat{Q}_i(s_i, a^*_{i:i+k-1}) = \frac{1}{m} \sum_{j=1}^m Y_j.$$

By the definition of the optimal value function V^* and the assumption that all rewards are nonnegative, we have that $X_j, Y_j \in [0, V_i^*(s_i)]$. We also know that the expectations of the X_j and Y_j are bounded:

$$\mathbb{E}[X_j] = Q_i^1(s_i, a_{i:i+k-1}) \leq Q_i^k(s_i, a_i) \leq \max_{a_i} Q_i^k(s_i, a_i).$$

$$\mathbb{E}[Y_j] = Q_i^1(s_i, a^*_{i:i+k-1}) = Q_i^k(s_i, a_i^*) \leq \max_{a_i} Q_i^k(s_i, a_i).$$

The variance of a random variable with mean μ bounded on an interval $[\alpha, \beta]$ is at most $(\beta - \mu)(\mu - \alpha)$. This means we can bound the variance of X_j and Y_j as well:

$$\operatorname{Var}(X_j) \leq \max_{a_i} Q_i^k(s_i, a_i) V_i^*(s_i) \qquad \operatorname{Var}(Y_j) \leq \max_{a_i} Q_i^k(s_i, a_i) V_i^*(s_i).$$

Now define

$$Z = \hat{Q}_i^1(s_i, a^*_{i:i+k-1}) - \hat{Q}_i^1(s_i, a_i, a_{i+1:i+k-1}) = \frac{1}{m} \sum_{j=1}^m Y_j - X_j = \frac{1}{m} \sum_{j=1}^m Z_j,$$

where $Z_j = Y_j - X_j$. Since X_j and Y_j are independent,

$$\operatorname{Var}(Z_j) = \operatorname{Var}(X_j) + \operatorname{Var}(Y_j) \leq 2 \max_{a_i} Q_i^k(s_i, a_i) V_i^*(s_i).$$

Also define centered versions $\bar{Z}_j = Z_j - \mathbb{E}[Z_j]$ and $\bar{Z} = Z - \mathbb{E}[Z]$. By (9), $\mathbb{E}[Z] \geq \Delta_i^k(s_i)$. Furthermore, since $X_j, Y_j \in [0, V_i^*(s_i)]$, we also know that $|Z_j| \leq V_i^*(s_i)$ and $|\bar{Z}_j| \leq V_i^*(s_i) + \mathbb{E}[Z] \leq 2V_i^*(s_i)$.

We can now finally apply Bernstein's inequality to bound the probability of one term in (8):

$$\begin{aligned}
&\mathbb{P}\left(\hat{Q}_i(s_i, a_i, a_{i+1:k-1}) \geq \hat{Q}_i(s_i, a^*_{i:i+k-1})\right) \\
&\quad = \mathbb{P}\left(Z \leq 0\right) \\
&\quad = \mathbb{P}\left(\bar{Z} \leq -\mathbb{E}[Z]\right) \\
&\quad \leq \mathbb{P}\left(\bar{Z} \leq -\Delta_i^k(s_i)\right) \\
&\quad \overset{\text{(i)}}{\leq} \exp\left\{\frac{-\frac{1}{2} m \Delta_i^k(s_i)^2}{\operatorname{Var}(Z_j) + \frac{2}{3} V_i^*(s_i) \Delta_i^k(s_i)}\right\} \\
&\quad \overset{\text{(ii)}}{\leq} \exp\left\{\frac{-\frac{1}{2} m \Delta_i^k(s_i)^2}{2 \max_{a_i} Q_i^k(s_i, a_i) V_i^*(s_i) + \frac{2}{3} V_i^*(s_i) \Delta_i^k(s_i)}\right\} \\
&\quad \overset{\text{(iii)}}{\leq} \exp\left\{\frac{-m \Delta_i^k(s_i)^2}{6 \max_{a_i} Q_i^k(s_i, a_i) V_i^*(s_i)}\right\} \\
&\quad \overset{\text{(iv)}}{\leq} \exp\left(-\log(2TA^k)\right) \\
&\quad = \frac{1}{2TA^k}.
\end{aligned} \tag{10}$$

Here, (i) is a direct application of Bernstein's inequality to the sum $\bar{Z} = \frac{1}{m} \sum_{j=1}^m \bar{Z}_j$. (ii) uses the bound on $\operatorname{Var}(Z_j) = \operatorname{Var}(\bar{Z}_j)$ and (iii) uses the fact that $\Delta_i^k(s_i) \leq \max_{a_i} Q_i^k(s_i, a_i)$ by definition of the gap Δ_i^k. Finally, (iv) uses the definition of m; since $s_i \in \mathcal{S}_i^{\text{opt}}$ by assumption,

$$m \geq \log\left(2TA^k\right) \max_{a_i \in \mathcal{A}} \frac{6 Q_t^k(s_i, a_i) V_t^*(s_i)}{\Delta_t^k(s_i)^2}.$$

Applying (10) to (8) gives

$$\begin{aligned}
&\mathbb{P}\left(\exists a_i \in \arg\max_{a_i \in \mathcal{A}} \hat{Q}_i^k(s_i, a_i) \; : \; a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i)\right) \\
&\quad \leq \sum_{a_i \notin \arg\max_{a_i \in \mathcal{A}} Q_i^k(s_i, a_i), a_{i+1:i+k-1} \in \mathcal{A}^{k-1}} \frac{1}{2TA^k} \\
&\quad \leq \frac{1}{2T},
\end{aligned}$$

which is the only thing left that is needed to complete the proof. ■

B Additional theoretical results concerning the effective horizon

In this appendix, we present some additional theoretical results concerning the effective horizon. First, we explore two algorithms—one in the style of policy gradient and one similar to fitted Q-iteration—whose sample complexities can also be bounded by a quantity related to the effective horizon. Then we show conditions under which the effective horizon is small, as well as information-theoretic lower bounds for the sample complexity of RL in terms of the effective horizon.

B.1 PG-GORP and FQI-GORP

The two algorithms we introduce in this section, PG-GORP and FQI-GORP, can be viewed as a bridge between GORP, which we use to define the effective horizon, and PPO and DQN, the deep RL algorithms whose performance we predict using the effective horizon in Section 6. They help to explain why the effective horizon is not only useful for understanding the performance of GORP, but also other RL algorithms.

We will actually give sample complexity bounds on PG-GORP and FQI-GORP in terms of the bound in Theorem 5.4, rather than the effective horizon itself. Supposing that the MDP is k-QVI-solvable, define

$$\bar{H}_k = k + \max_{t \in [T], s \in \mathcal{S}_i^{\text{opt}}, a \in \mathcal{A}} \log_A \left(\frac{Q_t^k(s, a) V_t^*(s)}{\Delta_t^k(s)^2} \right) + \log_A 6 \log \left(2 T A^k \right).$$

Our bounds will also depend on a quantity measuring how far the exploration policy π^{expl} is from the uniformly random policy π^{rand}:

$$\left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty = \max_{(t,s,a) \in [T] \times \mathcal{S} \times \mathcal{A}} \frac{1}{A \, \pi_t^{\text{expl}}(a \mid s)}.$$

$\left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty = 1$ in the case when $\pi^{\text{expl}} = \pi^{\text{rand}}$, and increases as the smallest probabilities π^{expl} assigns to actions becomes smaller.

We now introduce the first algorithm, PG-GORP.

Algorithm 2 The PG-GORP algorithm.

```
 1: procedure PG-GORP($m$)
 2:     $\pi \leftarrow \pi^{\text{expl}}$.
 3:     for $i = 1, \dots, T$ do
 4:         Sample $m$ episodes following $\pi$.
 5:         for $a \in \mathcal{A}$ do
 6:             $\hat{\nabla}_i(a \mid s_i) \leftarrow \frac{1}{m} \sum_{j=1}^m \frac{\mathbf{1}_{a_i^j = a}}{\pi_i(a \mid s_i)} \sum_{t=i}^T \gamma^{t-i} R(s_t^j, a_t^j)$.
 7:         end for
 8:         $\pi_i(s_i) \leftarrow \arg\max_{a \in \mathcal{A}} \hat{\nabla}_i(a \mid s_i)$.
 9:     end for
10:     return $\pi$
11: end procedure
```

PG-GORP resembles the REINFORCE algorithm [48], which gave rise to other policy gradient-based algorithms like PPO. At each iteration, Algorithm 2 first samples a number of episodes following its current policy (line 4). Then, it computes a an approximate gradient over the policy parameters—in this case, just the action probabilities $\pi_i(a \mid s_i)$—via the so-called "policy gradient theorem," which states

$$\begin{aligned} \nabla_{\pi_i(\cdot \mid s_i)} J(\pi) &\approx \frac{1}{m} \sum_{j=1}^m \nabla_{\pi_i(\cdot \mid s_i)} \log \pi_i(a_i^j \mid s_i) \sum_{t=i}^T \gamma^{t-i} R(s_t^j, a_t^j) \\ &= \frac{1}{m} \sum_{j=1}^m \frac{\nabla_{\pi_i(\cdot \mid s_i)} \pi_i(a_i^j \mid s_i)}{\pi_i(a_i^j \mid s_i)} \sum_{t=i}^T \gamma^{t-i} R(s_t^j, a_t^j) \\ &= \hat{\nabla}_i(\cdot \mid s_i). \end{aligned}$$

Then, in line 8, Algorithm 2 applies optimization to π based on its estimate of the gradient. In this case, it optimizes until π_i assigns all probability to only one action $\pi_i(s_i)$ at s_i.

Theorem B.1 (Sample complexity of PG-GORP)**.** *Suppose that an MDP is 1-QVI-solvable and that all rewards are nonnegative. Then the sample complexity of PG-GORP is at most*

$$2T^2 A^{\bar{H}_1} \left\| \frac{\pi^{\textit{rand}}}{\pi^{\textit{expl}}} \right\|_\infty.$$

Proof. Let

$$m = A^{\bar{H}_1} \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty = 12 A \log(2TA) \max_{t \in [T], s \in \mathcal{S}_i^{\text{opt}}, a \in \mathcal{A}} \left(\frac{Q_t^k(s,a) V_t^*(s)}{\Delta_t^1(s)^2} \right) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty .$$

We will show that Algorithm 2 converges with probability at least $1/2$ with this choice of parameter, giving the sample complexity bound in the theorem since the algorithm clearly samples $T^2 m$ total timesteps from the environment.

Similarly to the proof of Theorem 5.4, we will prove this by showing that with probability at least $1 - 1/(2T)$, at each iteration $\pi_i(s_i) \in \arg\max_{a \in \mathcal{A}} Q_i^{\pi^{\text{expl}}}(s_i, a)$. This gives $\pi \in \Pi(Q^1)$, which by 1-QVI-solvability means π must be optimal.

Consider the ith iteration of Algorithm 2. Define for each $a \in \mathcal{A}$ and $j \in [m]$ the random variable

$$X_j(a) = \frac{\mathbf{1}_{a_i^j = a}}{\pi_i(a \mid s_i)} \sum_{t=i}^{T} \gamma^{t-i} R(s_t^j, a_t^j).$$

First, can see that

$$0 \leq X_j(a) \leq \frac{V_i^*(s_i)}{\pi_i(a \mid s_i)} = \frac{V_i^*(s_i)}{\pi_i^{\text{expl}}(a \mid s_i)} \leq A \, V_i^*(s_i) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty ,$$

since $\pi_i = \pi_i^{\text{expl}}$ until line 8.

Second, we have

$$\mathbb{E}\left[X_j(a)\right] = \frac{1}{\pi_i(a \mid s_i)} \mathbb{P}\left(a_i^j = a\right) \mathbb{E}\left[\sum_{t=i}^{T} \gamma^{t-i} R(s_t^j, a_t^j) \mid a_i^j = a\right] = Q_i^{\pi^{\text{expl}}}(s_i, a).$$

Finally, using same the reasoning as in the proof of Theorem 2, we can bound the variance of $X_j(a)$:

$$\operatorname{Var}\left(X_j\right) \leq A \, V_i^*(s_i) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty \mathbb{E}\left[X_j\right] = A \, Q_i^{\pi^{\text{expl}}}(s_i, a) V_i^*(s_i) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty .$$

We now apply Bernstein's inequality to

$$\hat{\nabla}_i(a \mid s_i) = \frac{1}{m} \sum_{j=1}^{m} X_j(a)$$

for each $a \in \mathcal{A}$. If $a \in \arg\max_{a \in \mathcal{A}} Q_i^{\pi^{\text{expl}}}(s_i, a)$, we apply a lower tail bound:

$$\begin{aligned}
&\mathbb{P}\left(\hat{\nabla}_i(a \mid s_i) \leq Q_i^{\pi^{\text{expl}}}(s_i, a) - \frac{1}{2}\Delta_i^1(s_i)\right) \\
&\quad \leq \exp\left\{ - \frac{m \left(\Delta_i^1(s_i)\right)^2 / 8}{\left(Q_i^{\pi^{\text{expl}}}(s_i, a) + \frac{1}{3}\Delta_i^1(s_i)\right) V_i^*(s_i) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty} \right\} \\
&\quad \leq \exp\left\{ - \frac{m \left(\Delta_i^1(s_i)\right)^2}{12 Q_i^{\pi^{\text{expl}}}(s_i, a) V_i^*(s_i) \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty} \right\} \\
&\quad \leq \frac{1}{2TA}.
\end{aligned}$$

If $a \notin \arg\max_{a \in \mathcal{A}} Q_i^{\pi^{\text{expl}}}(s_i, a)$, we apply an identical upper tail bound:

$$\mathbb{P}\left(\hat{\nabla}_i(a \mid s_i) \geq Q_i^{\pi^{\text{expl}}}(s_i, a) + \frac{1}{2}\Delta_i^1(s_i)\right) \leq \frac{1}{2TA}.$$

These tail bounds hold simultaneously for all actions with probability at least $1 - \frac{1}{2T}$. Furthermore, assuming they hold and using the definition of the gap $\Delta_i^1(s_i)$, it must be that

$$\arg\max_{a \in \mathcal{A}} \hat{\nabla}_i(a \mid s_i) \subseteq \arg\max_{a \in \mathcal{A}} Q_i^{\pi^{\text{expl}}}(s_i, a),$$

which is enough to show that $\pi_i(s_i) \in \arg\max_{a \in \mathcal{A}} Q_i^{\pi^{\text{expl}}}(s_i, a)$ with probability at least $1 - \frac{1}{2T}$ and thus prove the theorem. ■

Now that we have seen that PG-GORP enjoys similar sample complexity bounds to GORP in the common case that an MDP is 1-QVI-solvable, we introduce FQI-GORP. FQI-GORP derives its name from fitted Q-iteration (FQI), which was originally proposed by Ernst et al. [49]. DQN was inspired by neural FQI [50], so FQI-GORP provides a natural connection to DQN.

Algorithm 3 The FQI-GORP algorithm.

```
 1: procedure FQI-GORP($k, m$)
 2:     for $i = 1, \dots, T$ do
 3:         Sample $A^k m$ episodes, following $\pi$ for timesteps 1 to $i - 1$ and then $\pi^{\text{expl}}$.
 4:         $\hat{Q}^1_{i+k-1} \leftarrow \arg\min_{\hat{Q}^1_{i+k-1}} \frac{1}{A^k m} \sum_{j=1}^{A^k m} \left( \hat{Q}^1_{i+k-1}(s^j_{i+k-1}, a^j_{i+k-1}) - \sum_{t=i+k-1}^{T} R(s^j_t, a^j_t) \right)^2$.
 5:         for $t = i + k - 2, \dots, i$ do
 6:             $\hat{Q}^{i+k-t}_t \leftarrow \arg\min_{\hat{Q}^{i+k-t}_t} \frac{1}{A^k m} \sum_{j=1}^{A^k m} \left( \hat{Q}^{i+k-t}_t(s^j_t, a^j_t) - R(s^j_t, a^j_t) - \max_{a' \in \mathcal{A}} \hat{Q}^{i+k-t-1}_{t+1}(s^j_{t+1}, a') \right)^2$.
 7:         end for
 8:         $\pi_i(s_i) \leftarrow \arg\max_{a \in \mathcal{A}} \hat{Q}^k_i(s_i, a)$.
 9:     end for
10:     return $\pi$
11: end procedure
```

FQI-GORP iteratively constructs a series of Q-functions at each iteration by minimizing a mean-squared temporal difference error loss, similar to FQI and DQN.

Theorem B.2 (Sample complexity of FQI-GORP)**.** *Suppose that an MDP is k-QVI-solvable and that all rewards are nonnegative. Then the sample complexity of FQI-GORP is at most*

$$T^2 \left\| \frac{\pi^{\textit{rand}}}{\pi^{\textit{expl}}} \right\|_\infty^k \max \left\{ 4A^{\bar{H}_k}, 10 \log(4TA^k) \right\}.$$

Proof. Let

$$m = \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty^k \max \left\{ 24 \log(2TA) \max_{t \in [T], s \in \mathcal{S}^{\text{opt}}_i, a \in \mathcal{A}} \left(\frac{Q^k_t(s,a) V^*_t(s)}{\Delta^1_t(s)^2} \right), \frac{10 \log(4TA^k)}{A^k} \right\}.$$

We will show that FQI-GORP with parameters k and m will return an optimal policy with probability at least $1/2$. Since FQI-GORP samples a number of timesteps from the environment equal to $T^2 A^k m$, this will prove the bound in the theorem.

Consider the ith iteration of Algorithm 2. We will show that with probability at least $1 - 1/(2T)$, for every $s_{i+k-1} \in \mathcal{S}$ reachable at timestep $i + k - 1$ starting from s_i, and for every action a_{i+k-1}

$$\left| \hat{Q}^1_{i+k-1}(s^j_{i+k-1}, a^j_{i+k-1}) - Q^1_{i+k-1}(s^j_{i+k-1}, a^j_{i+k-1}) \right| < \frac{\Delta^1_i(s_i)}{2}. \tag{11}$$

To prove (11), it is first helpful to write an explicit formula fitted Q-value, assuming that the loss in line 4 of Algorithm 3 is minimized:

$$\hat{Q}^1_{i+k-1}(s,a) = \frac{\sum_{j=1}^{A^k m} \mathbf{1}_{s^j_{i+k-1} = s \wedge a^j_{i+k-1} = a} \sum_{t=i+k-1}^{T} R(s^j_t, a^j_t)}{\sum_{j=1}^{A^k m} \mathbf{1}_{s^j_{i+k-1} = s \wedge a^j_{i+k-1} = a}}.$$

That is, the Q-value is a simple average of several reward-to-go values, each of which has expectation $Q^1_{i+k-1}(s,a)$. The probability of reaching some reachable state-action pair (s,a) at timestep $i+k-1$ must be at least $A^{-k} \left\| \frac{\pi^{\text{rand}}}{\pi^{\text{expl}}} \right\|_\infty^{-k}$. Thus, we can bound the sample size below via a concentration inequality for binomial random variables:

$$\mathbb{P} \left(\sum_{j=1}^{A^k m} \mathbf{1}_{s^j_{i+k-1} = s \wedge a^j_{i+k-1} = a} < 12 \log(2TA) \max_{a_i \in \mathcal{A}} \frac{Q^k_i(s_i, a_i) V^*_i(s_i)}{\Delta^1_i(s_i)^2} \right) \leq \exp\{ \frac{-3(10 \log(4TA^k))}{28} \} \leq \frac{1}{4TA^k}.$$

If the sample size is at least $12 \log(2TA) \max_{a_i \in \mathcal{A}} \frac{Q_i^k(s_i, a_i) V_i^*(s_i)}{\Delta_i^1(s_i)^2}$, then Bernstein's inequality (as applied in the proofs of Theorems 5.4 and B.1) gives

$$\mathbb{P}\left(\left|\hat{Q}^1_{i+k-1}(s^j_{i+k-1}, a^j_{i+k-1}) - Q^1_{i+k-1}(s^j_{i+k-1}, a^j_{i+k-1})\right| \geq \frac{\Delta_i^1(s_i)}{2}\right) \leq \frac{1}{4TA^k}.$$

Thus, taking a union bound, we have that the probability (11) does *not* hold for some reachable state-action pair at timestep $i+k-1$ must be at most $1/(2T)$, since there can be at most A^k such pairs.

We will now show by induction that given that (11) holds for all reachable state-action pairs,

$$\left|\hat{Q}_t^{i+k-t}(s^j_{i+k-1}, a^j_{i+k-1}) - \hat{Q}_t^{i+k-t} i + k - 1(s^j_{i+k-1}, a^j_{i+k-1})\right| < \frac{\Delta_i^1(s_i)}{2}. \tag{12}$$

holds for $t = i+k-1, \dots, i$ for all reachable state-action pairs at t. The base case of $t = i+k-1$ is already taken care of, so we only need to show the inductive step.

Assume (12) holds for all reachable state-action pairs at $t+1$. We can write an explicit formula for the fitted Q-values at timestep t, given that the loss function on line 6 of Algorithm 3 is minimized:

$$\begin{aligned}\hat{Q}_t^{i+k-t}(s, a) &= \frac{\sum_{j=1}^{A^k m} \mathbf{1}_{s_t^j = s \wedge a_t^j = a} \left(R(s_t^j, a_t^j) + \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^{i+k-t-1}(s_{t+1}^j, a')\right)}{\sum_{j=1}^{A^k m} \mathbf{1}_{s_t^j = s \wedge a_t^j = a}} \\ &= R(s, a) + \max_{a' \in \mathcal{A}} \hat{Q}_{t+1}^{i+k-t-1}(f(s, a), a').\end{aligned}$$

Given that (12) holds for each pair $(f(s,a), a')$, it is now easy to see that (12) must hold for t as well.

By induction (12) must hold for $t = i$ with probability at least $1 - 1/(2T)$. Given that it holds, and by definition of the gap, this implies that $\pi_i(s_i) \in \arg\max_{a \in \mathcal{A}} Q_i^k(s_i, a)$. Thus with probability at least $1/2$, $\pi \in \Pi(Q^k)$. By the assumption that the MDP is k-QVI solvable, π must be optimal with probability at least $1/2$. ■

B.2 Goal MDPs

Now, we will prove bounds on the effective horizon in one particular class of MDPs: goal MDPs.

Definition B.3 (Goal MDP). *An MDP is considered a* goal MDP *if there is some set of goal states $\mathcal{S}_{\textit{goal}}$ which are absorbing, i.e., $f(s, a) = s$ for every $s \in \mathcal{S}_{\textit{goal}}$, and furthermore the reward function is of the form*

$$R(s, a) = \begin{cases} 1 & s \notin \mathcal{S}_{\textit{goal}} \wedge f(s, a) \in \mathcal{S}_{\textit{goal}} \\ 0 & \textit{otherwise.} \end{cases}$$

That is, in a goal MDP reward is only received for reaching some set of goal states; the total episode reward is 1 if a goal state is reached and 0 otherwise. As an example, all of the Minigrid environments in BRIDGE are goal MDPs. We can show the following bound on the effective horizon in goal MDPs.

Theorem B.4 (The effective horizon in goal MDPs). *Suppose that $\pi^{\textit{expl}}(a \mid s) > 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. Then any goal MDP is 1-QVI-solvable. Furthermore, suppose that there is some $p > 0$ such that, for all timesteps $t \in [T]$ and all state-action pairs s_t, a_t at that timestep from which a goal state can be reached,*

$$\mathbb{P}_{\pi^{\textit{expl}}}\left(s_T \in \mathcal{S}_{\textit{goal}} \mid s_t, a_t\right) \geq p.$$

Then the effective horizon can be bounded as

$$H \leq 1 + \log_A \frac{\log(2T)}{p}. \tag{13}$$

Before we see the proof, note that Theorem B.4 agrees with our intuition that it should be harder to find an optimal policy for a goal MDP when it is less likely that the exploration policy reaches the goal, i.e., when p is smaller.

For instance, consider the MDP in Figure 2a. In this MDP, the minimum probability of reaching the goal with the random exploration policy after taking some action is exponentially small: $p = 1/A^{T-1}$. Applying Theorem B.4 gives a bound of $H \leq T + \log_A \log(2T)$.

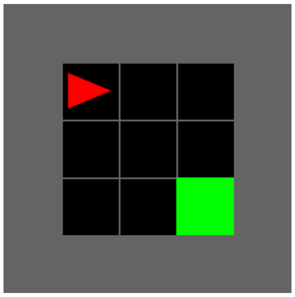


Figure 4: Empty-5x5, one of the Minigrid MDPs from BRIDGE and an example of a goal MDP (Definition B.3). The agent (red triangle) can turn left, turn right, or go forward, and its goal is to reach the green square, which gives a reward of 1.

In contrast, consider the Minigrid gridworld in Figure 4 from BRIDGE. Here we can bound $p \approx 0.00137$, which gives $H \leq 1 + \log_A(1/p) + \log_A \log(2T) = 1 + \log_3 729 + \log_3 \log(200) \leq 7 + 1.52 = 8.52 \ll T = 100$. The techniques in Appendix C give a much tighter bound of $H \leq 1.64$.

Proof. We can assume that $V_1^*(s_1) = 1$, since otherwise every trajectory in the MDP gives reward 0 and thus the effective horizon is trivially bounded. First, we will show that

$$Q_t^{\pi^{\text{expl}}}(s,a) > 0 \qquad \Leftrightarrow \qquad Q_t^*(s,a) = 1. \tag{14}$$

This is enough to imply the MDP is 1-QVI-solvable, since any policy in $\Pi(Q^{\pi^{\text{expl}}})$ must take only actions with $Q_t^{\pi^{\text{expl}}}(s,a) > 0$, which must be optimal according to (14).

To show that (14) holds, first consider the $\Rightarrow$ implication. Assume $Q_t^{\pi^{\text{expl}}}(s,a) > 0$ and by way of contradiction suppose that $Q_t^*(s,a) \neq 1$, which means that $Q_t^*(s,a) = 0$. Clearly this cannot happen since this would imply $Q_t^{\pi^{\text{expl}}}(s,a) \leq 0$. Now, consider the $\Leftarrow$ direction. If $Q_t^*(s,a) = 1$, then there must be some sequence of actions starting with a which leads from s to a goal state. By assumption, π^{expl} assigns positive probability to each action in this sequence. Thus $Q_t^{\pi^{\text{expl}}}(s,a) = \mathbb{P}_{\pi^{\text{expl}}}(s_T \in \mathcal{S}_{\text{goal}} \mid s,a) > 0$.

Next, we will prove the bound on H from (13). From (14), we can see that Algorithm 1 (GORP) will return an optimal policy for the MDP as long as at each iteration i it picks some action a_i with $Q_i^{\pi^{\text{expl}}}(s_i,a_i) > 0$. In turn, this will happen as long as $\hat{Q}_i^1(s_i,a_i) > 0$ for some such a_i, since $\hat{Q}_i^1(s_i,a)$ must be 0 for any suboptimal a.

Thus, we need only show that

$$\mathbb{P}\left(\exists a_i \in \mathcal{A} \quad \hat{Q}_i^1(s_i,a_i) > 0\right) \geq 1 - \frac{1}{2T} \tag{15}$$

holds for each i when $m \geq \log(2T)/p$. This will allow us to conclude via a union bound that Algorithm 1 will find an optimal policy with probability at least 1/2 in this case, which gives the desired bound on the effective horizon.

To show (15), consider iteration i of Algorithm 1 and let $a_i \in \mathcal{A}$ such that $Q_i^{\pi^{\text{rand}}}(s_i,a_i) > 0$. We can assume that such an action exists as long as Algorithm 1 has succeeded in iterations previous to i. Let $X_j = \sum_{t=i}^{T} \gamma^{t-i} R(s_t^j, a_t^j)$ be the reward-to-go from the jth episode sampled to evaluate $\hat{Q}_i^1(s_i,a_i)$. By the definition of a goal MDP, each $X_j(a_i) \in \{0,1\}$. Furthermore, since $\hat{Q}_i^1(s_i,a_i) =$

$\frac{1}{m}\sum_{j=1}^{m} X_j$, then $\hat{Q}_i^1(s_i, a) > 0$ as long as some $X_j = 1$. This implies

$$\begin{aligned}
&\mathbb{P}\left(\exists a_i \in \mathcal{A} \quad \hat{Q}_i^1(s_i, a_i) > 0\right) \\
&\quad \geq \mathbb{P}\left(\hat{Q}_i^1(s_i, a_i) > 0\right) \\
&\quad = \mathbb{P}\Big(\exists j \in [m] \quad X_j = 1\Big) \\
&\quad = 1 - \mathbb{P}\Big(\forall j \in [m] \quad X_j = 0\Big) \\
&\quad = 1 - \mathbb{P}\Big(X_j = 0\Big)^m \\
&\quad \overset{\text{(i)}}{\geq} 1 - (1-p)^m \\
&\quad \geq 1 - \exp(-mp) \\
&\quad \geq 1 - \frac{1}{2T},
\end{aligned}$$

the bound previously proposed in (15) which we argued gives the desired bound. (i) uses the assumption that $Q_i^{\pi^{\text{expl}}}(s_i, a_i) = \mathbb{P}(X_j = 1) \geq p$. $\blacksquare$

B.3 Lower bounds

Next, we will show that there are MDPs in which exponential dependence on the effective horizon is unavoidable. That is, in some cases there is an information-theoretic lower bound on the sample complexity of RL proportional to A^H.

Theorem B.5. *Fix $T \geq 1$, $A \geq 2$, and $H \in [T]$. Then for any RL algorithm, there is an MDP with A actions, horizon T, and effective horizon at most H for which the algorithm's sample complexity is at least*

$$T\lfloor T/H \rfloor \left(\lceil A^H/2 \rceil - 1\right) = \Omega(T^2 A^H / H).$$

Note that this matches the upper bound on the sample complexity of GORP given in Lemma 5.3 up to a factor of roughly $2H$. When $H = T$, it exactly agrees with the lower bound given in Theorem 2.1.

Proof. The proof uses MDPs with the same state space and transition function as in Theorem 2.1. Define A^T such MDPs which differ only in their reward functions:

$$\mathbb{M} = \{\mathcal{M}_{a_{1:T}} \mid a_{1:T} \in \mathcal{A}^T\} \quad \text{where the MDP } \mathcal{M}_{a_{1:T}} \text{ has } R(s_{a'_{1:\ell}}, a') = \begin{cases} 1 & a'_{1:\ell}, a' = a_{1:\ell+1} \text{ and } \ell \equiv 0 \pmod{H} \\ 0 & \text{otherwise.} \end{cases}$$

That is, each MDP has a single optimal sequence of actions that gives reward 1 every H timesteps.

By the same argument as in the proof of Theorem 2.1, for any RL algorithm there must be some MDP in $\mathbb{M}$ such that after interacting with the environment for less than $\lceil A^H/2 \rceil - 1$ episodes, the algorithm cannot with probability at least $1/2$ identify the optimal actions for timesteps $t = 1$ to $t = H$. We can repeat this line of reasoning for timesteps $t = H + 1$ to $t = 2H$, and so on for a total of $\lfloor T/H \rfloor$ steps, to show that with less than $\lfloor T/H \rfloor \left(\lceil A^H/2 \rceil - 1\right)$ episodes, there must be some MDP in $\mathbb{M}$ whose optimal action sequence cannot be identified with probability greater than $(1/2)^{\lfloor T/H \rfloor} \leq 1/2$. Thus, the sample complexity of the RL algorithm on this MDP must be at least

$$T\lfloor T/H \rfloor \left(\lceil A^H/2 \rceil - 1\right),$$

which is the desired bound.

It only remains to be shown that the effective horizon of the MDPs in $\mathbb{M}$ is actually H. To see why, consider running Algorithm 1 with $k = H$ and $m = 1$. That is, at each iteration i, GORP will try all H-length action sequences followed by actions from π^{expl}. Then, it will pick the action sequence with the highest empirical reward-to-go. From the definition of the MDPs in $\mathbb{M}$, all action sequences starting with a suboptimal action must have empirical reward-to-go of 0. Furthermore, at least one H-length action sequence starting with an optimal action must get reward-to-go of at least 1. Thus, GORP will with probability 1 choose an optimal action at each timestep. This means that the effective horizon must be at most $H + \log_A 1 = H$. $\blacksquare$

C Tighter bounds on the effective horizon

In Theorem 5.4, we obtained bounds on the effective horizon and thus on the sample complexity of GORP. However, we find that the bounds given by Theorem 5.4 are often very loose compared to the empirical performance of GORP due to two factors. First, Theorem 5.4 requires considering the worst case of $Q_t^k(s,a)V_t^*(s)/\Delta_t^k(s)^2$ over all optimal states. However, in many MDPs in our dataset, there are optimal states with extremely small gaps that in practice are almost never reached by GORP. When Theorem 5.4 is applied, these states make the sample complexity bounds very large despite GORP working well empirically. Second, Theorem 5.4 uses asymptotically tight techniques for bounding the sample complexity that can be quite loose for small sample sizes. Below, we describe the algorithm we use to provably bound the sample complexity of GORP (and thus the effective horizon) that gives much tighter results.

Consider the GORP algorithm as given in Algorithm 1. Let a_t denote the random variable corresponding to the action ultimately chosen by the algorithm for timestep t. Let s_t denote the state reached by actions $a_1, \dots, a_{t-1}$. Denote by $\mathbb{P}_m$ the probability measure given by running the algorithm with parameter m. We would like to bound the probability that the algorithm does not achieve the optimal return in the MDP. Let this event by denoted as

$$\begin{aligned}\mathcal{E} &:= \sum_{t=1}^{T} R_t(s_t, a_t) < V_1^*(s_1) \\ &\Leftrightarrow \exists t \quad a_t \notin \mathcal{A}_t^*(s_t),\end{aligned}$$

where $\mathcal{A}_t^*(s)$ denotes the set of optimal actions in state s at timestep t, i.e.

$$\mathcal{A}_t^*(s) = \arg\max_{a \in \mathcal{A}} Q_t^*(s).$$

It is straightforward to see from Algorithm 1 that it requires $T^2 A^k m$ timesteps of interaction the environment. Thus the sample complexity of the algorithm is

$$T^2 A^k m \quad \text{where} \quad m = \min\{m \in \mathbb{N} \mid \mathbb{P}_m(\mathcal{E}) < 1/2\}.$$

Clearly, we can upper bound the sample complexity using any m such that the probability of failure is bounded as $\mathbb{P}_m(\mathcal{E}) < 1/2$. To do so, for each value of k, we perform a binary search over values of m from 1 to 10^{100}. For each possible m, we calculate an upper bound on $\mathbb{P}_m(\mathcal{E})$. If the upper bound is below $1/2$, we then search below m; if it is greater, we search above m. When the search has converged to a relative precision of $1/100$, we output $T^2 A^k m$ as the sample complexity and $H_k = k + \log_A m$ as the effective horizon for that particular value of k.

C.1 Upper bounding $\mathbb{P}_m(\mathcal{E})$

We upper bound the failure probability $\mathbb{P}_m(\mathcal{E})$ recursively. Let $\mathcal{O}_t$ denote the event that all actions taken before t have been optimal, i.e.,

$$\mathcal{O}_t := \forall t' < t \quad a_t \in \mathcal{A}_t^*(s_t).$$

To begin the recursion, note that at the final timestep,

$$\mathbb{P}(\mathcal{E} \mid \mathcal{O}_T, s_T, a_T) = \mathbf{1}\{a_T \notin \mathcal{A}_T^*(s_T)\}.$$

We will use two recursion rules: one from next states to state-action pairs and one from state-action pairs to states. The first rule is

$$\mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t) = \sum_{a_t \in \mathcal{A}} \mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t, a_t)\mathbb{P}(a_t \mid s_t) \tag{16}$$

and the second (for $t < T$) is

$$\mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t, a_t) = \begin{cases} 1 & a_t \notin \mathcal{A}_t^*(s_t) \\ \mathbb{P}(\mathcal{E} \mid \mathcal{O}_{t+1}, s_{t+1}) & a_t \in \mathcal{A}_t^*(s_t). \end{cases} \tag{17}$$

We apply these results recursively from $t = T, \dots, 1$ to finally obtain $\mathbb{P}(\mathcal{E} \mid \mathcal{O}_1, s_1) = \mathbb{P}(\mathcal{E})$.

The remaining difficulty is calculating $\mathbb{P}(a_t \mid s_t)$. Recall from Algorithm 1 that a_t is chosen as the first action of a k-action sequence

$$a_{t:t+k-1} \in \arg\max_{a_{t:t+k-1} \in \mathcal{A}^k} \hat{Q}_t(s_t, a_{t:t+k-1}),$$

where $\hat{Q}_t(s_t, a_{t:t+k-1})$ is the empirical mean return-to-go from m episodes starting in state s_t and taking actions $a_t, \ldots, a_{t+k-1}$ followed by actions sampled from the exploration policy. To simplify notation, let $\vec{a}_t$ denote $a_{t:t+k-1}$. We use various inequalities, described in detail below, to bound the probability that a particular k-action sequence is chosen:

$$\begin{aligned} \underline{p}(\vec{a}_t) &\leq \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) > \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \\ &\leq \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \geq \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \leq \overline{p}(\vec{a}_t). \end{aligned}$$

Letting $p(\vec{a}_t)$ denote the actual probability an action sequence is chosen, we can rewrite (16) to

$$\mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t) = \sum_{a_t \in \mathcal{A}} \mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t, a_t) \sum_{a_{t+1:t+k-1} \in \mathcal{A}^{k-1}} p(\vec{a}_t). \tag{18}$$

Given the bounds on $p(\vec{a}_t)$ (i.e., $\underline{p}$ and $\overline{p}$), we formulate a linear program with the bounds as constraints, plus the constraint that $\sum_{\vec{a}_t \in \mathcal{A}^k} p(\vec{a}_t) = 1$, with the objective of maximizing (18). Solving this gives an upper bound on $\mathbb{P}(\mathcal{E} \mid \mathcal{O}_t, s_t)$. We can then propagate this bound recursively using (17) to bound $\mathbb{P}(\mathcal{E})$.

Calculating $\underline{p}$ and $\overline{p}$

We use up to four methods to bound $p(\vec{a}_t)$, and pick the one which gives the lowest conditional failure probability after solving the linear program described above. As previously, let $\mathcal{D}_t(s_t, \vec{a}_t)$ denote the distribution of the reward-to-go starting in s_t and taking actions $a_t, \ldots, a_{t+k-1}$ followed by actions sampled from the exploration policy. Thus we can write

$$\hat{Q}_t(s_t, \vec{a}_t) = \frac{1}{m} \sum_{i=1}^{m} X_i \qquad \text{where} \quad X_1, \ldots, X_m \overset{\text{i.i.d.}}{\sim} \mathcal{D}_t(s_t, \vec{a}_t).$$

Most of the following methods use the following decomposition:

$$\begin{aligned} &\mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) > \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \\ &\overset{\text{(i)}}{=} \int \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) < \hat{Q}_t(s_t, \vec{a}_t) \right) \, d\mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \right) \\ &\overset{\text{(ii)}}{\geq} \sum_{i=1}^{N} \mathbb{P}_m \left(q_{i-1} < \hat{Q}_t(s_t, \vec{a}_t) \leq q_i \right) \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) \leq q_{i-1} \right) \end{aligned} \tag{19}$$

for some sequence $-\infty = q_0 \leq q_1 \leq \ldots \leq q_N = \infty$. Here, (i) uses the fact that the random variables $\hat{Q}_t(s_t, \vec{a}_t)$ across all action sequences $\vec{a}_t \in \mathcal{A}^k$ are independent. (ii) is a lower bound on the integral via a Riemann sum.

Alternatively, suppose we know that the CDF of $\hat{Q}_t(s_t, \vec{a}_t)$ is bounded by

$$\mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \leq x \right) \geq F_{\underline{Z}}(x)$$

where $F_{\underline{Z}}(x)$ is the continuous CDF of some random variable $\underline{Z}$. Then

$$\begin{aligned} &\mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) > \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \\ &\geq \mathbb{P}_m \left(\underline{Z} > \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \\ &= \int \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) < \hat{Q}_t(s_t, \vec{a}_t) \right) \, d\mathbb{P}_m \left(\underline{Z} \right) \\ &\geq \frac{1}{N} \sum_{i=1}^{N} \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) \leq F_{\underline{Z}}^{-1} \left(\frac{i-1}{N} \right) \right) \end{aligned} \tag{20}$$

by a similar argument. We also have equivalent bounds in the other direction:

$$\mathbb{P}_m\left(\hat{Q}_t(s_t,\vec{a}_t) \geq \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t,\vec{a}'_t)\right) \leq \sum_{i=1}^{N} \mathbb{P}_m\left(q_{i-1} < \hat{Q}_t(s_t,\vec{a}_t) \leq q_i\right) \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m\left(\hat{Q}(s_t,\vec{a}'_t) \leq q_i\right) \tag{21}$$

$$\mathbb{P}_m\left(\hat{Q}_t(s_t,\vec{a}_t) \geq \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t,\vec{a}'_t)\right) \leq \frac{1}{N} \sum_{i=1}^{N} \prod_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \mathbb{P}_m\left(\hat{Q}(s_t,\vec{a}'_t) \leq F_{\overline{Z}}^{-1}\left(\frac{i}{N}\right)\right) \tag{22}$$

where the CDF of $\overline{Z}$ is greater than or equal to that of $\hat{Q}_t(s_t,\vec{a}_t)$. We use $N = 100$ when using these bounds.

Binomial bounds In the case where for all $\vec{a}_t \in \mathcal{A}^k$, the distribution $\mathcal{D}_t(s_t,\vec{a}_t)$ has mass on only 0 and some other value C, we have

$$\hat{Q}_t(s_t,\vec{a}_t) \sim \frac{C}{m} \text{ Binom}\left(m, \frac{1}{C} Q_t(s_t,\vec{a}_t)\right).$$

This case occurs in many environments that are goal-based, i.e. where the agent gets reward only for reaching some goal and then the episode ends. We find that it significantly improves the sample complexity bounds in those environments. Without loss of generality, we may assume $C = 1$. We then apply (19) and (21) to obtain $\underline{p}(\vec{a}_t)$ and $\overline{p}(\vec{a}_t)$. We let $q_0, \dots, q_N$ be set such that

$$\mathbb{P}_m\left(q_{i-1} < \hat{Q}_t(s_t,\vec{a}_t) \leq q_i\right) \approx \frac{1}{N}$$

using either the exact inverse CDF of the binomial distribution for small m or a normal approximation for large m. Then, we can calculate all terms in the bounds (19) and (21) using the binomial CDF. Since the CDF is more expensive to calculate for larger m, we only use the binomial-based bounds when $m \leq 10^6$ and $k = 1$.

Berry-Esseen bounds For this type of bound, we calculate the variance $\sigma^2 = \text{Var}(\mathcal{D}_t(s_t,\vec{a}_t))$ and third absolute moment

$$\rho = \mathbb{E}_{X \sim \mathcal{D}_t(s_t,\vec{a}_t)}[|X|^3]$$

for each $\vec{a}_t \in \mathcal{A}^k$. Then by the Berry-Esseen theorem [51], we have that

$$\left|\mathbb{P}_m(\hat{Q}_t(s_t,\vec{a}_t) \leq u) - \mathbb{P}_{X \sim \mathcal{N}(Q_t(s_t,\vec{a}_t),\sigma^2/m)}(X \leq u)\right| \leq \frac{\min\{0.3328(\rho/\sigma^3 + 0.429), 0.33554(\rho/\sigma^3 + 0.415)\}}{\sqrt{m}}.$$

The resulting upper and lower bounds on the CDFs of $\hat{Q}_t(s_t,\vec{a}_t)$ for all $\vec{a}_t \in \mathcal{A}^k$ can be used in (20) and (22) to calculate $\underline{p}(\vec{a}_t)$ and $\overline{p}(\vec{a}_t)$. Since this bound requires order N evaluations of the normal CDF and inverse CDF, which is somewhat expensive, we only use it when $A^k \leq 100$.

Bernstein bounds Similarly to the Berry-Esseen bounds, Bernstein's inequality can be used to bound the CDF of $\hat{Q}_t(s_t,\vec{a}_t)$, and is sometimes superior to Berry-Esseen for large m due to giving tail bounds that decay exponentially rather than quadratically. In particular, suppose $\mathcal{D}_t(s_t,\vec{a}_t)$ is supported on the interval $[\alpha,\beta]$; we can compute these bounds via value iteration. Then

$$\text{Var}(\mathcal{D}_t(s_t,\vec{a}_t)) \leq V = (\beta - Q_t(s_t,\vec{a}_t))\,(Q_t(s_t,\vec{a}_t) - \alpha)\,.$$

Bernstein's inequality gives the following bounds on the CDF of $\hat{Q}_t(s_t,\vec{a}_t)$:

$$\mathbb{P}_m\left(\hat{Q}_t(s_t,\vec{a}_t) \leq Q_t(s_t,\vec{a}_t) + u\right) \geq \begin{cases} 1 & u \leq 0 \\ 1 - \exp\left\{-\frac{mu^2/2}{V + (\beta-\alpha)u/3}\right\} & \text{otherwise} \end{cases}$$

$$\mathbb{P}_m\left(\hat{Q}_t(s_t,\vec{a}_t) \leq Q_t(s_t,\vec{a}_t) + u\right) \leq \begin{cases} 1 & u \geq 0 \\ \exp\left\{-\frac{mu^2/2}{V + (\beta-\alpha)u/3}\right\} & \text{otherwise.} \end{cases}$$

Similarly to the Berry-Esseen bounds, we use these in (20) and (22) to calculate $\underline{p}(\vec{a}_t)$ and $\overline{p}(\vec{a}_t)$ when $A^k \leq 100$.

Bennett bounds The final method we use to calculate $\underline{p}(\vec{a}_t)$ and $\overline{p}(\vec{a}_t)$ is computationally cheaper than the others, so we can use it no matter the size of A^k. As in the Bernstein bounds, we calculate the interval support and bound the variance of each $\mathcal{D}_t(s_t, \vec{a}_t)$. We then let u be the arithematic mean of the highest action sequence Q value and the second-highest, i.e.

$$u = \frac{1}{2} \left(\max_{\vec{a}_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}_t) + \max_{\vec{a}'_t \notin \arg\max_{\vec{a}_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}_t)} Q_t(s_t, \vec{a}'_t) \right).$$

Then, we for each action sequence with less-than-highest Q-values, i.e. for each $\vec{a}_t \notin \arg\max_{\vec{a}_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}_t)$, we calculate the upper bound

$$\begin{aligned}
&\mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \geq \max_{\vec{a}'_t \in \mathcal{A}^k \setminus \{\vec{a}_t\}} \hat{Q}(s_t, \vec{a}'_t) \right) \\
&\leq \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \geq \max_{\vec{a}'_t \in \arg\max_{\vec{a}_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}_t)} \hat{Q}(s_t, \vec{a}'_t) \right) \\
&= 1 - \mathbb{P}_m \left(\exists \vec{a}'_t \in \arg\max_{\vec{a}'_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}'_t) \qquad \hat{Q}_t(s_t, \vec{a}_t) < \hat{Q}(s_t, \vec{a}'_t) \right) \\
&\leq 1 - \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) < u \qquad \wedge \qquad \forall \vec{a}'_t \in \arg\max_{\vec{a}'_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}'_t) \quad \hat{Q}(s_t, \vec{a}'_t) > u \right) \\
&= 1 - \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) < u \right) \prod_{\vec{a}'_t \in \arg\max_{\vec{a}'_t \in \mathcal{A}^k} \hat{Q}(s_t, \vec{a}'_t)} \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) > u \right) \\
&= 1 - \left(1 - \mathbb{P}_m \left(\hat{Q}_t(s_t, \vec{a}_t) \geq u \right) \right) \prod_{\vec{a}'_t \in \arg\max_{\vec{a}'_t \in \mathcal{A}^k} \hat{Q}(s_t, \vec{a}'_t)} \left(1 - \mathbb{P}_m \left(\hat{Q}(s_t, \vec{a}'_t) \leq u \right) \right). \qquad (23)
\end{aligned}$$

Each of the tail bounds in (23) can be upper bounded using Bennett's inequality to obtain $\overline{p}(\vec{a}_t)$. We let $\overline{p}(\vec{a}_t) = 1$ if $\vec{a}_t \in \arg\max_{\vec{a}_t \in \mathcal{A}^k} Q_t(s_t, \vec{a}_t)$ and we set $\underline{p}(\vec{a}_t) = 0$ for all $\vec{a}_t \in A^k$.

D Previously proposed sample complexity bounds

In this appendix, we give proofs of sample complexity bounds based on properties previously proposed in the RL theory literature. We also compare these bounds to our effective horizon-based bounds in examples that showcase their failure modes.

D.1 Upper confidence bounds (UCB) and strategic exploration

A central problem in RL is exploration: how to efficiently reach enough states in an MDP in order to identify the optimal policy. One common way of approaching exploration is with upper-confidence bounds (UCB), which originated in the bandit literature. Algorithms using UCBs generally choose actions based on the current best estimate of that action's value plus an exploration "bonus" that incentivizes exploration of little-seen states. Examples in the RL literature include Kakade [18], Azar et al. [4], Jiang et al. [19], Jin et al. [5, 20], Du et al. [21], Jin et al. [22]. Generally, these UCB algorithms achieve minimax sample complexity in terms of some measure of the "size" of the state space—either the number of states $\mathcal{S}$ [4], or quantities like the Bellman-Eluder dimension [22].

A very simple UCB-type algorithm, R-MAX [23, 18], achieves a sample complexity bounded by SAT in deterministic, tabular MDPs. It depends on knowing the maximum reward at any state-action pair in the MDP, $R_{\text{max}} = \max_{(s,a) \in \mathcal{S} \times \mathcal{A}} R(s, a)$. It also requires access to a computational oracle that can calculate an optimal policy for any transition function f and reward function R, for instance via value iteration.

```
procedure R-MAX
    initialize $\hat{f}(s, a) \leftarrow s$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$
    initialize $\hat{R}(s, a) \leftarrow R_{\text{max}}$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$
    for $j = 1, \ldots, SA$ do
        for $t = 1, \ldots, T$ do
            take an action $a$ in the current state $s$ according to the optimal policy for $\hat{f}$ and $\hat{R}$
            $\hat{R}(s, a) \leftarrow$ the observed reward
            $\hat{f}(s, a) \leftarrow$ the observed next state $s'$
        end for
    end for
    return an optimal policy for $\hat{f}$ and $\hat{R}$
end procedure
```

This is the version of R-MAX for deterministic MDPs; there is a more complex version for stochastic MDPs. The exploration bonuses in R-MAX are simply the initialization of $\hat{R}$ to the maximum possible reward. This ensures that an optimal value function computed from $\hat{f}$ and $\hat{R}$ is always an upper bound on the true optimal value function. The following result shows that R-MAX finds the optimal policy, and thus proves that its sample complexity is at most SAT.

Theorem D.1. R-MAX *returns an optimal policy.*

Proof. Let $\hat{V}^*$ and $\hat{Q}^*$ be the optimal value function and Q-function under $\hat{f}$ and $\hat{R}$. We will begin by showing that at any point in the algorithm, $V^*_t(s) \leq \hat{V}^*_t(s)$ and $Q^*_t(s,a) \leq \hat{Q}^*_t(s,a)$ for all $(t,s,a) \in [T] \times \mathcal{S} \times \mathcal{A}$.

The proof is via induction on t from T to 1. To begin, clearly $\hat{Q}^*_T(s,a) \in \{Q^*_T(s,a), R_{\text{max}}\}$, so the bound holds for $\hat{Q}^*_T$. Now, suppose that at some $t \in [T]$ and all $(s,a) \in \mathcal{S} \times \mathcal{A}$, $Q^*_t(s,a) \leq \hat{Q}^*_t(s,a)$. Then

$$\hat{V}^*_t(s) = \max_{a \in \mathcal{A}} \hat{Q}^*_t(s,a) \geq \max_{a \in \mathcal{A}} Q^*_t(s,a) = V^*_t(s),$$

so the bound holds for $\hat{V}^*_t$ as well. Finally, say that at some $t \in [T-1]$ and for all $s \in \mathcal{S}$, $V^*_{t+1}(s) \leq \hat{V}^*_{t+1}(s)$. To show this implies $Q^*_t(s,a) \leq \hat{Q}^*_t(s,a)$ for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, consider two cases. First, if (s,a) has been seen by the algorithm, then

$$\hat{Q}^*_t(s,a) = \hat{R}(s,a) + \hat{V}^*_{t+1}(\hat{f}(s,a) = R(s,a) + \hat{V}^*_{t+1}(f(s,a) \geq R(s,a) + V^*_{t+1}(f(s,a) = Q^*_t(s,a).$$

Otherwise, if (s,a) has not been seen, then $\hat{R}(s,a) = R_{\text{max}}$ and $\hat{f}(s,a) = s$. In this case,

$$\hat{Q}^*_t(s,a) = (T - t + 1) R_{\text{max}} \geq Q^*_t(s,a).$$

By induction we see that $V^*_t(s) \leq \hat{V}^*_t(s)$ and $Q^*_t(s,a) \leq \hat{Q}^*_t(s,a)$ for all $(t,s,a) \in [T] \times \mathcal{S} \times \mathcal{A}$.

Next, we will prove that any optimal policy π for $\hat{f}$ and $\hat{R}$ must either (a) be optimal for f and R or (b) reach a previously unseen state-action pair. In particular, we will show that if $V^\pi_1(s_1) < \hat{V}^*_1(s_1)$, then π must reach a previously unseen state-action pair. Otherwise, $V^\pi_1(s_1) \geq \hat{V}^*_1(s_1) \geq V^*_1(s_1)$, showing that π is optimal for f and R.

We will again work inductively starting from the last timestep. First, suppose that $Q^\pi_T(s,a) < \hat{Q}^*_T(s,a)$ for some $(s,a) \in \mathcal{S} \times \mathcal{A}$. This is equivalent to $R(s,a) < \hat{R}(s,a)$, which clearly means that (s,a) cannot have been explored.

Now, suppose that at some timestep $t \in [T]$, we know that for any $(s,a) \in \mathcal{S} \times \mathcal{A}$, $Q^\pi_t(s,a) < \hat{Q}^*_t(s,a)$ implies that π must explore some new state-action pair at or after timestep t starting in (s,a). If $V^\pi_t(s) < \hat{V}^*_t(s)$ for some $s \in \mathcal{S}$, then

$$Q^\pi_t(s, \pi_t(s)) < \max_{a \in \mathcal{A}} \hat{Q}^*_t(s,a) = \hat{Q}^*_t(s, \pi_t(s)).$$

By assumption this means π must explore some new state-action pair at or after timestep t, since it takes an action a satisfying $Q^\pi_t(s,a) < \hat{Q}^*_t(s,a)$.

Finally, suppose that for some $t \in [T]$, we know that for any $s \in \mathcal{S}$, $V^\pi_{t+1}(s) < \hat{V}^*_{t+1}(s)$ implies that π must explore some new state-action pair at or after timestep $t+1$ starting in s. Suppose for some $(s,a) \in \mathcal{S} \times \mathcal{A}$ that $Q^\pi_t(s,a) < \hat{Q}^*_t(s,a)$. Then

$$R(s,a) + V^\pi_{t+1}(f(s,a)) < \hat{R}(s,a) + V^\pi_{t+1}(\hat{f}(s,a))$$

which implies that either $R(s,a) < \hat{R}(s,a)$ or $V^\pi_{t+1}(f(s,a)) < V^\pi_{t+1}(\hat{f}(s,a))$. In the first case, (s,a) must be unexplored. In the second case, either (s,a) is unexplored, or

$$V^\pi_{t+1}(f(s,a)) < V^\pi_{t+1}(f(s,a)).$$

In any of these cases, π must explore a new state-action pair either in this timestep or in the future starting from (s,a).

Inductively, this shows that $V_1^\pi(s_1) < \hat{V}_1^*(s_1)$ implies that π must reach a previously unseen state-action pair. We will show that this property implies that R-MAX must return an optimal policy.

In particular, note that after the jth loop iteration in R-MAX, it must either have an optimal policy or have explored at least j of the state-action pairs in the MDP. This is a simple consequence of the above property: at each iteration, either the policy used by R-MAX must be optimal or it must explore at least one additional state-action pair. This means that after all the SA loop iterations, R-MAX will either have an optimal policy or have explored *all* the state-action pairs, in which case it will also have an optimal policy. ■

D.2 Covering length

The *covering length* of an MDP was originally proposed by Even-Dar and Mansour [52] and later used by Liu and Brunskill [7] to prove sample complexity bounds on RL algorithms which use random exploration. Liu and Brunskill [7] show various bounds on the covering length using graph-theoretic notions. While they focus on discounted infinite-horizon MDPs, we use a version of covering length adapted to finite-horizon MDPs, similar to that used by Dann et al. [6].

Definition D.2 (Covering length)**.** *The* covering length L *of an MDP under an exploration policy* $\pi^{\textit{expl}}$ *is the number of episodes needed until all state-action pairs have been visited with probability at least* $1/2$.

One can easily show sample complexity bounds based on the covering length.

Theorem D.3 (Covering length sample complexity bound)**.** *There is an RL algorithm which can solve any MDP with sample complexity* TL *given an exploration policy* $\pi^{\textit{expl}}$*, where* L *is the covering length of* $\pi^{\textit{expl}}$*.*

Proof. Consider the following RL algorithm:

```
procedure COVERINGLENGTHRL($\pi^{\text{expl}}$, $L$)
    collect a dataset of $L$ episodes, sampling actions according to $\pi^{\text{expl}}$
    record $\hat{R}(s, a)$ and $\hat{f}(s, a)$ for all state-action pairs seen in the dataset
    define $\hat{R}(s, a)$ and $\hat{f}(s, a)$ arbitrarily for state-action pairs not seen in the dataset
    run value iteration using $\hat{R}$ and $\hat{f}$ to obtain a policy $\pi$
    return $\pi$
end procedure
```

By the definition of covering length, with probability at least $1/2$ the algorithm should produce $\hat{R}(s, a) = R(s, a)$ and $\hat{f}(s, a) = f(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. In this case, π will be an optimal policy. Thus, COVERINGLENGTHRL returns an optimal policy with probability at least $1/2$ while interacting with the environment for TL timesteps. This means the sample complexity of COVERINGLENGTHRL is at most TL. ■

To bound the covering length for MDPs in the BRIDGE dataset, we make use of the following result.

Lemma D.4 (Bounds on the covering length)**.** *Define the occupancy measure* μ *of* $\pi^{\textit{expl}}$ *as*

$$\mu_t(s, a) = \mathbb{P}_{\pi^{\textit{expl}}}(s_t = s \wedge a_t = a).$$

Suppose that for every state-action pair (s, a)*, there is some timestep* t *when* $\mu_t(s, a) > 0$*. Then*

$$\frac{\log(2)}{2 \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ \sum_{t \in [T]} \mu_t(s, a)} \leq L \leq \left\lceil \frac{\log(2SA)}{\min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ \max_{t \in [T]} \mu_t(s, a)} \right\rceil.$$

We calculate $\mu_{\text{min}} = \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ \max_{t \in [T]} \mu_t(s, a)$ for each MDP in BRIDGE and use the upper bound from Lemma D.4 to obtain a sample complexity bound of $T \log(2SAT)/\mu_{\text{min}}$. Since $\sum_{t \in [T]} \mu_t(s, a) \leq T \max_{t \in [T]} \mu_t(s, a)$ the upper and lower bounds in Lemma D.4 agree up to a factor of $T \log(2SA)/\log(2)$, so this is reasonably tight. In fact, in 122 of the 155 MDPs in BRIDGE, $\min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ \sum_{t \in [T]} \mu_t(s, a) = \min_{(s,a) \in \mathcal{S} \times \mathcal{A}} \ \max_{t \in [T]} \mu_t(s, a)$, making the upper and lower bounds tight up to only logarithmic factors.

Proof. Let $\mathbb{P}_m$ be a probability measure corresponding to sampling m episodes following π^{expl}. Let

$\mathcal{E}_t^j(s,a)$ denote the event that the jth episode has $(s_t, a_t) = (s, a)$. Let $\mathcal{E}_t(s,a)$ be the event that $\mathcal{E}_t^j(s,a)$ occurs in at least one one of those episodes, i.e.

$$\mathcal{E}_t(s,a) \coloneqq \bigvee_{j=1}^{m} \mathcal{E}_t^j(s,a).$$

Finally, let $\mathcal{C}$ be the event defined by

$$\mathcal{C} \coloneqq \bigwedge_{(s,a)\in\mathcal{S}\times\mathcal{A}} \bigvee_{t\in[T]} \mathcal{E}_t(s,a).$$

That is, $\mathcal{C}$ is when every state-action pair has been seen at some timestep in at least one episode. We can thus equivalently define $L = \min\{m \mid \mathbb{P}_m(\mathcal{C}) \geq 1/2\}$.

We will now start by showing the upper bound of $L \leq \lceil \log(2SA)/p \rceil$. Let $m = \lceil \log(2SA)/p \rceil$. We can write

$$\begin{aligned}
\mathbb{P}_m(\neg\mathcal{C}) &= \mathbb{P}_m\left(\exists (s,a) \in \mathcal{S}\times\mathcal{A} \;:\; \forall t \in [T], j \in [m] \quad \neg\mathcal{E}_t^j(s,a)\right) \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mathbb{P}_m\left(\forall t \in [T], j \in [m] \quad \neg\mathcal{E}_t^j(s,a)\right) \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \min_{t\in[T]} \mathbb{P}_m\left(\forall j \in [m] \quad \neg\mathcal{E}_t^j(s,a)\right) \\
&= \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left(1 - \max_{t\in[T]} \mathbb{P}_m(\mathcal{E}_t^1(s,a))\right)^m \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left(1 - \min_{(s',a')\in\mathcal{S}\times\mathcal{A}} \max_{t\in[T]} \mathbb{P}_m(\mathcal{E}_t^1(s',a'))\right)^m \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \left(1 - \min_{(s',a')\in\mathcal{S}\times\mathcal{A}} \max_{t\in[T]} \mu_t(s',a')\right)^m \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \exp\left(-m \min_{(s',a')\in\mathcal{S}\times\mathcal{A}} \max_{t\in[T]} \mu_t(s',a')\right) \\
&\leq \sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \frac{1}{2SA} \\
&= \frac{1}{2},
\end{aligned}$$

which proves the upper bound.

To show the lower bound, take any $(s,a) \in \arg\min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \sum_{t\in[T]} \mu_t(s,a)$. First, suppose that $\sum_{t\in[T]} \mu_t(s,a) \geq \log(2)$. Then the lower bound in the lemma must be at most $1/2$, which is clearly true. Thus, we may subsequently assume $\sum_{t\in[T]} \mu_t(s,a) < \log(2)$.

Suppose $m < C/\sum_{t\in[T]} \cdot\mu_t(s,a)$ Then

$$\begin{aligned}
\mathbb{P}_m(\neg\mathcal{C}) &= \mathbb{P}_m\left(\exists(s',a')\in\mathcal{S}\times\mathcal{A} \ : \ \forall t\in[T], j\in[m] \quad \neg\mathcal{E}_t^j(s',a')\right) \\
&\geq \mathbb{P}_m\left(\forall t\in[T], j\in[m] \quad \neg\mathcal{E}_t^j(s,a)\right) \\
&= \left(1-\mathbb{P}_m\left(\exists t\in[T] \quad \mathcal{E}_t^1(s,a)\right)\right)^m \\
&\geq \left(1-\sum_{t\in[T]}\mathbb{P}_m\left(\mathcal{E}_t^1(s,a)\right)\right)^m \\
&= \left(1-\sum_{t\in[T]}\mu_t(s,a)\right)^m \\
&\overset{\text{(i)}}{\geq} \exp\left(-2m\sum_{t\in[T]}\mu_t(s,a)\right) \\
&> \frac{1}{2},
\end{aligned}$$

where (i) uses the fact that $1-x\geq\exp(-2x)$ for $x\in[0,\log(2)]$. This shows that with probability greater than $1/2$, not all state-action pairs will be seen in m episodes, and thus establishes that $L > m$, which is the desired bound. ■

D.3 Effective planning window (EPW)

Perhaps the closest existing concepts to our effective horizon are various notions of "effective planning window." This generally refers to tree-based planning algorithms which only consider action sequences of some length W from the current state, rather than considering action sequences all the way until the end of the MDP. For instance, Kearns et al. [24] show that in discounted MDPs, one need only plan to some ϵ-horizon to obtain an ϵ-optimal policy. Jiang et al. [53] build on this and show that one may want to use a different discount factor for planning than the one that is used for evaluation. Malik et al. [8] also introduce a notion of effective planning window based on the number of timesteps one must look ahead in an MDP to avoid terminal states.

We do not directly apply any of these previous results to our setting. Since we are concerned primarily with finite-horizon undiscounted MDPs, it does not make much sense to apply a discount factor as in Kearns et al. [24] and Jiang et al. [53]. We find that the assumptions in Malik et al. [8] are quite unusual and do not really hold in any of the environments in BRIDGE. In particular, the analysis in Malik et al. [8] requires that a trajectory through an MDP is either optimal or ends early in a terminal state.

Instead of directly using these results, we define a notion of effective planning window based on the length of action sequences one must consider in an MDP while ignoring any rewards after the sequence.

Definition D.5 (Effective planning window)**.** *Define $Q_t^1(s,a) = R(s,a)$ for all $(t,s,a)\in[T]\times\mathcal{S}\times\mathcal{A}$ and let $Q^i = \text{QVI}(Q^{i-1})$ for $i = 2,\dots,T$. The effective planning window of an MDP is the minimum $W\in[T]$ such that all policies in $\Pi(Q^W)$ are optimal.*

Note that the effective planning window bears significant similarity to the k-QVI-solvability property from Definition 5.1. However, Q^1 is defined as equal to the reward function for the EPW, while in Definition 5.1 it is equal to $Q^{\pi^{\text{expl}}}$.

The EPW also results in sample complexity bounds of T^2A^W very similar to those of T^2A^H for the effective horizon. However, we find empirically that $H < W$ in 82% of the MDPs in BRIDGE, making the effective horizon-based bounds generally tighter.

Theorem D.6. *For any MDP with effective planning window W, there is an RL algorithm whose sample complexity is at most T^2A^W.*

Proof. We will use the following algorithm:

```
1: procedure PLANOVERWINDOW($W$)
2:     for $i = 1, \ldots, T$ do
3:         for $a_{i:i+W-1} \in \mathcal{A}^k$ do
4:             Sample an episode following $\pi_1, \ldots, \pi_{i-1}$, then actions $a_{i:i+W-1}$, and then arbitrary actions.
5:             $\hat{R}_i(s_i, a_{i:i+W-1}) \leftarrow \sum_{t=i}^{i+W-1} R(s_t, a_t)$.
6:         end for
7:         $\pi_i(s_i) \leftarrow \arg\max_{a_i \in \mathcal{A}} \max_{a_{i+1:i+W-1} \in \mathcal{A}^{k-1}} \hat{R}_i(s_i, a_{i:i+W-1})$.
8:     end for
9:     return $\pi$
10: end procedure
```

Again, this algorithm is quite similar to GORP (Algorithm 1) except that it only samples a single episode per action sequence and it ignores rewards beyond the planning window. Clearly, PLANOVERWINDOW will take $T^2 A^W$ steps in the environment. Thus, to bound the sample complexity, we only need to show it returns an optimal policy with probability at least $1/2$.

To prove this, we will show that

$$\max_{a_{i+1:i+W-1} \in \mathcal{A}^{k-1}} \hat{R}_i(s_i, a_{i:i+W-1}) = Q_i^W(s_i, a_i). \tag{24}$$

Based on line 7 of the algorithm, this is enough to show that $\pi \in \Pi(Q^W)$, and thus that π must be optimal by Definition D.5.

To prove (24), we will first show by induction that $Q_t^j(s_t, a_t) = \max_{a_{t+1:t+j-1} \in \mathcal{A}^{j-1}} \sum_{t'=t}^{t+j-1} R(s_{t'}, a_{t'})$, where $s_{t'+1} = f(s_{t'}, a_{t'})$ for $t' = i, \ldots, i+W-2$. The base case when $j = 1$ is by definition: $Q_t^1(s_t, a_t) = R(s_t, a_t)$. For the inductive step, assume the formula holds for j and note that

$$\begin{aligned}
Q_t^{j+1}(s_t, a_t) &= \text{QVI}(Q_t^j)(s_t, a_t) \\
&= R(s_t, a_t) + \max_{a_{t+1} \in \mathcal{A}} Q_{t+1}^j(f(s_t, a_t), a_{t+1}) \\
&= R(s_t, a_t) + \max_{a_{t+1} \in \mathcal{A}} \max_{a_{t+2:t+j} \in \mathcal{A}^{j-1}} \sum_{t'=t+1}^{t+j} R(s_{t'}, a_{t'}) \\
&= \max_{a_{t+1:t+j} \in \mathcal{A}^j} \sum_{t'=t}^{t+j} R(s_{t'}, a_{t'}).
\end{aligned}$$

Next, note that by the way $\hat{R}$ is constructed, $\hat{R}(s_i, a_{i:i+W-1}) = \sum_{t=i}^{i+W-1} R(s_t, a_t)$, where $s_{t+1} = f(s_t, a_t)$ for $t = i, \ldots, i+W-2$. Thus combining this with the formula proved by induction, (24) clearly holds and the proof is complete. ■

D.4 Other bounds

One other work that derives sample complexity bounds for RL with random exploration is Dann et al. [6]. They define an algorithm which maintains at all times a current best policy π, and acts according to this policy but with some exploration noise, e.g., via an ϵ-greedy policy $\text{expl}_\epsilon(\pi)$. They introduce the notion of a "myopic exploration gap," which is defined as

$$\alpha = \sup_{\pi', c \geq 1} \frac{1}{\sqrt{c}} \left(J(\pi') - J(\pi) \right)$$

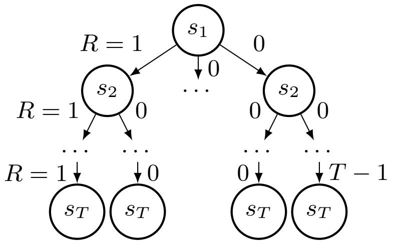
$$\text{such that for all } (t, s, a) \in [T] \times \mathcal{S} \times \mathcal{A}$$

$$\mu_t^{\pi'} \leq c \mu_t^{\text{expl}_\epsilon(\pi)}(s, a)$$

$$\mu_t^{\pi} \leq c \mu_t^{\text{expl}_\epsilon(\pi)}(s, a).$$

This gap is shown to generalize the notion of covering length as well as various others from the literature. However, we find that it is not so useful in many environments in BRIDGE.

The problem we find is illustrated in the MDP below:



In this MDP, one need simply follow the actions which give rewards of 1 to achieve the optimal return of T. One can show $H = W = 1$ in this MDP, which give identical sample complexities of $T^2 A$.

However, the difficulty with the myopic exploration gap is that the analysis in Dann et al. [6] cannot rule out the policy π which takes all actions to the right (achieving return $T - 1$) from being chosen at some point while running their RL algorithm. If this happens, then the only way to find a better policy is to completely switch to the policy π' which takes all left actions (achieving return T). This implies that α is maximized when $c = A^T$, leading to $\alpha = (1/A)^{T/2}$. Since the sample complexity bounds in Dann et al. [6] are $O(1/\alpha^2)$, this gives a bound proportional to A^T, which is no better than the worst case.

Thus, whenever there are “distracting” rewards, no matter how distant, as in this case, the theory from Dann et al. [6] cannot give good sample complexity bounds. There is also no easy way calculate α directly for an arbitrary environment. For these reasons, we do not include their bounds in our experiments.

E Dataset details

In this appendix, we give a detailed explanation of how we chose the MDPs in BRIDGE and how we constructed their tabular representations. See Figure 5 for an overview of BRIDGE.

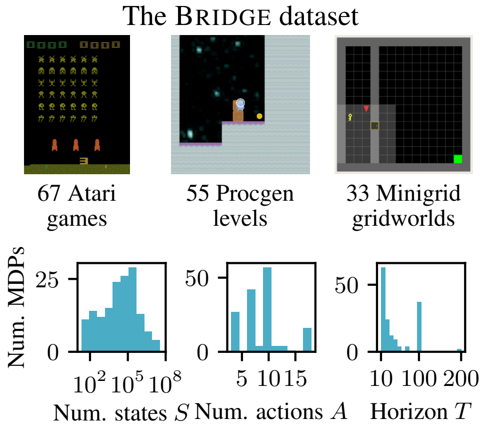


Figure 5: Our BRIDGE dataset consists of 155 deterministic MDPs with full tabular representations. We include MDPs from three popular RL benchmarks which cover a range of state space sizes, action state sizes, and horizons.

E.1 Environments

We limited the horizon of environments for BRIDGE to some $T \in \{10, 15, 20, 30, 50, 70, 100, 200\}$, depending on the environment, in order to avoid the state space becoming intractably large. We use subscripts to denote the horizon to which an environment is limited. For instance, PONG$_{50}$ refers to the Atari game Pong limited to 50 timesteps.

Frameskip We carefully used frameskip for each environment. Frameskip is a standard practice in Atari [54] in which each action taken in the environment is played for a certain number of frames; the agent only receives the next state after all these frames have completed. We use unusually high frameskips in order to capture episodes with longer wall-clock times in a small number of environment timesteps. The frameskip values we use are listed in Table 4. For most Atari games, we use a frameskip of 30, corresponding to taking 2 actions per second. The frameskips for Procgen environments vary; we chose ones that tended to align with how long it took the agent to perform various low-level tasks in the environment like moving one space. We did not use frameskip for Minigrid.

Environment	Frameskip
MontezumaRevenge	24
All other Atari games	30
Bigfish	8
Chaser	2
Climber	6
Coinrun	8
Dodgeball	8
Fruitbot	8
Heist	2
Jumper	8
Leaper	6
Maze	1
Miner	1
Ninja	8
Plunder	8
Starpilot	8
All Minigrid gridworlds	1

Table 4: Frameskip values used for the MDPs in Bridge.

Atari games For each of the 57 Atari games in the Arcade Learning Environment (ALE) benchmark [43], we attempted to construct tabular representations for each horizon $T \in \{10, 20, 30, 50, 70, 100, 200\}$. However, we excluded environments once the state space exceeded 100 million states. We kept multiple horizon-limited versions of games, i.e., Bridge contains Pong$_{10}$, Pong$_{20}$, Pong$_{30}$, etc. For some games, even $T = 10$ produced too many states, so we did not include these at all. We use the minimal action sets for each MDP rather than all 18 possible Atari actions.

We made one exception to these procedures for Montezuma's Revenge, as it is an environment well-known for being difficult to explore, so we wanted to make sure to include it in Bridge. We found that with $T = 10$, there was not enough time to get any reward, and with $T = 20$ there were too many states. We found that using $T = 15$ and a frameskip of 24 did allow an agent to receive reward, so we used this version in Bridge.

We made a couple other modifications to the standard Atari setup. First, we limited agents to one life: as soon as a life is lost, the episode ends. Second, in Skiing$_{10}$, we added an additional 200 frames of Noop actions after the 10 timesteps (= 300 frames) in each episode. This is necessary to correctly reflect the reward incentives in Skiing with longer horizons.

Finally, we scaled the rewards for many Atari games to make the reward scale more uniform across different games. Often, when deep RL is applied to Atari, rewards are *clipped* to $[-1, 1]$ to avoid instability. However, the MDP with clipped rewards may have a different optimal policy than the unclipped MDP. Thus, instead of clipping, we use scaling. We generally choose the scale factor based on the multiples of points received in the game: for instance, in Atlantis, rewards are always received in multiples of 100 so we scale by $1/100$. Table 5 lists the reward scaling factors for all games where we apply scaling.

Game	Reward scaling factor
Alien	$1/10$
Amidar	$1/10$
Assault	$1/21$
Asterix	$1/50$
Asteroids	$1/10$
Atlantis	$1/100$
BankHeist	$1/10$
BattleZone	$1/1000$
BeamRider	$1/44$
Centipede	$1/100$
ChopperCommand	$1/100$
CrazyClimber	$1/100$
DemonAttack	$1/10$
Frostbite	$1/10$
Gopher	$1/20$
Hero	$1/25$
Kangaroo	$1/100$
MontezumaRevenge	$1/100$
MsPacman	$1/10$
NameThisGame	$1/10$
Phoenix	$1/20$
PrivateEye	$1/100$
Qbert	$1/25$
RoadRunner	$1/100$
Qeaquest	$1/20$
Skiing	$1/100$
SpaceInvaders	$1/5$
TimePilot	$1/100$
VideoPinball	$1/100$
WizardOfWor	$1/100$

Table 5: Factors by which Atari games rewards are scaled by in Bridge, for those where we apply reward scaling.

Procgen levels The Procgen benchmark [44] consists of 16 games. For each game, one can generate an arbitrary number of random levels, each of which is identified by a seed. Furthermore, each game has an "easy" and "hard" difficulty, each with different levels, and some have an additional "exploration" level which presents a particularly difficult exploration challenge.

While the benchmark is designed to measure generalization of RL agents trained on some number of levels to unseen levels, we use each level as a separate MDP. For each game, we attempted to construct an MDP for the easy levels with seeds 0, 1, and 2, the hard level with seed 0, and the exploration level if it exists for that environment. We denote $\textsc{Maze}^{\text{E1}}_{30}$ to be the easy level with seed 1 for the Maze game limited to $T = 30$ timesteps; $\textsc{Maze}^{\text{H0}}_{30}$ is the analogous hard level with seed 0 and $\textsc{Maze}^{\text{EX}}_{30}$ is the exploration level.

We generally increased the horizon for each game to the highest value in $\{10, 20, 30, 40, 50, 70, 100, 200\}$ before the number of states was greater than 100 million. The horizon values we ultimately chose can be seen in the table in Appendix G.3.

Minigrid gridworlds Minigrid [14] is an extensible framework for building gridworlds. We considered all the pre-built gridworlds included in Minigrid for inclusion in Bridge except for those requiring natural language observations for specifying the task to be completed. We also excluded gridworlds with more than 1 million states, since for technical reasons we were unable to parallelize the construction of tabular MDPs for Minigrid. For gridworlds with randomized start states, we chose the start state with seed 0. We use $T = 100$ for all the gridworlds.

E.2 Constructing tabular representations

For each of the environments described above, we wrote a program to compute a full tabular representation of the transition function f and reward function R. Our program uses a search procedure to iteratively explore every state-action pair. We keep a queue of states that need to be explored, which at first is just the initial state. In parallel, a number of worker threads take states from this queue. After popping a state, a worker thread sets the environment to that state and then takes a previously unexplored action, storing the resulting next state and reward. If there are still unexplored actions in the current state, it adds it back to the queue. If the next state is not terminal and has not had all its actions, it can continue this process.

While the search procedure for exhaustively enumerating states is conceptually simple, we experienced difficulties implementing it efficiently due to the massive scale of some of the MDPs in BRIDGE. For instance, the full state of the Atari simulator used in the ALE is about 10-12 KB of data. Storing 100 million states by themselves would thus require over one terabye of memory! We avoided this problem by aggressively compressing state data using dictionary compression. Other challenges included efficiently parallelizing the data structures we used to store the queue of states, the transition function, and the reward function. Our final implementation is able to explore more than 20,000 state-action pairs per second in PONG while running on 64 cores.

Once we have enumerated all states and actions that can be reached in the given horizon, we also apply a consolidation step to reduce the number of states. Often, the internal representation of states in the Atari and Procgen environments includes extra or superfluous data, which leads to duplicate states in our tabular representation. We repeatedly consolidate states that (a) have the same screen, (b) have the same rewards for each action, and (c) lead to the same next states for each action. When no more states can be consolidated, we store the resulting transition and reward functions.

We excluded any MDPs for which every sequence of actions results in the same total reward, since these are uninteresting from an RL perspective.

E.3 Reward shaping

For each Minigrid environment, we constructed one or more versions with shaped rewards for our experiments on the effects of reward shaping. We used three potential functions for shaping:

1. $\Phi_{\text{dist}}(s)$: the negative distance from the state to the nearest goal. Distance is measured as the minimum number of moves needed to reach the goal, assuming there are no obstacles in the way.
2. $\Phi_{\text{doors}}(s)$: the number of doors that are open.
3. $\Phi_{\text{pickup}}(s)$: the number of objects that have been picked up at least once.

For one or more potential functions Φ, we augment each reward $R(s, a)$ with $\Phi(f(s, a)) - \Phi(s)$. The potential functions are chosen to incentivize useful behavior in the environments: moving towards goals, picking up objects like keys that could be helpful, and opening doors to reach more parts of the gridworld.

For each Minigrid MDP, we use all potential functions that apply to that MDP. For instance, if an MDP does not have any doors, we do not use Φ_{doors}. We also apply the combination of Φ_{doors} and Φ_{pickup} if both are applicable to an MDP.

When analysing the reward shaping results, we only include MDPs for which PPO/DQN converged on both the unshaped and shaped versions.

E.4 Datasheet for BRIDGE

We provide a datasheet, as proposed by Gebru et al. [55], for the BRIDGE dataset.

E.4.1 Motivation

For what purpose was the dataset created? We have described the purpose extensively in the paper: we aim to bridge the theory-practice gap in RL. BRIDGE allows this by providing tabular representations of popular deep RL benchmarks such that instance-dependent bounds can be calculated and compared to empirical RL performance.

Who created the dataset (e.g., which team, research group) and on behalf of which entity (e.g., company, institution, organization)? Not specified for the double-blind reviewing process.

Who funded the creation of the dataset? Also not specified for the double-blind reviewing process.

Any other comments? No.

E.4.2 Composition

What do the instances that comprise the dataset represent (e.g., documents, photos, people, countries)? The instances are Markov Decision Processes (MDPs).

How many instances are there in total (of each type, if appropriate)? There are 155 MDPs in BRIDGE. They include 67 MDPs based on Atari games from the Arcade Learning Environment [43], 55 MDPs based on Procgen games [44], and 33 MDPs based on MiniGrid gridworlds [14].

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? The MDPs in BRIDGE are based on a small subset of the many environments that are used for empirically evaluating RL algorithms. We aimed to cover a range of the most popular environments. To make our analysis possible, we excluded environments that were not deterministic or did not have discrete action spaces. We also reduced the horizon of many of the environments to make it tractable to compute their tabular representations.

What data does each instance consist of? For each MDP, we provide the following data:

- A transition function and a reward function, which are represented as a matrix with an entry for each state-action pair in the MDP.
- A corresponding `gym` environment [56] that can be used to train policies for the MDP with various RL algorithms.
- Properties of the MDP that are calculated from its tabular representation, including the effective planning window, bounds on the effective horizon, bounds on the covering length, etc.
- Results of running RL algorithms (PPO, DQN, and GORP) on the MDP. This includes the empirical sample complexity as well as various metrics logged during training.
- For MiniGrid MDPs, there are additional versions of the MDP with shaped reward functions (see Appendix E.3) which also include all of the above data.
- For Atari and Procgen MDPs, there is additionally a non-uniform exploration policy (see Appendix F.2). For Atari games, this is trained via behavior cloning from the Atari-HEAD [57] dataset; for Procgen games, it is trained on other Procgen levels. We include the above data recalculated using the non-uniform exploration policy in place of the uniformly random exploration policy.

Is there a label or target associated with each instance? In this paper, we aim to bound and/or estimate the empirical sample complexity of RL algorithms, so these could be considered targets for each instance.

Is any information missing from individual instances? There is no information missing.

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)? No.

Are there recommended data splits (e.g., training, development/validation, testing)? No.

Are there any errors, sources of noise, or redundancies in the dataset? We do not believe there are errors or sources of noise in the dataset. The tabular representations of the MDPs have been carefully tested for correspondence with the environments they are based on. There is some redundancy, as many Atari games are represented more than once with varying horizons.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? The dataset is mostly self-contained, except that the `gym` environments rely on external libraries. There are archival versions of these available through package managers like PyPI.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor– patient confidentiality, data that includes the content of individuals' non-public communications)? No.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? No.

E.4.3 Collection process

How was the data associated with each instance acquired? The data was collected using open-source implementations of each environment.

What mechanisms or procedures were used to collect the data (e.g., hardware apparatuses or sensors, manual human curation, software programs, software APIs)? As described in Appendix E.2, we developed a software tool to construct the tabular representations of the MDPs in BRIDGE. We validated the correctness of the tabular MDPs through extensive testing to ensure they corresponded exactly with the `gym` implementations of the environments.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)? The MDPs in BRIDGE were selected from three collections of commonly used RL environments: the Arcade Learning Environment, ProcGen, and MiniGrid. We chose these three collections to represent a broad set of deterministic environments with discrete action spaces. Within each collection, the environments were further filtered based on the criteria described in Appendix E.1.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)? Only the authors were involved in the data collection process.

Over what timeframe was the data collected? The dataset was assembled between February 2022 and January 2023. The RL environments from which the MDPs in BRIDGE were constructed were created prior to this; see the cited works for each collection of environments for more details.

Were any ethical review processes conducted (e.g., by an institutional review board)? No.

E.4.4 Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? Yes, various preprocessing and analysis was done. See Appendix E.2 for details.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? Yes, this is included with the dataset.

Is the software that was used to preprocess/clean/label the data available? Yes, this is available with the rest of our code.

Any other comments? No.

E.4.5 Uses

Has the dataset been used for any tasks already? The dataset has thus far only been used to validate our theory of the effective horizon in this paper.

Is there a repository that links to any or all papers or systems that use the dataset? There is not. However, we will require that any uses of the dataset cite this paper, allowing one to use tools like Semantic Scholar or Google Scholar to find other papers which use the BRIDGE dataset.

What (other) tasks could the dataset be used for? We hope that the BRIDGE dataset is used for further efforts to bridge the theory-practice gap in RL. The dataset could be used to identify other properties or assumptions that hold in common environments, or to calculate instance-dependent sample complexity bounds and compare them to the empirical sample complexity of RL algorithms.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? As we have already mentioned, BRIDGE is restricted to deterministic MDPs with discrete action spaces and relatively short horizons. This could mean that analyses of the dataset like ours do not generalize to the broader space of RL environments that may have continuous action spaces, stochastic transitions, and/or long horizons. We have included some experiments, like those in Appendix F.1, to show that our theory of the effective horizon generalizes beyond the MDPs in BRIDGE. We encourage others to do the same and we hope to address some of these limitations in the future with extensions to BRIDGE.

Are there tasks for which the dataset should not be used? We do not foresee any particular tasks for which the dataset should not be used.

Any other comments? No.

E.4.6 Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? Yes, we will distribute the dataset publicly.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? We are still finalizing the method through which the dataset will be distributed.

When will the dataset be distributed? We plan to make the dataset public in May or June 2023.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? It will be distributed under CC-BY-4.0.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? The Atari ROMs used to construct the Atari MDPs in BRIDGE are copyrighted by the original creators of the games. However, they are widely used throughout the reinforcement learning literature and to our knowledge the copyright holders have not complained about this. Since we are not legal experts, we do not know if releasing our dataset violates their copyright, but we do not believe that we are harming them since the tabular representations in BRIDGE are only useful for research purposes and cannot be used to play the games in any meaningful way.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? No.

Any other comments? No.

E.4.7 Maintenance

Who will be supporting/hosting/maintaining the dataset? We (the authors) will support and maintain the dataset.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)? Redacted for double-blind review.

Is there an erratum? We will record reports of any errors in the dataset and release new versions with descriptions of what was fixed as necessary.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? We will release new versions of the dataset to correct any reported errors as described above. We may also expand the dataset in the future with more MDPs or new kinds of MDPs, such as stochastic or continuous-action-space MDPs. Any updates will be communicated through the service we use to host the dataset (TBD).

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were the individuals in question told that their data would be retained for a fixed period of time and then deleted)? No.

Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to dataset consumers. We hope to find a host for the dataset that will retain older versions of the dataset. We only plan to maintain the latest version of the dataset, however. We will note this policy in the dataset's description.

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? There is no predefined mechanism to contribute to the dataset, but we will consider external contributions on a case-by-case basis. We encourage others to extend and build on the dataset.

Any other comments? No.

F Experiment details

In this appendix, we describe details of the experiments from Section 6. In particular, we describe how we calculate the empirical sample complexity of PPO and DQN.

We use the implementations of PPO and DQN from Stable-Baselines3 (SB3) [58]. For the network archictures, we use convolutional neural nets (CNNs) for Atari and Procgen and a fully-connected network for Minigrid. The network architectures are the default CNN and fully-connected architectures chosen by SB3. We used the below hyperparameters for PPO and DQN, which are mainly taken from the tuned Atari hyperparameters in the RL Baselines3 Zoo [59] repository.

We used a discount rate of $\gamma = 1$ for all environments and algorithms except for PPO and DQN in MiniGrid environments, where we used $\gamma = 0.99$. We found that the performance of PPO and DQN significantly degraded when we used $\gamma = 1$ for MiniGrid.

PPO We use the following hyperparameters for PPO:

Hyperparameter	Value
Training timesteps	5,000,000
Number of environments	8
Number of steps per rollout	$\{128, 1280\}$
Clipping parameter (ϵ)	0.1
Value function coefficient	0.5
Entropy coefficient	0.01
Optimizer	Adam
Learning rate	2.5×10^{-4}
Number of epochs per training batch	4
Minibatch size	256
GAE coefficient (λ)	0.95
Advantage normalization	Yes
Gradient clipping	0.5

Table 6: Hyperparameters we use for PPO.

For each environment, we try rollout lengths of 128 and 1,280, as we find this is the most sensitive hyperparameter to tune.

DQN We use the following hyperparameters for DQN:

Hyperparameter	Value
Training timesteps	5,000,000
Timesteps before learning starts	0
Replay buffer size	100,000
Target network update interval	1,000
Training frequency	4
Gradient steps per training step	1
Optimizer	Adam
Learning rate	10^{-4}
Exploration fraction	$\{0.1, 1\}$
Final ϵ	0.01
Learning rate	10^{-3}
Gradient clipping	10

Table 7: Hyperparameters we use for DQN.

We try decaying the ϵ value for ϵ-greedy over the course of either 500 thousand or 5 million timesteps, as we found this was the most sensitive hyperparameter to tune for DQN.

Calculating the empirical sample complexity In order to compute the empirical sample complexities of PPO and DQN, throughout training we run evaluation episodes and see if the algorithms have discovered an optimal policy yet. During the evaluation, PPO policies take actions according to the argmax over the probabilities they assign to each action, rather than sampling as during training episodes. DQN takes actions greedily with respect to its current Q-function (i.e., with $\epsilon = 0$). If the total episode reward during the evaluation is the optimal return, then we terminate the training run and record the total number of timesteps interacted with the environment as the empirical sample complexity. We take the median sample complexity over 5 random seeds and then the minimum over all hyperparameter settings to get the final empirical sample complexity.

We found that in some environments PPO achieved optimal reward during almost all the training episodes but none of the evaluation episodes. This can happen if a policy does not assign the highest probability to an optimal action in some states but can make up for this by being very likely overall to obtain the highest possible total reward. Thus, if more than half of the training episodes during an iteration achieve the optimal return, we also count this as converging to an optimal policy for the purposes of calculating the empirical sample complexity.

F.1 GORP vs. deep RL algorithms over longer horizons

To show that GORP is not just effective over short horizons, we ran additional experiments comparing GORP, PPO, and DQN in more typical Atari benchmark environments with frameskip 4 and a horizon of $T = 27,000$ (corresponding to a maximum of 30 minutes of gameplay). Similarly to the environments in BRIDGE, we limit agents to a single life and make the environments entirely deterministic. The hyperparameters for PPO and DQN are identical to those given in Tables 6 and 7 except for the following changes: we train for 50 million timesteps; we use a discount rate of $\gamma = 0.99$; and, we set an entropy coefficient of 0.01 for PPO.

We use a training batch size of 10^4 for PPO and decay ϵ for DQN over the course of the first 5 million timesteps. For GORP, we use $k = 1$ and tune m for each environment.

F.2 Exploration policies

For the experiments in Section 6 we needed pre-trained policies to initialize PPO with; here, we describe the details of how we trained them. We used two different training methods: one for Atari environments and one for Procgen environments.

In the Atari environments, we trained policies via behavior cloning (BC), i.e., supervised learning, from human data in the Atari-HEAD dataset [57]. We resampled the actions and processed the screen images from the dataset to align with the frameskip and observation preprocessing of our Atari environments. We trained a BC policy on each environment for 400 batches of 500 timesteps each. We used Adam with a learning rate of 10^{-3}. We also added an entropy bonus to the loss function with a weight of 0.1 to avoid the BC policy assigning very little weight to some actions. Our theoretical results in Section B.1 suggest that this should improve the sample complexity. Since not all Atari games are included in Atari-HEAD, we only used a subset for the experiments in Section 6.

In the Procgen environments, we pre-trained policies on a set of levels not included in BRIDGE. In particular, we trained a policy with PPO on 500 easy levels for 25 million timesteps, which in very similar to the methodology in Cobbe et al. [44]. We also use an entropy bonus in PPO with weight 0.1 for the same reason as above.

Once we have the pre-trained policies, we compute tabular representations of them on the corresponding MDPs in BRIDGE. To do so, we feed the observations for every state in the MDP through the pre-trained policy network and record the resulting action distribution. This allows us to compute $Q^{\pi^{\text{expl}}}$ and thus obtain bounds on the effective horizon when using the pre-trained exploration policy.

F.3 Computational resources

For deep RL experiments, we used a mix of A100, A4000, and A6000 GPUs from Nvidia. We ran the algorithms either on separate GPUs or sometimes we ran multiple random seeds simultaneously on the same hardware. We used 1-8 CPU threads to run the RL environments. Using this setup, PPO and DQN generally took 2-8 hours to complete 5 million timesteps of training. We used early stopping when the algorithms found an optimal policy before 5 million timesteps, so the amount of compute per experiment was often less than this.

For constructing and analyzing the tabular MDPs in BRIDGE, we used up to 128 CPU threads and 500 GB of memory. The amount of time necessary to construct and analyze the MDPs ranged from less than a minute to around 5 days.

G Additional experiment results

Here, we present additional results from the experiments in Section 6.

G.1 Example of the effective horizon failing to predict generalization

As we described in the discussion, the effective horizon cannot model generalization across different states. For instance, in PONG-30 (Pong limited to 30 timesteps/15 seconds), the effective horizon gives a sample complexity of roughly 5 billion timesteps and empirically GORP takes over 80 million timesteps to converge to an optimal policy (Appendix G.3). However, both PPO and DQN converge in under 500,000 environment steps. We hypothesize this is because they are able to generalize the skill of hitting the ball across the multiple rounds of the Pong game, which the effective horizon cannot capture because it considers learning separately at every timestep.

G.2 Additional plots

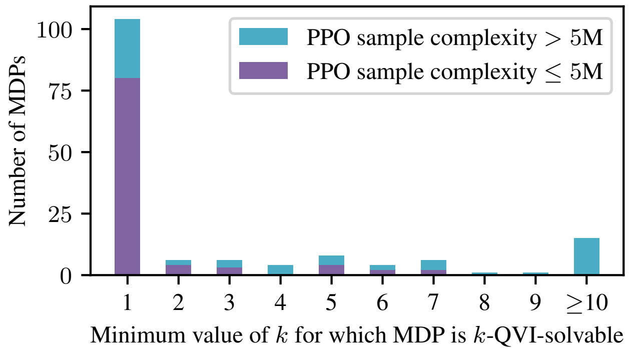


Figure 6: The distribution of the minimum values of k for which the MDPs in BRIDGE are k-QVI solvable. About two thirds are 1-QVI-solvable, meaning they can be solved by simply acting greedily with respect to the Q-function of the random policy. The MDPs are split into those which PPO can and cannot solve in 5 million steps; among those that can be solved efficiently, the values of k are even lower.

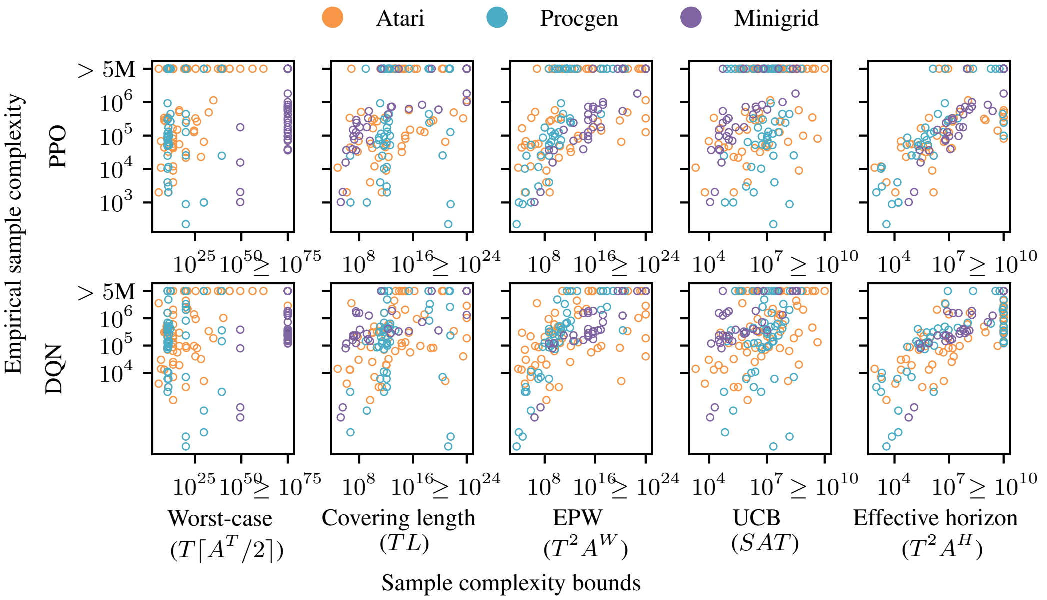


Figure 7: A comparison of sample complexity bounds and the empirical sample complexities of PPO and DQN across the MDPs in BRIDGE. In each plot, every dot represents one MDP and its color indicates which benchmark it comes from. Our effective horizon-based bound most closely correlates with empirical sample complexity. See Table 2 for a quantitative comparison of the bounds.

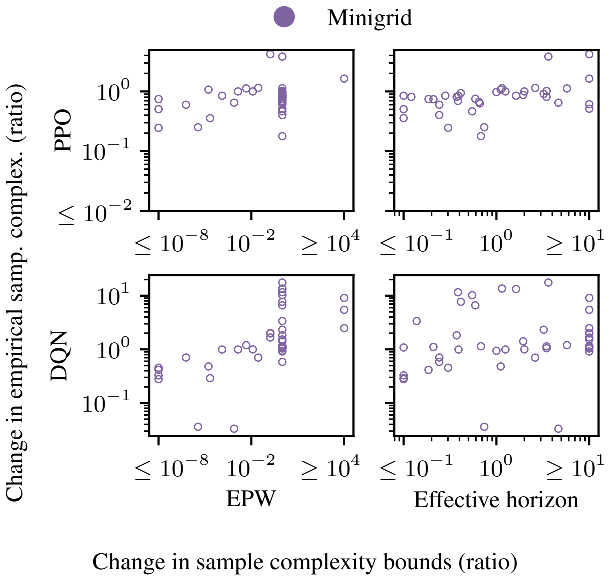


Figure 8: A comparison between the empirical change in sample complexity and the change predicted by sample complexity bounds due to reward shaping. See Table 3a for a quantitative comparison.

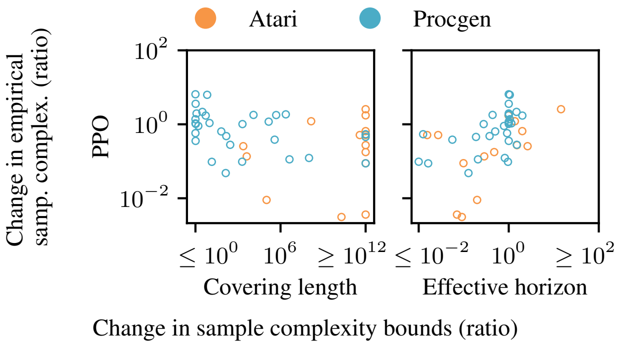


Figure 9: A comparison between the empirical change in the sample complexity of PPO and the change predicted by sample complexity bounds due to initializing with a pre-trained policy. The initial policies for Atari are trained from human data and those for Procgen are trained on other procedurally generated levels. See Table 3b for a quantitative comparison.

G.3 List of MDPs in BRIDGE with statistics

The following table lists all the MDPs in BRIDGE, along with various properties: the number of states S, number of actions A, horizon T, minimum k for which the MDP is k-QVI-solvable, a bound on the effective horizon using the techniques in Appendix C, a bound on the covering length L using Lemma D.4, and the effective planning window W.

MDP	S	A	T	Min k	Bound on H	Bound on L	W
ALIEN$_{10}$	7.97×10^5	18	10	1	3.5	2.27×10^{12}	6
AMIDAR$_{20}$	1.00×10^5	10	20	4	12	4.03×10^{11}	12
ASSAULT$_{10}$	7.21×10^5	7	10	3	9.8	4.56×10^9	8
ASTERIX$_{10}$	9.03×10^4	9	10	1	3.5	4.99×10^{10}	3
ASTEROIDS$_{10}$	5.11×10^6	14	10	7	10	2.72×10^{12}	9
ATLANTIS$_{10}$	49	4	10	1	1.5	6.12×10^3	3
ATLANTIS$_{20}$	471	4	20	1	4.7	5.53×10^8	3
ATLANTIS$_{30}$	4.56×10^3	4	30	19	23	2.89×10^{12}	23
ATLANTIS$_{40}$	2.06×10^4	4	40	5	27.7	9.62×10^{14}	9
ATLANTIS$_{50}$	5.63×10^4	4	50	12	43	4.27×10^{18}	38
ATLANTIS$_{70}$	1.41×10^5	4	70	47	62	5.11×10^{24}	50
BANKHEIST$_{10}$	4.24×10^6	18	10	1	3.5	6.73×10^{13}	9
BATTLEZONE$_{10}$	7.01×10^4	18	10	1	2.3	5.01×10^8	4
BEAMRIDER$_{20}$	2.39×10^4	9	20	1	3.3	1.10×10^{13}	10
BOWLING$_{30}$	2.50×10^5	6	30	2	18.5	1.73×10^{16}	28
BREAKOUT$_{10}$	238	4	10	1	4.1	7.92×10^6	5
BREAKOUT$_{20}$	1.27×10^3	4	20	1	5.4	1.01×10^{13}	13
BREAKOUT$_{30}$	2.80×10^3	4	30	4	13	1.15×10^{19}	29
BREAKOUT$_{40}$	6.12×10^3	4	40	4	15.7	1.31×10^{25}	29
BREAKOUT$_{50}$	1.62×10^4	4	50	3	17.3	1.49×10^{31}	20
BREAKOUT$_{70}$	7.83×10^4	4	70	40	48.5	1.86×10^{43}	61
BREAKOUT$_{100}$	1.31×10^5	4	100	74	87.0	2.23×10^{61}	75
BREAKOUT$_{200}$	1.39×10^5	4	200	105	114.9	3.60×10^{121}	108
CENTIPEDE$_{10}$	1.32×10^7	18	10	7	10	7.13×10^{13}	7
CHOPPERCOMMAND$_{10}$	1.39×10^6	18	10	1	3.9	1.95×10^{11}	7
CRAZYCLIMBER$_{20}$	1.77×10^3	9	20	1	3.1	4.02×10^9	8
CRAZYCLIMBER$_{30}$	4.63×10^5	9	30	1	3.9	2.15×10^{19}	18
DEMONATTACK$_{10}$	6.32×10^4	6	10	5	9	8.19×10^8	9
ENDURO$_{10}$	2.54×10^7	9	10	7	10	6.95×10^{10}	9
FISHINGDERBY$_{10}$	2.80×10^5	18	10	6	9	3.60×10^{12}	9
FREEWAY$_{10}$	198	3	10	1	4.3	1.39×10^5	6
FREEWAY$_{20}$	3.16×10^3	3	20	1	7.4	1.14×10^{10}	13
FREEWAY$_{30}$	1.02×10^4	3	30	1	7.8	1.26×10^{14}	26

Freeway$_{40}$	2.08×10^4	3	40	1	7.9	2.64×10^{17}	35
Freeway$_{50}$	3.40×10^4	3	50	1	8.1	5.95×10^{21}	44
Freeway$_{70}$	7.02×10^4	3	70	1	9.1	3.89×10^{29}	66
Freeway$_{100}$	1.51×10^5	3	100	39	55.4	2.82×10^{41}	67
Freeway$_{200}$	6.33×10^5	3	200	3	30.3	1.26×10^{78}	150
Frostbite$_{10}$	5.73×10^4	18	10	1	2.9	8.01×10^{10}	3
Gopher$_{30}$	778	8	30	1	1.9	7.62×10^{10}	5
Gopher$_{40}$	8.18×10^3	8	40	5	9.7	4.86×10^{12}	13
Hero$_{10}$	4.89×10^3	18	10	1	1	1.85×10^9	3
IceHockey$_{10}$	2.53×10^6	18	10	1	3.2	1.01×10^{11}	5
Kangaroo$_{20}$	1.30×10^5	18	20	1	3.7	2.40×10^{17}	11
Kangaroo$_{30}$	5.84×10^6	18	30	21	24	1.07×10^{30}	23
MontezumaRevenge$_{15}$	8.47×10^3	18	15	1	8.2	1.08×10^{15}	15
MsPacman$_{20}$	1.85×10^6	9	20	11	11	8.15×10^{11}	11
NameThisGame$_{20}$	6.04×10^3	6	20	2	8	9.40×10^6	5
Phoenix$_{10}$	4.64×10^4	8	10	5	10	1.45×10^{10}	8
Pong$_{20}$	255	6	20	1	3.2	5.60×10^{10}	5
Pong$_{30}$	2.01×10^3	6	30	6	14.9	4.15×10^{15}	18
Pong$_{40}$	1.60×10^4	6	40	6	14.3	2.96×10^{20}	23
Pong$_{50}$	1.25×10^5	6	50	13	19.6	2.04×10^{25}	25
Pong$_{70}$	2.89×10^6	6	70	20	28.1	8.69×10^{34}	25
Pong$_{100}$	3.46×10^7	6	100	42	70.1	2.05×10^{49}	25
PrivateEye$_{10}$	1.29×10^4	18	10	1	4.1	8.64×10^8	8
Qbert$_{10}$	289	6	10	5	5	3.80×10^5	5
Qbert$_{20}$	3.75×10^6	6	20	5	14.6	1.18×10^{14}	9
RoadRunner$_{10}$	2.37×10^7	18	10	3	9.9	7.34×10^{13}	9
Seaquest$_{10}$	5.46×10^3	18	10	1	1.2	4.15×10^8	5
Skiing$_{10}$	1.75×10^4	3	10	8	10	6.83×10^5	10
SpaceInvaders$_{10}$	994	6	10	1	3.0	4.38×10^5	3
Tennis$_{10}$	3.79×10^5	18	10	3	8.2	2.29×10^{11}	6
TimePilot$_{10}$	5.03×10^3	10	10	1	3.9	5.76×10^6	6
Tutankham$_{10}$	1.66×10^4	8	10	3	7.7	1.68×10^9	9
VideoPinball$_{10}$	1.25×10^5	9	10	2	9.5	5.10×10^{10}	8
WizardOfWor$_{20}$	8.92×10^3	10	20	7	13	1.89×10^{13}	13
BigFish$^{E0}_{10}$	2.31×10^4	9	10	5	9	2.26×10^{10}	8
BigFish$^{E1}_{10}$	2.74×10^4	9	10	1	2.8	4.57×10^{10}	4
BigFish$^{E2}_{10}$	1.96×10^5	9	10	9	10	5.26×10^{10}	10
BigFish$^{H0}_{10}$	9.26×10^3	9	10	1	4.3	4.19×10^{10}	7
Chaser$^{E0}_{20}$	8.13×10^5	9	20	2	12.1	1.16×10^{17}	15
Chaser$^{E1}_{20}$	3.98×10^5	9	20	1	4.4	1.62×10^{17}	9
Chaser$^{E2}_{20}$	5.03×10^5	9	20	5	15.2	1.34×10^{17}	10
Chaser$^{H0}_{20}$	8.75×10^5	9	20	18	20	1.04×10^{17}	19
Climber$^{E0}_{10}$	2.42×10^5	9	10	1	3.4	2.67×10^{10}	7
Climber$^{E1}_{10}$	1.18×10^5	9	10	1	4.0	8.46×10^9	9
Climber$^{E2}_{10}$	1.12×10^5	9	10	1	2.8	8.43×10^9	6
Climber$^{H0}_{10}$	2.33×10^5	9	10	1	6.6	2.66×10^{10}	10
CoinRun$^{E0}_{10}$	2.23×10^5	9	10	1	4.2	1.77×10^{10}	8
CoinRun$^{E1}_{10}$	6.23×10^4	9	10	1	3.6	4.05×10^9	7
CoinRun$^{E2}_{10}$	1.74×10^5	9	10	1	5.0	1.30×10^{10}	9
CoinRun$^{H0}_{10}$	2.72×10^5	9	10	1	2.3	1.34×10^{10}	7
Dodgeball$^{E0}_{10}$	1.19×10^5	10	10	1	4.2	1.47×10^{11}	7
Dodgeball$^{E1}_{10}$	1.95×10^4	10	10	2	9.1	1.29×10^{11}	10
Dodgeball$^{E2}_{10}$	3.42×10^4	10	10	1	3.7	1.34×10^{11}	6
Dodgeball$^{H0}_{10}$	7.24×10^4	10	10	7	10	1.42×10^{11}	8
FruitBot$^{E0}_{40}$	230	9	40	13	18.6	2.32×10^7	10
FruitBot$^{E1}_{40}$	379	9	40	18	24.1	6.00×10^{11}	18
FruitBot$^{E2}_{40}$	161	9	40	1	3.1	1.12×10^5	2
FruitBot$^{H0}_{40}$	620	9	40	6	18.7	6.56×10^{13}	24
Heist$^{E1}_{10}$	8.28×10^4	9	10	1	6.3	4.96×10^{10}	10

Jumper$_{10}^{H0}$	1.30×10^{5}	9	10	1	8.5	5.11×10^{10}	10
Jumper$_{20}^{E0}$	1.20×10^{5}	9	20	1	1	8.86×10^{19}	1
Jumper$_{20}^{E1}$	8.29×10^{5}	9	20	1	1.3	2.01×10^{20}	2
Jumper$_{20}^{E2}$	1.38×10^{6}	9	20	1	2.4	1.29×10^{19}	5
Jumper$_{20}^{EX}$	3.40×10^{6}	9	20	1	13.1	1.09×10^{20}	17
Leaper$_{20}^{E1}$	1.05×10^{4}	9	20	1	11.2	1.48×10^{20}	16
Leaper$_{20}^{E2}$	2.20×10^{5}	9	20	1	4.3	1.85×10^{20}	8
Leaper$_{20}^{H0}$	1.71×10^{4}	9	20	1	13.1	1.54×10^{20}	15
Leaper$_{20}^{EX}$	3.31×10^{4}	9	20	1	12.8	1.62×10^{20}	16
Maze$_{30}^{E0}$	244	9	30	1	1.3	1.65×10^{5}	1
Maze$_{30}^{E1}$	1.46×10^{3}	9	30	1	14.2	9.02×10^{22}	23
Maze$_{30}^{E2}$	655	9	30	1	1.3	4.48×10^{7}	3
Maze$_{30}^{H0}$	1.75×10^{3}	9	30	1	7.9	8.26×10^{23}	15
Maze$_{100}^{EX}$	3.72×10^{4}	9	100	1	40.3	4.27×10^{63}	76
Miner$_{10}^{E0}$	5.96×10^{4}	9	10	1	3.8	4.84×10^{10}	8
Miner$_{10}^{E1}$	6.86×10^{4}	9	10	1	5.2	4.89×10^{10}	9
Miner$_{10}^{E2}$	5.56×10^{4}	9	10	1	1.3	4.82×10^{10}	3
Miner$_{10}^{H0}$	1.04×10^{5}	9	10	1	3.4	5.04×10^{10}	7
Ninja$_{10}^{E0}$	2.02×10^{5}	13	10	1	4.4	2.67×10^{11}	9
Ninja$_{10}^{E1}$	4.87×10^{5}	13	10	1	3.9	2.82×10^{11}	8
Ninja$_{10}^{E2}$	2.11×10^{5}	13	10	1	7.0	2.67×10^{11}	9
Ninja$_{10}^{H0}$	6.61×10^{4}	13	10	1	6.8	2.47×10^{11}	10
Plunder$_{10}^{E0}$	1.55×10^{4}	10	10	1	1.3	1.26×10^{11}	5
Plunder$_{10}^{E1}$	2.06×10^{4}	10	10	1	1	1.29×10^{11}	3
Plunder$_{10}^{E2}$	9.99×10^{3}	10	10	1	3.3	1.22×10^{11}	7
Plunder$_{10}^{H0}$	8.28×10^{3}	10	10	1	1.3	1.20×10^{11}	5
Starpilot$_{10}^{E0}$	3.24×10^{5}	11	10	2	8.0	4.09×10^{11}	9
Starpilot$_{10}^{E1}$	1.76×10^{5}	11	10	1	3.7	3.93×10^{11}	8
Starpilot$_{10}^{E2}$	9.50×10^{4}	11	10	7	9	3.77×10^{11}	7
Starpilot$_{10}^{H0}$	2.46×10^{5}	11	10	4	9.2	4.02×10^{11}	8
Empty-5x5	37	3	100	1	1.6	1.76×10^{3}	5
Empty-6x6	65	3	100	1	2.3	3.47×10^{3}	7
Empty-8x8	145	3	100	1	3.0	9.28×10^{3}	11
Empty-16x16	785	3	100	1	7.9	1.22×10^{6}	27
DoorKey-5x5	265	6	100	1	3.6	3.85×10^{7}	11
DoorKey-6x6	1.30×10^{3}	6	100	1	5.0	5.00×10^{10}	14
DoorKey-8x8	9.24×10^{3}	6	100	1	6.6	2.27×10^{15}	17
DoorKey-16x16	3.58×10^{5}	6	100	1	14.1	1.04×10^{42}	29
MultiRoom-N2-S4	61	6	100	1	2.4	3.27×10^{5}	7
MultiRoom-N4-S5	5.70×10^{3}	6	100	1	17.2	2.29×10^{43}	36
MultiRoom-N6	1.12×10^{4}	6	100	1	24.8	3.81×10^{72}	46
KeyCorridorS3R1	169	6	100	1	4.1	2.10×10^{7}	15
KeyCorridorS3R2	4.90×10^{3}	6	100	1	4.7	2.48×10^{13}	15
KeyCorridorS3R3	1.08×10^{5}	6	100	1	5.5	6.27×10^{25}	17
KeyCorridorS4R3	6.03×10^{5}	6	100	1	9.1	1.66×10^{42}	20
Unlock	1.08×10^{3}	6	100	1	4.6	1.36×10^{7}	15
UnlockPickup	1.72×10^{4}	6	100	1	5.3	6.26×10^{17}	16
BlockedUnlockPickup	5.10×10^{5}	6	100	1	10.4	5.85×10^{35}	24
ObstructedMaze-1Dl	3.20×10^{3}	6	100	1	4.9	1.92×10^{10}	14
ObstructedMaze-1Dlh	4.35×10^{3}	6	100	1	5.5	7.25×10^{10}	15
ObstructedMaze-1Dlhb	6.73×10^{4}	6	100	1	9.1	6.05×10^{16}	21
FourRooms	1.04×10^{3}	6	100	1	5.1	7.25×10^{11}	15
LavaCrossingS9N1	173	6	100	1	3.7	3.19×10^{5}	14
LavaCrossingS9N2	133	6	100	1	4.1	3.65×10^{5}	14
LavaCrossingS9N3	53	6	100	1	4.5	9.99×10^{5}	13
LavaCrossingS11N5	85	6	100	1	10.4	1.74×10^{9}	21
SimpleCrossingS9N1	173	6	100	1	3.6	1.84×10^{5}	14
SimpleCrossingS9N2	133	6	100	1	3.9	1.41×10^{5}	14
SimpleCrossingS9N3	53	6	100	1	3.9	1.04×10^{5}	13

SimpleCrossingS11N5	85	6	100	1	6.9	2.71×10^7	21
LavaGapS5	21	6	100	1	1.8	1.19×10^4	6
LavaGapS6	53	6	100	1	3.0	1.15×10^5	9
LavaGapS7	85	6	100	1	3.6	4.03×10^5	11

G.4 Table of bounds and empirical sample complexities

This table lists all the sample complexity bounds we calculate for each MDP in Bridge along with the empirical sample complexities of PPO, DQN, and GORP.

	Sample complexity bounds					**Empirical sample complexities**		
MDP	Worst-case $(T\lceil A^T/2\rceil)$	Covering length (TL)	EPW (T^2A^W)	UCB (SAT)	Effective horizon (T^2A^H)	PPO	DQN	GORP
Alien$_{10}$	1.79×10^{13}	2.27×10^{13}	3.40×10^{9}	1.43×10^{8}	2.78×10^{6}	$> 5 \times 10^{6}$	6.30×10^{4}	2.70×10^{4}
Amidar$_{20}$	1.00×10^{21}	8.06×10^{12}	4.00×10^{14}	2.01×10^{7}	4.00×10^{14}	$> 5 \times 10^{6}$	2.21×10^{6}	$> 10^{8}$
Assault$_{10}$	1.41×10^{9}	4.56×10^{10}	5.76×10^{8}	5.04×10^{7}	1.95×10^{10}	2.68×10^{5}	3.26×10^{5}	3.35×10^{6}
Asterix$_{10}$	1.74×10^{10}	4.99×10^{11}	7.29×10^{4}	8.13×10^{6}	2.20×10^{5}	5.20×10^{4}	7.00×10^{4}	4.50×10^{3}
Asteroids$_{10}$	1.45×10^{12}	2.72×10^{13}	2.07×10^{12}	7.16×10^{8}	2.89×10^{13}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Atlantis$_{10}$	5.24×10^{6}	6.11×10^{4}	6.40×10^{3}	1.96×10^{3}	800	1.10×10^{4}	1.40×10^{4}	800
Atlantis$_{20}$	1.10×10^{13}	1.11×10^{10}	2.56×10^{4}	3.77×10^{4}	2.80×10^{5}	1.65×10^{5}	2.95×10^{5}	1.90×10^{4}
Atlantis$_{30}$	1.73×10^{19}	8.66×10^{13}	6.33×10^{16}	5.48×10^{5}	6.33×10^{16}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Atlantis$_{40}$	2.42×10^{25}	3.85×10^{16}	4.19×10^{8}	3.29×10^{6}	3.14×10^{11}	$> 5 \times 10^{6}$	9.39×10^{5}	1.56×10^{7}
Atlantis$_{50}$	3.17×10^{31}	2.13×10^{20}	1.89×10^{26}	1.13×10^{7}	1.93×10^{29}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Atlantis$_{70}$	4.88×10^{43}	3.58×10^{26}	6.21×10^{33}	3.96×10^{7}	1.04×10^{41}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
BankHeist$_{10}$	1.79×10^{13}	6.73×10^{14}	1.98×10^{13}	7.63×10^{8}	2.16×10^{6}	9.90×10^{4}	2.06×10^{6}	3.78×10^{4}
BattleZone$_{10}$	1.79×10^{13}	5.01×10^{9}	1.05×10^{7}	1.26×10^{7}	6.84×10^{4}	5.70×10^{4}	8.70×10^{4}	3.60×10^{3}
BeamRider$_{20}$	1.22×10^{20}	2.20×10^{14}	1.39×10^{12}	4.30×10^{6}	5.22×10^{5}	3.20×10^{4}	9.50×10^{4}	2.52×10^{4}
Bowling$_{30}$	3.32×10^{24}	5.18×10^{17}	5.53×10^{24}	4.51×10^{7}	9.72×10^{6}	1.26×10^{5}	1.39×10^{6}	6.48×10^{4}
Breakout$_{10}$	5.24×10^{6}	7.92×10^{7}	1.02×10^{5}	9.52×10^{3}	2.24×10^{4}	6.80×10^{4}	3.00×10^{4}	2.62×10^{3}
Breakout$_{20}$	1.10×10^{13}	2.03×10^{14}	2.68×10^{10}	1.02×10^{5}	6.83×10^{5}	4.80×10^{4}	1.23×10^{5}	9.07×10^{3}
Breakout$_{30}$	1.73×10^{19}	3.46×10^{20}	2.59×10^{20}	3.36×10^{5}	6.04×10^{10}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	2.50×10^{6}
Breakout$_{40}$	2.42×10^{25}	5.22×10^{26}	4.61×10^{20}	9.80×10^{5}	4.35×10^{12}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	5.22×10^{6}
Breakout$_{50}$	3.17×10^{31}	7.46×10^{32}	2.75×10^{15}	3.24×10^{6}	1.58×10^{11}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	5.68×10^{6}
Breakout$_{70}$	4.88×10^{43}	1.30×10^{45}	2.61×10^{40}	2.19×10^{7}	7.87×10^{32}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Breakout$_{100}$	8.03×10^{61}	2.23×10^{63}	1.43×10^{49}	5.25×10^{7}	2.33×10^{56}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Breakout$_{200}$	2.58×10^{122}	7.19×10^{123}	4.21×10^{69}	1.12×10^{8}	6.26×10^{73}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Centipede$_{10}$	1.79×10^{13}	7.13×10^{14}	6.12×10^{10}	2.38×10^{9}	3.57×10^{14}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
ChopperCommand$_{10}$	1.79×10^{13}	1.95×10^{12}	6.12×10^{10}	2.49×10^{8}	8.87×10^{6}	$> 5 \times 10^{6}$	1.85×10^{5}	6.45×10^{4}
CrazyClimber$_{20}$	1.22×10^{20}	8.03×10^{10}	1.72×10^{10}	3.19×10^{5}	3.35×10^{5}	2.18×10^{4}	3.00×10^{3}	1.08×10^{4}
CrazyClimber$_{30}$	6.36×10^{29}	6.46×10^{20}	1.35×10^{20}	1.25×10^{8}	5.11×10^{6}	6.50×10^{4}	5.00×10^{3}	1.04×10^{5}
DemonAttack$_{10}$	3.02×10^{8}	8.19×10^{9}	1.01×10^{9}	3.79×10^{6}	1.01×10^{9}	2.88×10^{5}	3.05×10^{5}	1.53×10^{6}
Enduro$_{10}$	1.74×10^{10}	6.95×10^{11}	3.87×10^{10}	2.29×10^{9}	3.49×10^{11}	3.49×10^{5}	4.57×10^{5}	9.57×10^{8}
FishingDerby$_{10}$	1.79×10^{13}	3.60×10^{13}	1.98×10^{13}	5.04×10^{7}	1.98×10^{13}	$> 5 \times 10^{6}$	1.32×10^{6}	2.10×10^{7}
Freeway$_{10}$	2.95×10^{5}	1.39×10^{6}	7.29×10^{4}	5.94×10^{3}	1.11×10^{4}	2.00×10^{3}	4.00×10^{3}	8.70×10^{3}
Freeway$_{20}$	3.49×10^{10}	2.29×10^{11}	6.38×10^{8}	1.89×10^{5}	1.40×10^{6}	2.00×10^{3}	3.00×10^{3}	1.87×10^{5}
Freeway$_{30}$	3.09×10^{15}	3.78×10^{15}	2.29×10^{15}	9.21×10^{5}	4.87×10^{6}	1.50×10^{5}	5.60×10^{4}	1.09×10^{6}
Freeway$_{40}$	2.43×10^{20}	1.06×10^{19}	8.01×10^{19}	2.50×10^{6}	9.36×10^{6}	2.83×10^{5}	8.20×10^{4}	2.84×10^{6}
Freeway$_{50}$	1.79×10^{25}	2.97×10^{23}	2.46×10^{24}	5.10×10^{6}	1.81×10^{7}	3.29×10^{5}	4.00×10^{4}	5.35×10^{6}
Freeway$_{70}$	8.76×10^{34}	2.73×10^{31}	1.51×10^{35}	1.47×10^{7}	1.11×10^{8}	1.14×10^{6}	1.85×10^{5}	4.53×10^{7}
Freeway$_{100}$	2.58×10^{49}	2.82×10^{43}	9.27×10^{35}	4.54×10^{7}	2.71×10^{30}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Freeway$_{200}$	2.66×10^{97}	2.52×10^{80}	1.48×10^{76}	3.80×10^{8}	2.00×10^{12}	$> 5 \times 10^{6}$	2.88×10^{6}	$> 10^{8}$
Frostbite$_{10}$	1.79×10^{13}	8.01×10^{11}	5.83×10^{5}	1.03×10^{7}	3.89×10^{5}	4.50×10^{4}	2.17×10^{5}	5.40×10^{3}
Gopher$_{30}$	1.86×10^{28}	2.29×10^{12}	2.95×10^{7}	1.87×10^{5}	4.32×10^{4}	2.30×10^{4}	3.20×10^{4}	1.44×10^{4}
Gopher$_{40}$	2.66×10^{37}	1.94×10^{14}	8.80×10^{14}	2.62×10^{6}	8.30×10^{11}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	1.31×10^{7}
Hero$_{10}$	1.79×10^{13}	1.85×10^{10}	5.83×10^{5}	8.80×10^{5}	1.80×10^{3}	1.20×10^{4}	7.00×10^{3}	3.60×10^{3}
IceHockey$_{10}$	1.79×10^{13}	1.01×10^{12}	1.89×10^{8}	4.56×10^{8}	1.18×10^{6}	9.00×10^{3}	2.00×10^{4}	7.20×10^{3}
Kangaroo$_{20}$	1.27×10^{26}	4.80×10^{18}	2.57×10^{16}	4.69×10^{7}	1.57×10^{7}	2.62×10^{5}	8.00×10^{4}	3.60×10^{4}
Kangaroo$_{30}$	6.83×10^{38}	3.21×10^{31}	6.69×10^{31}	3.16×10^{9}	1.20×10^{33}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
MontezumaRevenge$_{15}$	5.06×10^{19}	1.62×10^{16}	1.52×10^{21}	2.29×10^{6}	4.22×10^{12}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
MsPacman$_{20}$	1.22×10^{20}	1.63×10^{13}	1.26×10^{13}	3.32×10^{8}	1.26×10^{13}	$> 5 \times 10^{6}$	4.43×10^{6}	$> 10^{8}$
NameThisGame$_{20}$	3.66×10^{16}	1.88×10^{8}	3.11×10^{6}	7.25×10^{5}	3.80×10^{7}	5.08×10^{5}	9.60×10^{4}	2.30×10^{5}
Phoenix$_{10}$	5.37×10^{9}	1.45×10^{11}	1.68×10^{9}	3.72×10^{6}	1.07×10^{11}	$> 5 \times 10^{6}$	3.37×10^{5}	6.55×10^{6}
Pong$_{20}$	3.66×10^{16}	1.12×10^{12}	3.11×10^{6}	3.06×10^{4}	8.64×10^{4}	5.50×10^{4}	1.90×10^{4}	4.80×10^{3}
Pong$_{30}$	3.32×10^{24}	1.25×10^{17}	9.14×10^{16}	3.61×10^{5}	4.66×10^{9}	1.41×10^{5}	9.70×10^{4}	8.40×10^{7}
Pong$_{40}$	2.67×10^{32}	1.18×10^{22}	1.26×10^{21}	3.85×10^{6}	2.35×10^{14}	4.93×10^{5}	3.38×10^{5}	2.49×10^{7}
Pong$_{50}$	2.02×10^{40}	1.02×10^{27}	7.11×10^{22}	3.76×10^{7}	4.71×10^{18}	$> 5 \times 10^{6}$	1.03×10^{6}	$> 10^{8}$
Pong$_{70}$	1.03×10^{56}	6.08×10^{36}	1.39×10^{23}	1.21×10^{9}	3.66×10^{25}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Pong$_{100}$	3.27×10^{79}	2.05×10^{51}	2.84×10^{23}	2.08×10^{10}	3.60×10^{58}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
PrivateEye$_{10}$	1.79×10^{13}	8.64×10^{9}	1.10×10^{12}	2.32×10^{6}	1.39×10^{7}	4.20×10^{4}	2.50×10^{4}	7.74×10^{4}
Qbert$_{10}$	3.02×10^{8}	3.80×10^{6}	7.78×10^{5}	1.73×10^{4}	7.78×10^{5}	3.39×10^{5}	1.30×10^{4}	2.41×10^{5}
Qbert$_{20}$	3.66×10^{16}	2.36×10^{15}	4.03×10^{9}	4.51×10^{8}	4.99×10^{13}	5.75×10^{5}	2.95×10^{5}	1.71×10^{7}
RoadRunner$_{10}$	1.79×10^{13}	7.34×10^{14}	1.98×10^{13}	4.26×10^{9}	2.52×10^{14}	8.10×10^{4}	1.32×10^{5}	2.24×10^{6}
Seaquest$_{10}$	1.79×10^{13}	4.15×10^{9}	1.89×10^{8}	9.84×10^{5}	3.60×10^{3}	4.00×10^{3}	1000	3.58×10^{3}
Skiing$_{10}$	2.95×10^{5}	6.83×10^{6}	5.90×10^{6}	5.25×10^{5}	5.90×10^{6}	$> 5 \times 10^{6}$	3.58×10^{6}	8.53×10^{6}
SpaceInvaders$_{10}$	3.02×10^{8}	4.38×10^{6}	2.16×10^{4}	5.96×10^{4}	2.28×10^{4}	4.40×10^{4}	1.30×10^{4}	1.20×10^{3}
Tennis$_{10}$	1.79×10^{13}	2.29×10^{12}	3.40×10^{9}	6.82×10^{7}	1.01×10^{8}	$> 5 \times 10^{6}$	5.94×10^{5}	1.17×10^{6}
TimePilot$_{10}$	5.00×10^{10}	5.76×10^{7}	1.00×10^{8}	5.03×10^{5}	8.83×10^{5}	3.30×10^{4}	2.06×10^{5}	7.00×10^{3}

Tutankham$_{10}$	5.37×10^{9}	1.68×10^{10}	1.34×10^{10}	1.33×10^{6}	8.45×10^{8}	3.19×10^{5}	4.98×10^{5}	2.04×10^{5}
VideoPinball$_{10}$	1.74×10^{10}	5.10×10^{11}	4.30×10^{9}	1.13×10^{7}	1.38×10^{8}	$> 5 \times 10^{6}$	1.58×10^{6}	5.64×10^{4}
WizardOfWor$_{20}$	1.00×10^{21}	3.78×10^{14}	4.00×10^{15}	1.78×10^{6}	4.00×10^{15}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
BigFish$^{E0}_{10}$	1.74×10^{10}	2.26×10^{11}	4.30×10^{9}	2.08×10^{6}	3.87×10^{10}	$> 5 \times 10^{6}$	1.21×10^{5}	1.14×10^{7}
BigFish$^{E1}_{10}$	1.74×10^{10}	4.57×10^{11}	6.56×10^{5}	2.47×10^{6}	4.68×10^{4}	5.30×10^{4}	1.10×10^{4}	1.79×10^{3}
BigFish$^{E2}_{10}$	1.74×10^{10}	5.26×10^{11}	3.49×10^{11}	1.76×10^{7}	3.49×10^{11}	$> 5 \times 10^{6}$	1.58×10^{6}	$> 10^{8}$
BigFish$^{H0}_{10}$	1.74×10^{10}	4.19×10^{11}	4.78×10^{8}	8.33×10^{5}	1.30×10^{6}	$> 5 \times 10^{6}$	3.10×10^{5}	3.99×10^{4}
Chaser$^{E0}_{20}$	1.22×10^{20}	2.32×10^{18}	8.24×10^{16}	1.46×10^{8}	8.89×10^{9}	4.42×10^{5}	2.03×10^{6}	2.38×10^{6}
Chaser$^{E1}_{20}$	1.22×10^{20}	3.23×10^{18}	1.55×10^{11}	7.16×10^{7}	6.27×10^{6}	$> 5 \times 10^{6}$	3.33×10^{6}	2.30×10^{5}
Chaser$^{E2}_{20}$	1.22×10^{20}	2.67×10^{18}	1.39×10^{12}	9.05×10^{7}	4.95×10^{14}	2.42×10^{5}	8.17×10^{5}	7.87×10^{6}
Chaser$^{H0}_{20}$	1.22×10^{20}	2.07×10^{18}	5.40×10^{20}	1.58×10^{8}	4.86×10^{21}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Climber$^{E0}_{10}$	1.74×10^{10}	2.67×10^{11}	4.78×10^{8}	2.18×10^{7}	1.59×10^{5}	2.50×10^{4}	1.43×10^{5}	4.50×10^{3}
Climber$^{E1}_{10}$	1.74×10^{10}	8.46×10^{10}	3.87×10^{10}	1.06×10^{7}	5.89×10^{5}	7.30×10^{4}	3.03×10^{5}	3.21×10^{4}
Climber$^{E2}_{10}$	1.74×10^{10}	8.43×10^{10}	5.31×10^{7}	1.01×10^{7}	5.04×10^{4}	8.70×10^{4}	9.00×10^{4}	5.67×10^{4}
Climber$^{H0}_{10}$	1.74×10^{10}	2.66×10^{11}	3.49×10^{11}	2.09×10^{7}	2.09×10^{8}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	1.94×10^{7}
CoinRun$^{E0}_{10}$	1.74×10^{10}	1.77×10^{11}	4.30×10^{9}	2.01×10^{7}	9.33×10^{5}	1.14×10^{5}	2.03×10^{5}	8.35×10^{5}
CoinRun$^{E1}_{10}$	1.74×10^{10}	4.05×10^{10}	4.78×10^{8}	5.61×10^{6}	2.88×10^{5}	5.40×10^{4}	8.60×10^{4}	1.72×10^{5}
CoinRun$^{E2}_{10}$	1.74×10^{10}	1.30×10^{11}	3.87×10^{10}	1.57×10^{7}	5.88×10^{6}	9.32×10^{5}	4.91×10^{5}	5.08×10^{6}
CoinRun$^{H0}_{10}$	1.74×10^{10}	1.34×10^{11}	4.78×10^{8}	2.45×10^{7}	1.44×10^{4}	4.00×10^{3}	6.00×10^{3}	1.24×10^{4}
Dodgeball$^{E0}_{10}$	5.00×10^{10}	1.47×10^{12}	1.00×10^{9}	1.19×10^{7}	1.51×10^{6}	1.31×10^{5}	9.80×10^{4}	1.76×10^{4}
Dodgeball$^{E1}_{10}$	5.00×10^{10}	1.29×10^{12}	1.00×10^{12}	1.95×10^{6}	2.99×10^{7}	3.29×10^{5}	4.46×10^{5}	5.08×10^{6}
Dodgeball$^{E2}_{10}$	5.00×10^{10}	1.34×10^{12}	1.00×10^{8}	3.42×10^{6}	4.98×10^{5}	1.74×10^{5}	1.26×10^{5}	1.59×10^{4}
Dodgeball$^{H0}_{10}$	5.00×10^{10}	1.42×10^{12}	1.00×10^{10}	7.24×10^{6}	1.00×10^{12}	$> 5 \times 10^{6}$	4.88×10^{6}	8.16×10^{6}
FruitBot$^{E0}_{40}$	2.96×10^{39}	9.28×10^{8}	5.58×10^{12}	8.28×10^{4}	9.80×10^{20}	$> 5 \times 10^{6}$	1.44×10^{5}	4.00×10^{7}
FruitBot$^{E1}_{40}$	2.96×10^{39}	2.40×10^{13}	2.40×10^{20}	1.36×10^{5}	1.74×10^{26}	$> 5 \times 10^{6}$	3.61×10^{5}	$> 10^{8}$
FruitBot$^{E2}_{40}$	2.96×10^{39}	4.47×10^{6}	1.30×10^{5}	5.80×10^{4}	1.43×10^{6}	2.50×10^{4}	2.00×10^{3}	1.01×10^{4}
FruitBot$^{H0}_{40}$	2.96×10^{39}	2.62×10^{15}	1.28×10^{26}	2.23×10^{5}	1.17×10^{21}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Heist$^{E1}_{10}$	1.74×10^{10}	4.96×10^{11}	3.49×10^{11}	7.45×10^{6}	1.06×10^{8}	$> 5 \times 10^{6}$	6.86×10^{5}	7.09×10^{7}
Jumper$^{H0}_{10}$	1.74×10^{10}	5.11×10^{11}	3.49×10^{11}	1.17×10^{7}	1.41×10^{10}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Jumper$^{E0}_{20}$	1.22×10^{20}	1.77×10^{21}	3.60×10^{3}	2.15×10^{7}	3.60×10^{3}	224	20	374
Jumper$^{E1}_{20}$	1.22×10^{20}	4.02×10^{21}	3.24×10^{4}	1.49×10^{8}	7.20×10^{3}	888	46	4.96×10^{3}
Jumper$^{E2}_{20}$	1.22×10^{20}	2.59×10^{20}	2.36×10^{7}	2.48×10^{8}	8.64×10^{4}	2.56×10^{4}	7.00×10^{3}	3.42×10^{4}
Jumper$^{EX}_{20}$	1.22×10^{20}	2.18×10^{21}	6.67×10^{18}	6.13×10^{8}	1.24×10^{15}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Leaper$^{E1}_{20}$	1.22×10^{20}	2.96×10^{21}	7.41×10^{17}	1.90×10^{6}	2.16×10^{13}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Leaper$^{E2}_{20}$	1.22×10^{20}	3.69×10^{21}	1.72×10^{10}	3.97×10^{7}	4.90×10^{6}	1.27×10^{5}	2.34×10^{5}	2.94×10^{6}
Leaper$^{H0}_{20}$	1.22×10^{20}	3.07×10^{21}	8.24×10^{16}	3.08×10^{6}	1.27×10^{15}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Leaper$^{EX}_{20}$	1.22×10^{20}	3.23×10^{21}	7.41×10^{17}	5.96×10^{6}	7.04×10^{14}	$> 5 \times 10^{6}$	2.61×10^{6}	$> 10^{8}$
Maze$^{E0}_{30}$	6.36×10^{29}	4.95×10^{6}	8.10×10^{3}	6.59×10^{4}	1.62×10^{4}	1000	66	4.57×10^{3}
Maze$^{E1}_{30}$	6.36×10^{29}	2.71×10^{24}	7.98×10^{24}	3.95×10^{5}	3.44×10^{16}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Maze$^{E2}_{30}$	6.36×10^{29}	1.35×10^{9}	6.56×10^{5}	1.77×10^{5}	1.62×10^{4}	1000	411	1.36×10^{4}
Maze$^{H0}_{30}$	6.36×10^{29}	2.48×10^{25}	1.85×10^{17}	4.73×10^{5}	3.01×10^{10}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Maze$^{EX}_{100}$	1.33×10^{97}	4.27×10^{65}	3.33×10^{76}	3.34×10^{7}	2.67×10^{42}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Miner$^{E0}_{10}$	1.74×10^{10}	4.84×10^{11}	4.30×10^{9}	5.37×10^{6}	4.58×10^{5}	9.40×10^{4}	7.40×10^{4}	4.69×10^{5}
Miner$^{E1}_{10}$	1.74×10^{10}	4.89×10^{11}	3.87×10^{10}	6.17×10^{6}	9.34×10^{6}	3.00×10^{5}	3.77×10^{5}	1.74×10^{6}
Miner$^{E2}_{10}$	1.74×10^{10}	4.82×10^{11}	7.29×10^{4}	5.00×10^{6}	1.80×10^{3}	1.02×10^{4}	2.00×10^{3}	1.79×10^{3}
Miner$^{H0}_{10}$	1.74×10^{10}	5.04×10^{11}	4.78×10^{8}	9.35×10^{6}	1.92×10^{5}	4.90×10^{4}	4.03×10^{5}	4.50×10^{3}
Ninja$^{E0}_{10}$	6.89×10^{11}	2.67×10^{12}	1.06×10^{12}	2.62×10^{7}	8.49×10^{6}	2.64×10^{5}	$> 5 \times 10^{6}$	5.32×10^{6}
Ninja$^{E1}_{10}$	6.89×10^{11}	2.82×10^{12}	8.16×10^{10}	6.33×10^{7}	2.43×10^{6}	4.45×10^{5}	4.23×10^{5}	1.85×10^{6}
Ninja$^{E2}_{10}$	6.89×10^{11}	2.67×10^{12}	1.06×10^{12}	2.74×10^{7}	5.97×10^{9}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Ninja$^{H0}_{10}$	6.89×10^{11}	2.47×10^{12}	1.38×10^{13}	8.60×10^{6}	3.57×10^{9}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Plunder$^{E0}_{10}$	5.00×10^{10}	1.26×10^{12}	1.00×10^{7}	1.55×10^{6}	2.00×10^{3}	1.15×10^{4}	1.00×10^{4}	5.00×10^{3}
Plunder$^{E1}_{10}$	5.00×10^{10}	1.29×10^{12}	1.00×10^{5}	2.06×10^{6}	1000	2.00×10^{3}	3.00×10^{3}	2.00×10^{3}
Plunder$^{E2}_{10}$	5.00×10^{10}	1.22×10^{12}	1.00×10^{9}	9.99×10^{5}	1.96×10^{5}	6.40×10^{4}	1.13×10^{5}	2.10×10^{4}
Plunder$^{H0}_{10}$	5.00×10^{10}	1.20×10^{12}	1.00×10^{7}	8.28×10^{5}	2.00×10^{3}	3.00×10^{3}	6.00×10^{3}	4.00×10^{3}
StarPilot$^{E0}_{10}$	1.30×10^{11}	4.09×10^{12}	2.36×10^{11}	3.56×10^{7}	1.99×10^{9}	$> 5 \times 10^{6}$	6.36×10^{5}	7.02×10^{5}
StarPilot$^{E1}_{10}$	1.30×10^{11}	3.93×10^{12}	2.14×10^{10}	1.94×10^{7}	6.90×10^{5}	5.20×10^{4}	1.37×10^{5}	6.24×10^{3}
StarPilot$^{E2}_{10}$	1.30×10^{11}	3.77×10^{12}	1.95×10^{9}	1.05×10^{7}	2.36×10^{11}	9.90×10^{4}	2.67×10^{5}	2.61×10^{5}
StarPilot$^{H0}_{10}$	1.30×10^{11}	4.02×10^{12}	2.14×10^{10}	2.71×10^{7}	1.04×10^{10}	$> 5 \times 10^{6}$	5.12×10^{5}	2.15×10^{7}
Empty-5x5	2.58×10^{49}	1.76×10^{5}	2.43×10^{6}	1.11×10^{4}	6.00×10^{4}	1.02×10^{3}	232	3.86×10^{4}
Empty-6x6	2.58×10^{49}	3.47×10^{5}	2.19×10^{7}	1.95×10^{4}	1.20×10^{5}	2.05×10^{3}	543	4.50×10^{4}
Empty-8x8	2.58×10^{49}	9.28×10^{5}	1.77×10^{9}	4.35×10^{4}	2.70×10^{5}	1.57×10^{4}	7.90×10^{4}	8.49×10^{4}
Empty-16x16	2.58×10^{49}	1.22×10^{8}	7.63×10^{16}	2.35×10^{5}	5.93×10^{7}	1.78×10^{5}	3.81×10^{5}	3.02×10^{7}
DoorKey-5x5	3.27×10^{79}	3.85×10^{9}	3.63×10^{12}	1.59×10^{5}	6.30×10^{6}	7.01×10^{4}	1.45×10^{5}	8.17×10^{5}
DoorKey-6x6	3.27×10^{79}	5.00×10^{12}	7.84×10^{14}	7.81×10^{5}	7.51×10^{7}	7.22×10^{5}	2.85×10^{5}	6.79×10^{6}
DoorKey-8x8	3.27×10^{79}	2.27×10^{17}	1.69×10^{17}	5.55×10^{6}	1.34×10^{9}	8.35×10^{5}	7.23×10^{5}	$> 10^{8}$
DoorKey-16x16	3.27×10^{79}	1.04×10^{44}	3.68×10^{26}	2.15×10^{8}	1.01×10^{15}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
MultiRoom-N2-S4	3.27×10^{79}	3.27×10^{7}	2.80×10^{9}	3.66×10^{4}	7.80×10^{5}	3.63×10^{4}	1.23×10^{5}	9.98×10^{4}
MultiRoom-N4-S5	3.27×10^{79}	2.29×10^{45}	1.03×10^{32}	3.42×10^{6}	2.33×10^{17}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
MultiRoom-N6	3.27×10^{79}	3.81×10^{74}	6.24×10^{39}	6.72×10^{6}	1.84×10^{23}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
KeyCorridorS3R1	3.27×10^{79}	2.10×10^{9}	4.70×10^{15}	1.01×10^{5}	1.46×10^{7}	3.31×10^{5}	1.72×10^{5}	2.82×10^{6}
KeyCorridorS3R2	3.27×10^{79}	2.48×10^{15}	4.70×10^{15}	2.94×10^{6}	4.21×10^{7}	6.32×10^{5}	3.40×10^{5}	6.59×10^{6}
KeyCorridorS3R3	3.27×10^{79}	6.27×10^{27}	1.69×10^{17}	6.48×10^{7}	1.80×10^{8}	1.01×10^{6}	1.33×10^{6}	3.01×10^{7}
KeyCorridorS4R3	3.27×10^{79}	1.66×10^{44}	3.66×10^{19}	3.62×10^{8}	1.17×10^{11}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
Unlock	3.27×10^{79}	1.36×10^{9}	4.70×10^{15}	6.51×10^{5}	3.76×10^{7}	1.79×10^{5}	2.70×10^{5}	5.36×10^{6}
UnlockPickup	3.27×10^{79}	6.26×10^{19}	2.82×10^{16}	1.03×10^{7}	1.31×10^{8}	6.00×10^{5}	3.58×10^{5}	1.37×10^{7}
BlockedUnlockPickup	3.27×10^{79}	5.85×10^{37}	4.74×10^{22}	3.06×10^{8}	1.25×10^{12}	1.81×10^{6}	$> 5 \times 10^{6}$	$> 10^{8}$
ObstructedMaze-1Dl	3.27×10^{79}	1.92×10^{12}	7.84×10^{14}	1.92×10^{6}	6.09×10^{7}	4.31×10^{5}	4.05×10^{5}	6.09×10^{6}
ObstructedMaze-1Dlh	3.27×10^{79}	7.25×10^{12}	4.70×10^{15}	2.61×10^{6}	1.77×10^{8}	7.32×10^{5}	1.86×10^{6}	2.12×10^{7}
ObstructedMaze-1Dlhb	3.27×10^{79}	6.05×10^{18}	2.19×10^{20}	4.04×10^{7}	1.28×10^{11}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
FourRooms	3.27×10^{79}	7.25×10^{13}	4.70×10^{15}	6.22×10^{5}	9.54×10^{7}	$> 5 \times 10^{6}$	3.26×10^{5}	1.00×10^{7}

LavaCrossingS9N1	3.27×10^{79}	3.19×10^{7}	7.84×10^{14}	1.04×10^{5}	7.80×10^{6}	1.26×10^{5}	1.80×10^{6}	9.86×10^{5}
LavaCrossingS9N2	3.27×10^{79}	3.65×10^{7}	7.84×10^{14}	7.98×10^{4}	1.63×10^{7}	2.93×10^{5}	1.55×10^{6}	2.09×10^{6}
LavaCrossingS9N3	3.27×10^{79}	9.99×10^{7}	1.31×10^{14}	3.18×10^{4}	3.32×10^{7}	9.70×10^{4}	1.56×10^{6}	5.24×10^{6}
LavaCrossingS11N5	3.27×10^{79}	1.74×10^{11}	2.19×10^{20}	5.10×10^{4}	1.19×10^{12}	$> 5 \times 10^{6}$	$> 5 \times 10^{6}$	$> 10^{8}$
SimpleCrossingS9N1	3.27×10^{79}	1.84×10^{7}	7.84×10^{14}	1.04×10^{5}	6.30×10^{6}	7.65×10^{4}	2.34×10^{5}	1.02×10^{6}
SimpleCrossingS9N2	3.27×10^{79}	1.41×10^{7}	7.84×10^{14}	7.98×10^{4}	1.01×10^{7}	9.63×10^{4}	2.15×10^{5}	1.55×10^{6}
SimpleCrossingS9N3	3.27×10^{79}	1.04×10^{7}	1.31×10^{14}	3.18×10^{4}	1.16×10^{7}	3.91×10^{4}	1.59×10^{5}	2.64×10^{6}
SimpleCrossingS11N5	3.27×10^{79}	2.71×10^{9}	2.19×10^{20}	5.10×10^{4}	2.49×10^{9}	5.42×10^{5}	1.26×10^{6}	$> 10^{8}$
LavaGapS5	3.27×10^{79}	1.19×10^{6}	4.67×10^{8}	1.26×10^{4}	2.40×10^{5}	3.69×10^{4}	1.19×10^{5}	4.33×10^{4}
LavaGapS6	3.27×10^{79}	1.15×10^{7}	1.01×10^{11}	3.18×10^{4}	2.28×10^{6}	1.49×10^{5}	2.35×10^{5}	1.96×10^{5}
LavaGapS7	3.27×10^{79}	4.03×10^{7}	3.63×10^{12}	5.10×10^{4}	6.78×10^{6}	2.41×10^{5}	2.38×10^{5}	8.20×10^{5}

G.5 Table of returns

This table lists the returns for the random and optimal policies in each MDP as well as the final returns achieved by PPO, DQN, and GORP. We ran PPO and DQN for 5 million timesteps in each environment, and GORP for both 5 million and 100 million timesteps.

	Returns					
MDP	Random policy	Optimal policy	PPO	DQN	GORP (5M)	(100M)
Alien$_{10}$	74.74	160	160	160	160	160
Amidar$_{20}$	6.77	109	68	68	65	68
Assault$_{10}$	8.66	126	126	126	126	126
Asterix$_{10}$	99.24	400	400	400	400	400
Asteroids$_{10}$	15.08	320	170	190	220	220
Atlantis$_{10}$	24.22	200	200	200	200	200
Atlantis$_{20}$	124.25	1,200	1,200	1,200	1,200	1,200
Atlantis$_{30}$	145.46	4,300	1,600	1,600	1,700	4,000
Atlantis$_{40}$	146.68	5,500	5,500	5,400	5,400	5,500
Atlantis$_{50}$	146.84	7,800	7,600	5,500	5,600	7,700
Atlantis$_{70}$	146.86	11,100	8,900	8,900	8,200	10,900
BankHeist$_{10}$	0.36	30	30	30	30	30
BattleZone$_{10}$	115.32	2,000	2,000	2,000	2,000	2,000
BeamRider$_{20}$	23.77	132	132	132	132	132
Bowling$_{30}$	1.82	9	9	9	9	9
Breakout$_{10}$	0.13	2	2	2	2	2
Breakout$_{20}$	0.17	4	4	4	4	4
Breakout$_{30}$	0.18	9	4	7	9	9
Breakout$_{40}$	0.18	11	9	9	9	11
Breakout$_{50}$	0.18	13	10	10	10	13
Breakout$_{70}$	0.18	22	13	14	4	17
Breakout$_{100}$	0.18	44	14	14	4	19
Breakout$_{200}$	0.18	60	14	14	4	17
Centipede$_{10}$	141.37	1,923	922	1,632	1,511	1,621
ChopperCommand$_{10}$	117.25	600	600	600	600	600
CrazyClimber$_{20}$	125	400	400	400	400	400
CrazyClimber$_{30}$	248.16	900	900	900	900	900
DemonAttack$_{10}$	8.31	50	50	50	50	50
Enduro$_{10}$	0.04	6	6	6	4	6
FishingDerby$_{10}$	0.16	8	6	8	6	8
Freeway$_{10}$	0.01	1	1	1	1	1
Freeway$_{20}$	0.02	2	2	2	2	2
Freeway$_{30}$	0.07	4	4	4	4	4
Freeway$_{40}$	0.1	5	5	5	5	5
Freeway$_{50}$	0.14	6	6	6	5	6
Freeway$_{70}$	0.23	9	9	9	7	9
Freeway$_{100}$	0.33	13	12	12	9	12
Freeway$_{200}$	0.72	25	24	25	12	20
Frostbite$_{10}$	9.13	70	70	70	70	70
Gopher$_{30}$	1.43	20	20	20	20	20
Gopher$_{40}$	10.05	180	100	100	100	180
Hero$_{10}$	8.18	75	75	75	75	75
IceHockey$_{10}$	0.01	1	1	1	1	1
Kangaroo$_{20}$	1.02	200	200	200	200	200

Kangaroo$_{30}$	3.84	500	200	200	400	400
MontezumaRevenge$_{15}$	0	100	0	0	0	0
MsPacman$_{20}$	148.91	480	460	470	470	470
NameThisGame$_{20}$	11.22	180	180	180	180	180
Phoenix$_{10}$	23.43	260	260	260	260	260
Pong$_{20}$	−2.71	−1	−1	−1	−1	−1
Pong$_{30}$	−4.25	−1	−1	−1	−1	−1
Pong$_{40}$	−5.99	0	−1	0	0	0
Pong$_{50}$	−7.85	1	−1	1	0	0
Pong$_{70}$	−11.47	2	−1	0	−1	0
Pong$_{100}$	−16.91	4	−1	−1	−1	1
PrivateEye$_{10}$	0.11	100	100	100	100	100
Qbert$_{10}$	50.95	425	425	425	425	425
Qbert$_{20}$	87.21	675	675	675	650	675
RoadRunner$_{10}$	0.05	600	600	600	600	600
Seaquest$_{10}$	0.78	20	20	20	20	20
Skiing$_{10}$	−12,652.16	−7,066	−8,149	−7,078	−7,069	−7,066
SpaceInvaders$_{10}$	3.46	40	40	40	40	40
Tennis$_{10}$	−0.75	1	0	1	1	1
TimePilot$_{10}$	8.17	200	200	200	200	200
Tutankham$_{10}$	0.07	23	23	23	23	23
VideoPinball$_{10}$	108.99	4,100	3,000	4,100	4,100	4,100
WizardOfWor$_{20}$	11.5	500	100	100	300	400
Bigfish$^{E0}_{10}$	0.37	4	4	4	4	4
Bigfish$^{E1}_{10}$	0.73	3	3	3	3	3
Bigfish$^{E2}_{10}$	2.98	8	7	8	7	7
Bigfish$^{H0}_{10}$	0.03	3	3	3	3	3
Chaser$^{E0}_{20}$	0.38	0.88	0.88	0.88	1	1
Chaser$^{E1}_{20}$	0.33	0.88	0.88	0.88	1	1
Chaser$^{E2}_{20}$	0.37	0.88	0.88	0.88	1	1
Chaser$^{H0}_{20}$	0.39	0.88	0.88	0.84	1	1
Climber$^{E0}_{10}$	0.14	2	2	2	2	2
Climber$^{E1}_{10}$	0.02	2	2	2	2	2
Climber$^{E2}_{10}$	0.03	11	11	11	11	11
Climber$^{H0}_{10}$	0.31	2	1	1	1	2
Coinrun$^{E0}_{10}$	0	10	10	10	10	10
Coinrun$^{E1}_{10}$	0	10	10	0	10	10
Coinrun$^{E2}_{10}$	0	10	10	0	10	10
Coinrun$^{H0}_{10}$	0.08	10	10	10	10	10
Dodgeball$^{E0}_{10}$	0.08	16	16	16	16	16
Dodgeball$^{E1}_{10}$	0.24	8	8	8	6	8
Dodgeball$^{E2}_{10}$	0.1	8	8	8	8	8
Dodgeball$^{H0}_{10}$	0.04	8	6	6	6	8
Fruitbot$^{E0}_{40}$	−4.04	5	2	5	2	5
Fruitbot$^{E1}_{40}$	−2.09	7	4	7	4	5
Fruitbot$^{E2}_{40}$	−5.84	1	1	1	1	1
Fruitbot$^{H0}_{40}$	−0.42	10	5	10	0	9
Heist$^{E1}_{10}$	0	10	0	0	0	10
Jumper$^{H0}_{10}$	0	10	0	0	0	0
Jumper$^{E0}_{20}$	4.99	10	10	10	10	10
Jumper$^{E1}_{20}$	4.96	10	10	10	10	10
Jumper$^{E2}_{20}$	0.06	10	10	10	10	10
Jumper$^{EX}_{20}$	0	10	0	0	0	0
Leaper$^{E1}_{20}$	0	10	0	0	0	0
Leaper$^{E2}_{20}$	0	10	10	10	10	10
Leaper$^{H0}_{20}$	0	10	0	0	0	0
Leaper$^{EX}_{20}$	0	10	0	0	0	0
Maze$^{E0}_{30}$	6.1	10	10	10	10	10
Maze$^{E1}_{30}$	0	10	0	0	0	0
Maze$^{E2}_{30}$	5.65	10	10	10	10	10
Maze$^{H0}_{30}$	0	10	0	0	0	0

Maze$^{\text{EX}}_{100}$	0	10	0	0	0	0
Miner$^{\text{E0}}_{10}$	0	1	1	1	1	1
Miner$^{\text{E1}}_{10}$	0.05	2	2	2	2	2
Miner$^{\text{E2}}_{10}$	0.11	1	1	1	1	1
Miner$^{\text{H0}}_{10}$	0.04	3	3	3	3	3
Ninja$^{\text{E0}}_{10}$	0	10	10	0	10	10
Ninja$^{\text{E1}}_{10}$	0	10	10	0	10	10
Ninja$^{\text{E2}}_{10}$	0	10	0	0	0	0
Ninja$^{\text{H0}}_{10}$	0	10	0	0	0	0
Plunder$^{\text{E0}}_{10}$	0.06	1	1	1	1	1
Plunder$^{\text{E1}}_{10}$	0.17	1	1	1	1	1
Plunder$^{\text{E2}}_{10}$	0	1	1	1	1	1
Plunder$^{\text{H0}}_{10}$	0.07	1	1	1	1	1
Starpilot$^{\text{E0}}_{10}$	0.23	10	9	10	10	10
Starpilot$^{\text{E1}}_{10}$	0.14	5	5	5	5	5
Starpilot$^{\text{E2}}_{10}$	0.08	5	5	5	5	5
Starpilot$^{\text{H0}}_{10}$	0.17	5	4	5	4	5
Empty-5x5	0.76	1	1	1	1	1
Empty-6x6	0.5	1	1	1	1	1
Empty-8x8	0.18	1	1	1	1	1
Empty-16x16	0	1	1	0	0	1
DoorKey-5x5	0.01	1	1	1	1	1
DoorKey-6x6	0	1	1	1	0	1
DoorKey-8x8	0	1	0	0	0	0
DoorKey-16x16	0	1	0	0	0	0
MultiRoom-N2-S4	0.09	1	1	1	1	1
MultiRoom-N4-S5	0	1	0	0	0	0
MultiRoom-N6	0	1	0	0	0	0
KeyCorridorS3R1	0	1	1	1	1	1
KeyCorridorS3R2	0	1	1	1	0	1
KeyCorridorS3R3	0	1	0	0	0	1
KeyCorridorS4R3	0	1	0	0	0	0
Unlock	0	1	1	1	1	1
UnlockPickup	0	1	1	0	0	1
BlockedUnlockPickup	0	1	0	0	0	0
ObstructedMaze-1Dl	0	1	1	0	0	1
ObstructedMaze-1Dlh	0	1	0	0	0	1
ObstructedMaze-1Dlhb	0	1	0	0	0	0
FourRooms	0	1	1	0	0	1
LavaCrossingS9N1	0	1	1	1	1	1
LavaCrossingS9N2	0	1	1	1	1	1
LavaCrossingS9N3	0	1	1	0	0	1
LavaCrossingS11N5	0	1	0	0	0	0
SimpleCrossingS9N1	0.01	1	1	1	1	1
SimpleCrossingS9N2	0	1	1	1	1	1
SimpleCrossingS9N3	0	1	1	1	1	1
SimpleCrossingS11N5	0	1	0	0	0	0
LavaGapS5	0.07	1	1	1	1	1
LavaGapS6	0.02	1	1	1	1	1
LavaGapS7	0.01	1	1	1	1	1

G.6 Learning curves for Bridge MDPs

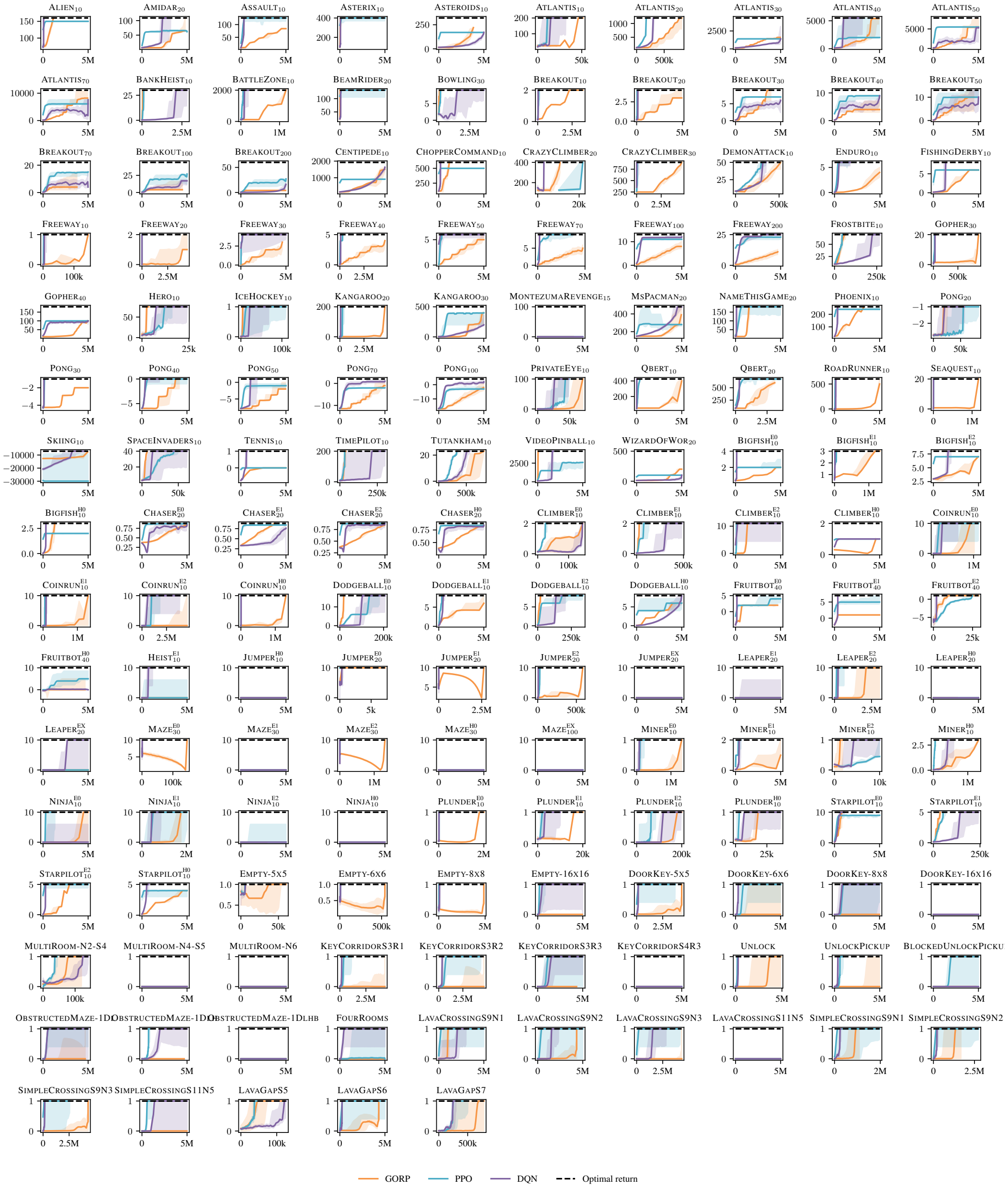


Figure 10: Learning curves for PPO, DQN, and GORP on all the MDPs in Bridge. Solid lines show the median return (over multiple random seeds) of the policies learned by each algorithm throughout training. We use 5 random seeds for PPO and DQN and 101 random seeds for GORP. The shaded region shows the range of returns over all random seeds for PPO and DQN, and shows the range from the 10th to 90th percentile over random seeds for GORP. The optimal return in each environment is shown as the dashed black line.