The Sample Complexity of Teaching-by-Reinforcement on Q-Learning

Xuezhou Zhang,[1] **Shubham Kumar Bharti,** [1] **Yuzhe Ma** [1] **Adish Singla** [2] **Xiaojin Zhu**[1]
[1] UW Madison
[2] MPI-SWS

Abstract

We study the sample complexity of teaching, termed as "teaching dimension" (TDim) in the literature, for the *teaching-by-reinforcement* paradigm, where the teacher guides the student through rewards. This is distinct from the *teaching-by-demonstration* paradigm motivated by robotics applications, where the teacher teaches by providing demonstrations of state/action trajectories. The teaching-by-reinforcement paradigm applies to a wider range of real-world settings where a demonstration is inconvenient, but has not been studied systematically. In this paper, we focus on a specific family of reinforcement learning algorithms, Q-learning, and characterize the TDim under different teachers with varying control power over the environment, and present matching optimal teaching algorithms. Our TDim results provide the minimum number of samples needed for reinforcement learning, and we discuss their connections to standard PAC-style RL sample complexity and teaching-by-demonstration sample complexity results. Our teaching algorithms have the potential to speed up RL agent learning in applications where a helpful teacher is available.

1 Introduction

In recent years, reinforcement learning (RL) has seen applications in a wide variety of domains, such as games [31, 24], robotics control [16, 2] and healthcare [17, 30]. One of the fundamental questions in RL is to understand the sample complexity of learning, i.e. the amount of training needed for an agent to learn to perform a task. In the most prevalent RL setting, an agent learns through continuous interaction with the environment and learns the optimal policy from natural reward signals. For standard algorithms such as Q-learning, naive interaction with MDP suffers exp complexity [19]. In contrast, many real-world RL scenarios involve a knowledgable (or even omniscient) teacher who aims at guiding the agent to learn the policy faster. For example, in the educational domain, a human student can be modeled as an RL agent, and a teacher will design a minimal curriculum to convey knowledge (policy) to the student (agent) [6].

In the context of reinforcement learning, teaching has traditionally been studied extensively under the scheme of *teaching-by-demonstration (TbD)*, where the teacher provides demonstrations of state/action trajectories under a good policy, and the agent aims to mimic the teacher as closely as possible [12]. However, in many applications, it is inconvenient for the teacher to demonstrate because the action space of the teacher is distinct from the action space of the learner. In contrast, it is usually easier for the teacher to *teach by reinforcements (TbR)*, i.e. with rewards and punishments. For example, in dog training, the trainer can't always demonstrate the task to be learned, e.g. fetch the ball with its mouth, but instead would let the dog know whether it performs well by giving treats strategically [6]; In personalizing virtual assistants, it's easier for the user to tell the assistant whether it has done a good job than to demonstrate how a task should be performed. Despite its many applications, TbR has not been studied systematically.

In this paper, we close this gap by presenting to our knowledge the first results on TbR. Specifically, we focus on a family of RL algorithms called Q-learning. Our main contributions are:

1. We formulate the optimal teaching problem in TbR.
2. We characterize the sample complexity of teaching, termed as "teaching dimension" (TDim), for Q-learning under four different teachers, distinguished by their power (or rather constraints) in constructing a teaching sequence. See Table 1 for a summary of results.
3. For each teacher level, we design an efficient teaching algorithm which matches the TDim.
4. We draw connections between our results and classic results on the sample complexity of RL and of TbD.

2 Related Work

Classic Machine Teaching Since computational teaching was first proposed in [29, 8], the teaching dimension has been studied in various learning settings. The vast majority focused on batch supervised learning. See [40] for a recent survey. Of particular interest to us though is teaching online learners such as Online Gradient Descent (OGD) [20, 18], active learners [9, 26], and sequential teaching for learners with internal learning state [11, 23, 5]. In contrast to OGD where the model update is fully determined given the teacher's data, the RL setting differs in that the teacher may not have full control over the agent's behavior (e.g. action selection) and the environment's evolution (e.g. state transition), making efficient teaching more challenging. Several recent work also study data poisoning attacks against sequential learners [39,

Preprint.

Table 1: Our Main Results on Teaching Dimension of Q-Learning

Teacher	**Level 1**	**Level 2**	**Level 3**	**Level 4**
Constraints	none	respect agent's a_t	$s_{t+1} : P(s_{t+1} \vert s_t, a_t) > 0$	$s_{t+1} \sim P(\cdot \vert s_t, a_t)$
TDim	S	$S(A-1)$	$O\left(SAH\left(\frac{1}{1-\varepsilon}\right)^D\right)$	$O\left(SAH\left(\frac{1}{(1-\varepsilon)p_{\min}}\right)^D\right)$

[22, 14, 38, 28, 21, 37]. The goal of data poisoning is to force the agent into learning some attacker-specified target policy, which is mathematically similar to teaching.

Teaching by Demonstration Several recent works studied teaching by demonstrations, particularly focusing on inverse reinforcement learning agents (IRL) [35, 15, 3, 10, 4, 36]. IRL is a sub-field of RL where the learners aim at recovering the reward function from a set of teacher demonstrations to infer a near-optimal policy. Teaching in IRL boils down to designing the most informative demonstrations to convey a target reward function to the agent. Their main difference to our work lies in the teaching paradigm. IRL belongs to TbD where the teacher can directly demonstrate the desired action in each state. The problem of exploration virtually disappears, because the optimal policy will naturally visit all important states. On the other hand, as we will see next, in the TbR paradigm, the teacher must strategically design the reward signal to *navigate* the learner to each state before it can be taught. In other words, the challenge of exploration remains in reinforcement-based teaching, making it much more challenging than demonstration-based teaching. It is worth mentioning that the NP-hardness in finding the optimal teaching strategy, similar to what we establish in this paper (see Appendix A), has also been found under the TbD paradigm [36].

Empirical Study of Teaching-by-Reinforcement Empirically, teaching in RL has been studied in various settings, such as reward shaping [25], where teacher speeds up learning by designing the reward function, and action advising [34, 1], where the teacher can suggest better actions to the learner during interaction with the environment. Little theoretical understanding is available in how much these frameworks accelerate learning. As we will see later, our teaching framework generalizes both approaches, by defining various levels of teacher's control power, and we provide order-optimal teaching strategies for each setting.

3 Problem Definitions

The machine teaching problem in RL is defined on a system with three entities: the underlying MDP environment, the RL agent (student), and the teacher. The teaching process is defined in Alg. 1. Whenever the boldface word "**may**" appears in the protocol, it depends on the level of the teacher and will be discussed later. In this paper, we assume that there is a clear separation between a training phase and a test phase, similar to the best policy identification (BPI) framework [7] in classic RL. In the training phase, the agent interacts with the MDP for a finite number of episodes and outputs a policy in the end. In the test phase, the output policy is fixed and evaluated. In our teaching framework, the teacher can decide when the training phase terminates, and so teaching is regarded as completed as soon as the target policy is learned. Specifically, in the case of Q-learning, we do not require that the estimated Q function converges to the true Q function w.r.t. the deployed policy, which is similarly not required in the BPI or PAC-RL frameworks, but only require that the deployed policy matches the target policy exactly.

Algorithm 1 Machine Teaching Protocol on Q-learning

Entities: MDP environment, learning agent with initial Q-table Q_0, teacher with target policy $\pi^\dagger$.

```
1: while $\pi_t \neq \pi^\dagger$ do
2:     MDP draws $s_0 \sim \mu_0$ after each episode reset. But the teacher may override $s_0$.
3:     for $t = 0, \dots H-1$ do
4:         The agent picks an action $a_t = \pi_t(s_t)$ with its current behavior policy $\pi_t$. But the teacher may override $a_t$ with a teacher-chosen action.
5:         The MDP evolves from $(s_t, a_t)$ to produce immediate reward $r_t$ and the next state $s_{t+1}$. But the teacher may override $r_t$ or move the system to a different next state $s_{t+1}$.
6:         The agent updates $Q_{t+1} = f(Q_t, e_t)$ from experience $e_t = (s_t, a_t, r_t, s_{t+1})$.
7: Once the agent learns $\pi^\dagger$, the teacher ends the teaching phase, and the learned policy is fixed and deployed.
```

Environment M: We assume that the environment is an episodic Markov Decision Process (MDP) parameterized by $M = (\mathcal{S}, \mathcal{A}, R, P, \mu_0, H)$ where $\mathcal{S}$ is the state space of size S, $\mathcal{A}$ is the action space of size A, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the transition probability, $\mu_0 : \mathcal{S} \to \mathbb{R}$ is the initial state distribution, and H is the episode length. Next, we define two quantities of interest of an MDP that we will use in our analysis.

Definition 1. *Let the* ***minimum transition probability*** $p_{\min}$ *of an MDP be defined as* $p_{\min} = \min_{s,s' \in \mathcal{S}, a \in \mathcal{A}, P(s'|s,a)>0} P(s'|s,a)$.

Definition 2. *Let the* ***diameter*** D *of an MDP be defined as the minimum path length to reach the hardest-to-get-to state in the underlying directed transition graph of the MDP.*

Specifically,

$$D = \max_{s \in S} \min_{T,(s_0,a_0,s_1,a_1,\dots,s_T=s)} T \tag{1}$$
$$\text{s.t.} \quad \mu_0(s_0) > 0, P(s_{t+1}|s_t,a_t) > 0, \forall t$$

RL agent L: We focus on a family of Q-learning agents $L \in \mathcal{L}$ with the following properties:

1. **Behavior policy**: The agent behaves according to the ε-greedy policy for some $\varepsilon \in [0,1]$, i.e.

$$\pi_t(s) := \begin{cases} \arg\max_a Q_t(s,a) & \text{w.p. } 1-\varepsilon \\ \text{Unif}(\mathcal{A} \backslash \arg\max_a Q_t(s,a)), & \text{w.p. } \varepsilon. \end{cases}$$

   Note this definition is slightly different but equivalent to standard ε-greedy exploration, where we merged the probability of choosing $\arg\max_a Q_t(s,a)$ in the second branch into the first. This simplifies our notation later.
2. **Learning Update**: Given experience $e_t = (s_t, a_t, r_t, s_{t+1})$ at time step t, the learning update $Q_{t+1} = f(Q_t, e_t)$ only modifies the (s_t, a_t) entry of the Q-table. Furthermore, the Q-table is "controllable": for any s_t, a_t, s_{t+1}, there exists a reward r such that the ranking of a_t within $Q_{t+1}(s_t, \cdot)$ can be made first, last or unchanged, respectively.

This family includes common Q-learning algorithms such as the standard ε-greedy Q-learning, as well as provably efficent variants like UCB-H and UCB-B [13].

Teacher: In this paper, we study four levels of teachers from the strongest to the weakest:

1. **Level 1**: The teacher can generate arbitrary transitions $(s_t, r_t, s_{t+1}) \in \mathcal{S} \times \mathbb{R} \times \mathcal{S}$, and override the agent chosen action a_t. None of these needs to obey the MDP (specifically μ_0, R, P).
2. **Level 2**: The teacher can still generate arbitrary current state s_t, reward r_t and next state s_{t+1}, but cannot override the agent's action a_t. The agent has "free will" in choosing its action.
3. **Level 3**: The teacher can still generate arbitrary reward r_t but can only generate MDP-supported initial state and next state, i.e. $\mu_0(s_0) > 0$, and $P(s_{t+1}|s_t, a_t) > 0$. However, it does not matter what the actual nonzero MDP probabilities are.
4. **Level 4**: The teacher can still generate arbitrary reward r_t but the initial state and next state must be sampled from the MDPs dynamics, i.e. $s_0 \sim \mu_0$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$.

In all levels, the teacher observes the current Q-table Q_t and knows the learning algorithm $Q_{t+1} = f(Q_t, e_t)$.

In this work, we are interested in analyzing the **teaching dimension**, a quantity of interest in the learning theory literature. We define an RL teaching problem instance by the MDP environment M, the student L with initial Q-table Q_0, and the teacher's target policy $\pi^\dagger$. We remark that the target policy $\pi^\dagger$ need not coincide with the optimal policy π^* for M. In any case, the teacher wants to control the experience sequence so that the student arrives at $\pi^\dagger$ quickly. Specifically,

Definition 3. *Given an RL teaching problem instance $(M, L, Q_0, \pi^\dagger)$, the* ***minimum expected teaching length*** *is* $\text{METaL}(M, L, Q_0, \pi^\dagger) = \min_{T,(s_t,a_t,r_t,s_{t+1})_{0:T-1}} \mathbb{E}[T]$, *s.t.* $\pi_T = \pi^\dagger$, *where the expectation is taken over the randomness in the MDP (transition dynamics) and the learner (stochastic behavior policy).*

METal depends on nuisance parameters of the RL teaching problem instance. For example, if Q_0 is an initial Q-table that already induces the target policy $\pi^\dagger$, then trivially METal=0. Following the classic definition of teaching dimension for supervised learning, we define TDim by the hardest problem instance in an appropriate family of RL teaching problems:

Definition 4. *The* ***teaching dimension*** *of an RL learner L w.r.t. a family of MDPs $\mathcal{M}$ is defined as the worst-case METal: $TDim = \max_{\pi^\dagger \in \{\pi: \mathcal{S} \to \mathcal{A}\}, Q_0 \in \mathbb{R}^{S \times A}, M \in \mathcal{M}} \text{METaL}(M, L, Q_0, \pi^\dagger)$.*

4 Teaching without MDP Constraints

We start our discussion with the strongest teachers. These teachers have the power of producing arbitrary state transition experiences that do not need to obey the transition dynamics of the underlying MDP. While the assumption on the teaching power may be unrealistic in some cases, the analysis that we present here provides theoretical insights that will facilitate our analysis of the more realistic/less powerful teaching settings in the next section.

4.1 Level 1 Teacher

The level 1 teacher is the most powerful teacher we consider. In this setting, the teacher can generate arbitrary experience e_t. The learner effectively becomes a "puppet" learner - one who passively accepts any experiences handed down by the teacher.

Theorem 1. *For a Level 1 Teacher, any learner $L \in \mathcal{L}$, and an MDP family $\mathcal{M}$ with $|\mathcal{S}| = S$ and a finite action space, the teaching dimension is $TDim = S$.*

It is useful to illustrate the theorem with the standard Q-learning algorithm, which is a member of $\mathcal{L}$. The worst case happens when $\arg\max_a Q_0(s,a) \neq \pi^\dagger(s), \forall s$. The teacher can simply choose one un-taught s at each step, and construct the experience $(s_t = s, a_t = \pi^\dagger(s), r_t, s_{t+1} = s')$ where s' is another un-taught state (Alg. 2 handles the end case). Importantly, the teacher chooses $r_t \in \left\{\frac{\max Q_t(s_t,\cdot)+\theta-(1-\alpha)Q_t(s_t,a_t)}{\alpha} - \gamma \max Q_t(s', \cdot) : \theta > 0\right\}$, knowing that the standard Q-learning update rule f is $Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha(r_t + \gamma \max_{a \in A} Q_t(s', a))$. This ensures that $Q_{t+1}(s, \pi^\dagger(s)) = \max_{a \neq \pi^\dagger(s)} Q_0(s,a) + \theta > \max_{a \neq \pi^\dagger(s)} Q_0(s,a)$, and thus the target policy is realized at state s. Subsequent teaching steps will not change the action ranking at state s. The same teaching principle applies to other learners in $\mathcal{L}$.

4.2 Level 2 Teacher

At level 2 the teacher can still generate arbitrary reward r_t and next state s_{t+1}, but now it cannot override the action

a_t chosen by the learner. This immediately implies that the teacher can no longer teach the desired action $\pi^\dagger(s)$ in a single visit to s: for example, Q_0 may be such that $Q_0(s, \pi^\dagger(s))$ is ranked last among all actions. If the learner is always greedy with $\varepsilon = 0$ in (1), the teacher will need to visit s for $(A-1)$ times, each time generating a punishing r_t to convince the learner that the top non-target action is worse than $\pi^\dagger(s)$. However, for a learner who randomly explores with $\varepsilon > 0$ it may perform $\pi^\dagger(s)$ just by chance, and the teacher can immediately generate an overwhelmingly large reward to promote this target action to complete teaching at s; it is also possible that the learner performs a non-target action that has already been demoted and thus wasting the step. Despite the randomness, interestingly our next lemma shows that for any ε it still takes in expectation $A-1$ visits to a state s to teach a desired action in the worst case.

Lemma 2. *For a Level 2 Teacher, any learner in $\mathcal{L}$, and an MDP family $\mathcal{M}$ with action space size A, it takes at most $A-1$ visits in expectation to a state s to teach the desired action $\pi^\dagger(s)$ on s.*

Proof Sketch: Let us consider teaching the target action $\pi^\dagger(s)$ for a particular state s. Consider a general case where there are $A-c$ actions above $\pi^\dagger(s)$ in the current ordering $Q_t(s,\cdot)$. In the worst case $c = 1$. We define the function $T(x)$ as the expected number of visits to s to teach the target action $\pi^\dagger(s)$ to the learner when there are x higher-ranked actions. For any learner in $\mathcal{L}$, the teacher can always provide a suitable reward to either move the action selected by the learner to the top of the ordering or the bottom. Using dynamic programming we can recursively express $T(A-c)$ as

$$\begin{aligned} T(A-c) &= 1 + (c-1)\frac{\varepsilon}{A-1}T(A-c) + \\ &\quad (1-\varepsilon+(A-c-1)\frac{\varepsilon}{A-1})T(A-c-1). \end{aligned}$$

Solving it gives $T(A-c) = \frac{A-c}{(1-(c-1)\frac{\varepsilon}{A-1})}$, which implies $\max_c T(A-c) = T(A-1) = A-1$.■

Lemma 2 suggests that the agent now needs to visit each state at most $(A-1)$ times to learn the target action, and thus teaching the target action on all states needs at most $S(A-1)$ steps:

Theorem 3. *For a Level 2 Teacher, any learner in $\mathcal{L}$, and an MDP family $\mathcal{M}$ with state space size S and action space size A, the teaching dimension is $TDim = S(A-1)$.*

We present a concrete level-2 teaching algorithm in Alg. 3 in the appendix. For both Level 1 and Level 2 teachers, we can calculate the exact teaching dimension due to a lack of constraints from the MDP. The next levels are more challenging, and we will be content with big O notation.

5 Teaching subject to MDP Constraints

In this section, we study the TDim of RL under the more realistic setting where the teacher must obey some notion of MDP transitions. In practice, such constraints may be unavoidable. For example, if the transition dynamics represent physical rules in the real world, the teacher may be physically unable to generate arbitrary s_{t+1} given s_t, a_t (e.g. cannot teleport).

5.1 Level 3 Teacher

In Level 3, the teacher can only generate a state transition to s_{t+1} which is in the support of the appropriate MDP transition probability, i.e. $s_{t+1} \in \{s : P(s \mid s_t, a_t) > 0\}$. However, the teacher can freely choose s_{t+1} within this set regardless of how small $P(s_{t+1} \mid s_t, a_t)$ is, as long as it is nonzero. Different from the previous result for Level 1 and Level 2 teacher, in this case, we are no longer able to compute the exact TDim of RL. Instead, we provide matching lower and upper-bounds on TDim.

Theorem 4. *For Level 3 Teacher, any learner in $\mathcal{L}$ with ε probability of choosing non-greedy actions at random, an MDP family $\mathcal{M}$ with episode length H and diameter $D \le H$, the teaching dimension is lower-bounded by*

$$TDim \ge \Omega\left((S-D)AH\left(\frac{1}{1-\varepsilon}\right)^D\right). \qquad (2)$$

proof. The proof uses a particularly hard RL teaching problem instance called the "peacock MDP" in Figure 1 to produce a tight lower bound. The MDP has S states where the first D states form a linear chain (the "neck"), the next $S-D-1$ states form a star (the "tail"), and the last state $s^{(\perp)}$ is a special absorbing state. The absorbing state can only be escaped when the agent resets after episode length H. The agent starts at $s^{(0)}$ after reset. It is easy to verify that the peacock MDP has a diameter D. Each state has A actions. For states along the neck, the a_1 action (in black) has probability $p > 0$ of moving right, and probability $1-p$ to go to the absorbing state $s^{(\perp)}$; all other actions (in red) have probability 1 of going to $s^{(\perp)}$. The a_1 action of $s^{(D-1)}$ has probability p to transit to each of the tail states. In the tail states, however, all actions lead to the absorbing state with probability 1. We consider a target policy $\pi^\dagger$ where $\pi^\dagger(s)$ is a red action a_2 for all the tail states s. It does not matter what $\pi^\dagger$ specifies on other states. We define Q_0 such that a_2 is $\arg\min_a Q_0(s,a)$ for all the tail states.

The proof idea has three steps: (1) By Lemma 2 the agent must visit each tail node s for $A-1$ times to teach the target action a_2, which was initially at the bottom of $Q_0(s,\cdot)$. (2) But the only way that the agent can visit a tail state s is to traverse the neck every time. (3) The neck is difficult to traverse as any ε-exploration sends the agent to $s^{(\perp)}$ where it has to wait for the episode to end.

We show that the expected number of steps to traverse the neck once is $H(\frac{1}{1-\varepsilon})^D$ even in the best case, where the agent's behavior policy (1) prefers a_1 at all neck states. In this best case, the agent will choose a_1 with probability $1-\varepsilon$ at each neck state s. If a_1 is indeed chosen by the agent, by construction the support of MDP transition $P(\cdot \mid s, a_1)$ contains the state to the right of s or the desired tail state (via the transition with probability $p > 0$). This enables the level 3 teacher to generate such a transition regardless of how small p is (which is irrelevant to a level 3 teacher). In other words, in the best case, the agent can move to the right once with probability $1-\varepsilon$. A successful traversal requires moving right D times consecutively, which has probability $(1-\varepsilon)^D$. The expected number of trials (to traverse) until

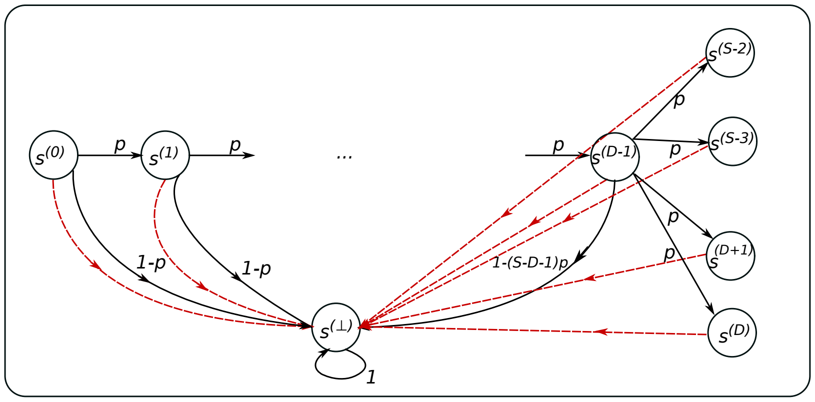


Figure 1: The "peacock" MDP

success is $(\frac{1}{1-\varepsilon})^D$. A trial fails if any time during a traversal the agent picked an exploration action a other than a_1. Then the support of $P(\cdot \mid s, a)$ only contains the absorbing state $s^{(\perp)}$, so the teacher has no choice but to send the agent to $s^{(\perp)}$. There the agent must wait for the episode to complete until resetting back to $s^{(0)}$. Therefore, any failed trial incurs exactly H steps of wasted teaching. Putting things together, the expected number of teaching steps until a successful neck traversal is done is at least $H(\frac{1}{1-\varepsilon})^D$.

There are $S-D-1$ tail states. Each needs an expected $A-1$ neck traversals to teach. This leads to the lower bound $(S-D-1)(A-1)H(\frac{1}{1-\varepsilon})^D = \Omega\left((S-D)AH\left(\frac{1}{1-\varepsilon}\right)^D\right)$. ■

Our next result shows that this lower bound is nearly tight, by constructing a level-3 teaching algorithm that can teach any MDP with almost the same sample complexity as above.

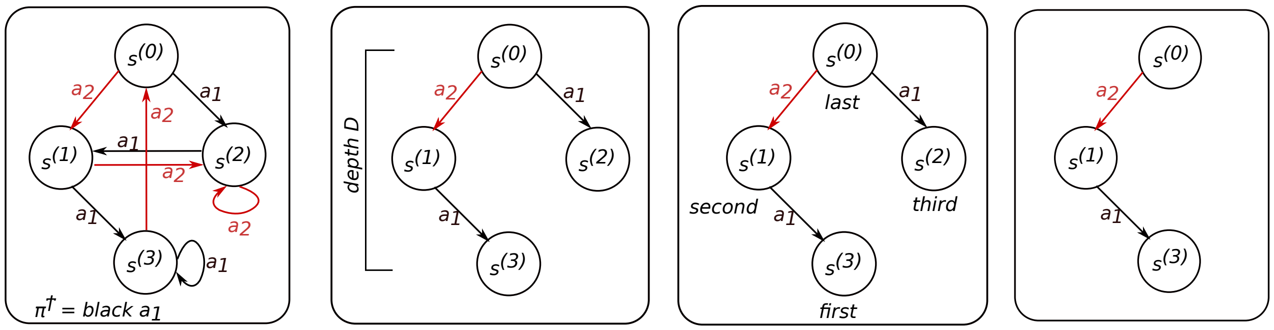
Theorem 5. *Under the same conditions of Theorem 4, the level-3 teaching dimension is upper-bounded by*

$$TDim \leq O\left(SAH\left(\frac{1}{1-\varepsilon}\right)^D\right). \tag{3}$$

proof. We analyze a level-3 teaching algorithm NavTeach (Navigation-then-Teach) which, like any teaching algorithm, provides an upper bound on TDim. The complete NavTeach algorithm is given in Alg 4 in the appendix; we walk through the main steps on an example MDP in Figure 2(a). For the clarity of illustration the example MDP has only two actions a_1, a_2 and deterministic transitions (black and red for the two actions respectively), though NavTeach can handle fully general MDPs. The initial state is $s^{(0)}$.

Let us say NavTeach needs to teach the "always take action a_1" target policy: $\forall s, \pi^\dagger(s) = a_1$. In our example, these black transition edges happen to form a tour over all states, but the path length is 3 while one can verify the diameter of the MDP is only $D = 2$. In general, though, a target policy $\pi^\dagger$ will not be a tour. It can be impossible or inefficient for the teacher to directly teach $\pi^\dagger$. Instead, NavTeach splits the teaching of $\pi^\dagger$ into subtasks for one "target state" s at a time over the state space in a carefully chosen order. Importantly, before teaching each $\pi^\dagger(s)$ NavTeach will teach a different navigation policy π^{nav} for that s. The navigation policy π^{nav} is a partial policy that creates a directed path from $s^{(0)}$ to s, which is similar to the neck in the earlier peacock example. The goal of π^{nav} is to quickly bring the agent to s often enough so that the *target* policy $\pi^\dagger(s) = a_1$ can be taught at s. That completes the subtask at s. Critically, NavTeach can maintain this target policy at s forever, while moving on to teach the next target state s'. This is nontrivial because NavTeach may need to establish a different navigation policy for s': the old navigation policy may be partially reused, or demolished. Furthermore, all these need to be done in a small number of steps. We now go through NavTeach on Figure 2(a). The first thing NavTeach does is to carefully plan the subtasks. The key is to make sure that (i) each navigation path is at most D long; (ii) once a target state s has been taught: $\pi^\dagger(s) = a_1$, it does not interfere with later navigation. To do so, NavTeach first constructs a directed graph where the vertices are the MDP states, and the edges are non-zero probability transitions of all actions. This is the directed graph of Figure 2(a), disregarding color. NavTeach then constructs a breadth-first-tree over the graph, rooted at $s^{(0)}$. This is shown in Figure 2(b). Breadth-first search ensures that all states are at most depth D away from the root. Note that this tree may uses edges that correspond to non-target actions, for example the red a_2 edge from $s^{(0)}$ to $s^{(1)}$. The ancestral paths from the root in the tree will form the navigation policy π^{nav} for each corresponding node s. Next, NavTeach orders the states to form subtasks. This is done with a depth-first traversal on the tree: a depth-first search is performed, and the nodes are ranked by the last time they are visited. This produces the order in Figure 2(c). The order ensures that later navigation is "above" any nodes on which we already taught the target policy, thus avoiding interference.

Now NavTeach starts the first subtask of teaching $\pi^\dagger(s^{(3)}) = a_1$, i.e. the black self-loop at $s^{(3)}$. As mentioned before, NavTech begins by teaching the navigation policy π^{nav} for this subtask, which is the ancestral path of $s^{(3)}$ shown in Figure 2(d). How many teaching steps does it take to establish this π^{nav}? Let us look at the nodes along the an-



(a) MDP (b) Breadth-First Tree (c) Depth-First Traversal (d) Navigation Policy

Figure 2: NavTeach algorithm demo

cestral path. By Lemma 2 the agent needs to be at the root $s^{(0)}$ $A-1$ times in expectation in order for the teacher to teach $\pi^{nav}(s^{(0)}) = a_2$; this is under the worst case scenario where the initial agent state Q_0 places a_2 at the bottom in state $s^{(0)}$. We will assume that after a visit to $s^{(0)}$, the remaining episode is simply wasted.[1] Therefore it takes at most $H(A-1)$ teaching steps to establish $\pi^{nav}(s^{(0)}) = a_2$. After that, it takes at most $H(A-1)(\frac{1}{1-\varepsilon})$ expected number of teaching steps to teach $\pi^{nav}(s^{(1)}) = a_1$. This is the same argument we used in Theorem 4: the teacher needs to make the agent traverse the partially-constructed ancestral path ("neck") to arrive at $s^{(1)}$. The worst case is if the agent performs a random exploration action anywhere along the neck; it falls off the neck and wastes the full episode. In general to establish a nagivation policy π^{nav} with path length d, NavTeach needs to teach each navigation edge at depth $i = 1 \dots d$ with at most $H(A-1)(\frac{1}{1-\varepsilon})^{i-1}$ teaching steps, respectively. After establishing this π^{nav} for $s^{(3)}$, NavTeach needs to go down the neck frequently to ensure that it visits $s^{(3)}$ $(A-1)$ times and actually teach the target policy $\pi^{\dagger}(s^{(3)}) = a_1$. This takes an additional at most $H(A-1)(\frac{1}{1-\varepsilon})^{d}$ teaching steps.

When the $s^{(3)}$ subtask is done, according to our ordering in Figure 2(c) NavTeach will tackle the subtask of teaching $\pi^{\dagger}$ at $s^{(1)}$. Our example is lucky because this new subtask is already done as part of the previous navigation policy. The third subtask is for $s^{(2)}$, where NavTeach will have to establish a new navigation policy, namely $\pi^{nav}(s^{(0)}) = a_1$. And so on. How many total teaching steps are needed? **A key insight is NavTeach only needs to teach any navigation edge in the breadth-first tree exactly once.** This is a direct consequence of the depth-first ordering: there can be a lot of sharing among navigation policies; a new navigation policy can often re-use most of the ancestral path from the previous navigation policy. Because there are exactly $S-1$ edges in the breadth-first tree of S nodes, the total teaching steps spent on building navigation policies is the sum of $S-1$ terms of the form $H(A-1)(\frac{1}{1-\varepsilon})^{i-1}$ where i is the depth of those navigation edges. We can upperbound the sum simply as $(S-1)H(A-1)(\frac{1}{1-\varepsilon})^{D}$. On the other hand, the total teaching steps spent on building the target policy $\pi^{\dagger}$ at all target states is the sum of S terms of the form $H(A-1)(\frac{1}{1-\varepsilon})^{d}$ where d is the depth of the target state. We can upperbound the sum similarly as $SH(A-1)(\frac{1}{1-\varepsilon})^{D}$. Putting navigation teaching and target policy teaching together, we need at most $(2S-1)H(A-1)(\frac{1}{1-\varepsilon})^{D} = O\left(SAH\left(\frac{1}{1-\varepsilon}\right)^{D}\right)$ teaching steps. ■

[1]It is important to note that the teacher always has a choice of r_t so that the teaching experience does not change the agent's Q_t state. For example, if the agent's learning algorithm f is a standard Q-update, then there is an r_t that keeps the Q-table unchanged. So while in wasted steps the agent may be traversing the MDP randomly, the teacher can make these steps "no-op" to ensure that they do not damage any already taught subtasks or the current navigation policy.

We remark that more careful analysis can in fact provide matching lower and upper bounds up to a constant factor, in the form of $\Theta\left((S-D)AH(1-\varepsilon)^{-D} + H\frac{1-\varepsilon}{\varepsilon}[(1-\varepsilon)^{-D}-1]\right)$. We omit this analysis for the sake of a cleaner presentation. However, the matching bounds imply that a deterministic learner, with $\varepsilon = 0$ in the ε-greedy behavior policy, has the smallest teaching dimension. This observation aligns with the common knowledge in the standard RL setting that algorithms exploring with stochastic behavior policies are provably sample-inefficient [19].

Corollary 6. *For Level 3 Teacher, any learner in $\mathcal{L}$ with $\varepsilon = 0$, and any MDP M within the MDP family $\mathcal{M}$ with $|\mathcal{S}| = S$, $|\mathcal{A}| = A$, episode length H and diameter $D \le H$, we have $TDim = \Theta(SAH)$.*

5.2 Level 4 Teacher

In Level 4, the teacher no longer has control over state transitions. The next state will be sampled according to the transition dynamics of the underlying MDP, i.e. $s_{t+1} \sim P(\cdot|s_t, a_t)$. As a result, the only control power left for the teacher is the control of reward, coinciding with the reward shaping framework. Therefore, our results below can be viewed as a sample complexity analysis of RL under *optimal reward shaping*. Similar to Level 3, we provide near-matching lower and upper-bounds on TDim.

Theorem 7. *For Level 4 Teacher, and any learner in $\mathcal{L}$, and an MDP family $\mathcal{M}$ with $|\mathcal{S}| = S$, $|\mathcal{A}| = A \geq 2$, episode length H, diameter $D \leq H$ and minimum transition probability $p_{\min}$, the teaching dimension is lower-bounded by $TDim \geq \Omega\left((S-D)AH\left(\frac{1}{p_{\min}(1-\varepsilon)}\right)^D\right)$.*

Theorem 8. *For Level 4 Teacher, any learner in $\mathcal{L}$, and any MDP M within the MDP family $\mathcal{M}$ with $|\mathcal{S}| = S$, $|\mathcal{A}| = A$, episode length H, diameter $D \leq H$ and minimum transition probability $p_{\min}$, Alg. 4 can teach any target policy $\pi^\dagger$ in a expected number of steps at most $TDim \leq O\left(SAH\left(\frac{1}{p_{\min}(1-\varepsilon)}\right)^D\right)$.*

The proofs for Theorem 7 and 8 are similar to those for Theorem 4 and 5, with the only difference that under a level 4 teacher the expected time to traverse a length D path is at most $H(1/p_{\min}(1-\varepsilon))^D$ in the worst case. The $p_{\min}$ factor accounts for sampling from $P(\cdot \mid s_t, a_t)$. Similar to Level 3 teaching, we observe that a deterministic learner incurs the smallest TDim, but due to the stochastic transition, an exponential dependency on D is unavoidable in the worst case.

Corollary 9. *For Level 4 Teacher, any learner in $\mathcal{A}$ with $\varepsilon = 0$, and any MDP M within the MDP family $\mathcal{M}$ with $|\mathcal{S}| = S$, $|\mathcal{A}| = A$, episode length H, diameter $D \leq H$ and minimum transition probability $p_{\min}$, we have $TDim \leq O\left(SAH\left(\frac{1}{p_{\min}}\right)^D\right)$.*

6 Sample efficiencies of standard RL, TbD and TbR

In the standard RL setting, some learners in the learner family $\mathcal{L}$, such as UCB-B, are provably efficient and can learn a δ-optimal policy in $O(H^3SA/\delta^2)$ iterations [13], where δ-optimal means that the cumulative rewards achieved by the output policy is only δ-worse than the optimal policy, i.e. $V^*(\mu_0) - V^\pi(\mu_0) \leq \delta$. One direct implication of such a measure is that the remote states that are unreachable also hardly affect the policy's performance, so quantities like the diameter of the MDP does not appear in the bound.

In contrast, in our TbR work, we aim at learning the *exact* optimal policy, and will thus suffer exponentially if some states are nearly unreachable. However, if we assume that all states have reasonable visiting probabilities, then even the weakest teacher (Level 3 and 4) can teach the optimal policy in $O(HSA)$ iterations, which is of H^2 factor better than the best achievable rate without a teacher. More interestingly, even the learners with a not as good learning algorithm, e.g. standard greedy Q-learning, which can never learn the optimal policy on their own, can now learn just as efficiently under the guidance of an optimal teacher.

Teaching-by-demonstration is the most sample efficient paradigm among the three, because the teacher can directly demonstrate the optimal behavior $\pi^\dagger(s)$ on any state s, and effectively eliminate the need for exploration and navigation. If the teacher can generate arbitrary (s, a) pairs, then he can teach any target policy with only S iterations, similar to our Level 1 teacher. If he is also constrained to obey the MDP, then it has been shown that he can teach a δ-optimal policy in $O(SH^2/\delta)$ iterations [32, 27], which completely drops the dependency on the action space size A compared to both RL and TbR paradigms. Intuitively, this is due to the teacher being able to directly demonstrate the optimal action, whereas, in both RL and TbR paradigms, the learner must try all actions before knowing which one is better.

In summary, in terms of sample complexity, we have

$$RL > TbR > TbD. \tag{4}$$

7 Conclusion and Discussions

We studied the problem of teaching Q-learning agents under various levels of teaching power in the Teaching-by-Reinforcement paradigm. At each level, we provided near-matching upper and lower bounds on the teaching dimension and designed efficient teaching algorithms whose sample complexity matches the teaching dimension in the worst case. Our analysis provided some insights and possible directions for future work:

1. **Agents are hard to teach if they randomly explore:** Even under an optimal teacher, learners with stochastic behavior policies ($\varepsilon > 0$) necessarily suffer from exponential sample complexity, coinciding with the observation made in the standard RL setting [19].
2. **Finding METaL is NP-hard:** While we can quantify the worst-case TDim, for a particular RL teaching problem instance we show that computing its METaL is NP-hard in Appendix A.
3. **The controllability issue:** What if the teacher cannot fully control action ranking in agent's Q_t via reward r (see agent "Learning Update" in section 3)? This may be the case when e.g. the teacher can only give rewards in $[0, 1]$. The TDim is much more involved because the teacher cannot always change the learner's policy in one step. Such analysis is left for future work.
4. **Teaching RL agents that are not Q-learners**: In the appendix, we show that our results also generalize to other forms of Temporal Difference (TD) learners, such as SARSA. Nevertheless, it remains an open question of whether even broader forms of RL agents (e.g. policy gradient and actor-critic methods) enjoy similar teaching dimension results.

References

[1] Amir, O.; Kamar, E.; Kolobov, A.; and Grosz, B. J. 2016. Interactive Teaching Strategies for Agent Training. In *IJCAI*, 804–811.

[2] Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems* 57(5): 469–483.

[3] Brown, D. S.; and Niekum, S. 2019. Machine teaching for inverse reinforcement learning: Algorithms and applications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7749–7758.

[4] Cakmak, M.; and Lopes, M. 2012. Algorithmic and human teaching of sequential decision tasks. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[5] Chen, Y.; Singla, A.; Mac Aodha, O.; Perona, P.; and Yue, Y. 2018. Understanding the Role of Adaptivity in Machine Teaching: The Case of Version Space Learners. In *Advances in Neural Information Processing Systems*, 1483–1493.

[6] Chuang, Y.-S.; Zhang, X.; Ma, Y.; Ho, M. K.; Austerweil, J. L.; and Zhu, X. 2020. Using Machine Teaching to Investigate Human Assumptions when Teaching Reinforcement Learners. *arXiv preprint arXiv:2009.02476* .

[7] Fiechter, C.-N. 1994. Efficient reinforcement learning. In *Proceedings of the seventh annual conference on Computational learning theory*, 88–97.

[8] Goldman, S. A.; and Kearns, M. J. 1992. On the complexity of teaching .

[9] Hanneke, S. 2007. Teaching dimension and the complexity of active learning. In *International Conference on Computational Learning Theory*, 66–81. Springer.

[10] Haug, L.; Tschiatschek, S.; and Singla, A. 2018. Teaching inverse reinforcement learners via features and demonstrations. In *Advances in Neural Information Processing Systems*, 8464–8473.

[11] Hunziker, A.; Chen, Y.; Mac Aodha, O.; Rodriguez, M. G.; Krause, A.; Perona, P.; Yue, Y.; and Singla, A. 2019. Teaching multiple concepts to a forgetful learner. In *Advances in Neural Information Processing Systems*.

[12] Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)* 50(2): 1–35.

[13] Jin, C.; Allen-Zhu, Z.; Bubeck, S.; and Jordan, M. I. 2018. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, 4863–4873.

[14] Jun, K.-S.; Li, L.; Ma, Y.; and Zhu, J. 2018. Adversarial attacks on stochastic bandits. In *Advances in Neural Information Processing Systems*, 3640–3649.

[15] Kamalaruban, P.; Devidze, R.; Cevher, V.; and Singla, A. 2019. Interactive Teaching Algorithms for Inverse Reinforcement Learning. In *IJCAI*, 2692–2700.

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

[17] Komorowski, M.; Celi, L. A.; Badawi, O.; Gordon, A. C.; and Faisal, A. A. 2018. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine* 24(11): 1716–1720.

[18] Lessard, L.; Zhang, X.; and Zhu, X. 2018. An optimal control approach to sequential machine teaching. *arXiv preprint arXiv:1810.06175* .

[19] Li, L. 2012. Sample complexity bounds of exploration. In *Reinforcement Learning*, 175–204. Springer.

[20] Liu, W.; Dai, B.; Humayun, A.; Tay, C.; Yu, C.; Smith, L. B.; Rehg, J. M.; and Song, L. 2017. Iterative machine teaching. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2149–2158. JMLR. org.

[21] Ma, Y.; Jun, K.-S.; Li, L.; and Zhu, X. 2018. Data poisoning attacks in contextual bandits. In *International Conference on Decision and Game Theory for Security*, 186–204. Springer.

[22] Ma, Y.; Zhang, X.; Sun, W.; and Zhu, J. 2019. Policy poisoning in batch reinforcement learning and control. In *Advances in Neural Information Processing Systems*, 14543–14553.

[23] Mansouri, F.; Chen, Y.; Vartanian, A.; Zhu, J.; and Singla, A. 2019. Preference-Based Batch and Sequential Teaching: Towards a Unified View of Models. In *Advances in Neural Information Processing Systems*, 9195–9205.

[24] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

[25] Ng, A. Y.; Harada, D.; and Russell, S. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, 278–287.

[26] Peltola, T.; Çelikok, M. M.; Daee, P.; and Kaski, S. 2019. Machine Teaching of Active Sequential Learners. In *Advances in Neural Information Processing Systems*, 11202–11213.

[27] Rajaraman, N.; Yang, L. F.; Jiao, J.; and Ramachandran, K. 2020. Toward the Fundamental Limits of Imitation Learning. *arXiv preprint arXiv:2009.05990* .

[28] Rakhsha, A.; Radanovic, G.; Devidze, R.; Zhu, X.; and Singla, A. 2020. Policy Teaching via Environment Poisoning: Training-time Adversarial Attacks against Reinforcement Learning. *arXiv preprint arXiv:2003.12909* .

[29] Shinohara, A.; and Miyano, S. 1991. Teachability in computational learning. *New Generation Computing* 8(4): 337–347.

[30] Shortreed, S. M.; Laber, E.; Lizotte, D. J.; Stroup, T. S.; Pineau, J.; and Murphy, S. A. 2011. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning* 84(1-2): 109–136.

[31] Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587): 484.

[32] Sun, W.; Venkatraman, A.; Gordon, G. J.; Boots, B.; and Bagnell, J. A. 2017. Deeply aggrevated: Differentiable imitation learning for sequential prediction. *arXiv preprint arXiv:1703.01030* .

[33] Svensson, O.; Tarnawski, J.; and Végh, L. A. 2017. A Constant-Factor Approximation Algorithm for the Asymmetric Traveling Salesman Problem. *CoRR* abs/1708.04215. URL http://arxiv.org/abs/1708.04215.

[34] Torrey, L.; and Taylor, M. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 1053–1060.

[35] Tschiatschek, S.; Ghosh, A.; Haug, L.; Devidze, R.; and Singla, A. 2019. Learner-aware Teaching: Inverse Reinforcement Learning with Preferences and Constraints. In *Advances in Neural Information Processing Systems*.

[36] Walsh, T. J.; and Goschin, S. 2012. Dynamic teaching in sequential decision making environments. *arXiv preprint arXiv:1210.4918* .

[37] Wang, Y.; and Chaudhuri, K. 2018. Data poisoning attacks against online learning. *arXiv preprint arXiv:1808.08994* .

[38] Zhang, X.; Ma, Y.; Singla, A.; and Zhu, X. 2020. Adaptive Reward-Poisoning Attacks against Reinforcement Learning. *arXiv preprint arXiv:2003.12613* .

[39] Zhang, X.; Zhu, X.; and Lessard, L. 2019. Online data poisoning attack. *arXiv preprint arXiv:1903.01666* .

[40] Zhu, X.; Singla, A.; Zilles, S.; and Rafferty, A. N. 2018. An Overview of Machine Teaching. *CoRR* abs/1801.05927.

Appendices

A The Computational Complexity of Finding METaL

In this section, we discuss another aspect of teaching, namely the computational complexity of finding the exact minimum expected teaching length of a particular teaching problem instance, i.e. $METal(M, L, Q_0, \pi^\dagger)$. Note this differs from TDim in that it is instance-specific.

For Level 1 and Level 2 teachers, the exact METaL can be found with polynomial-time algorithms Alg. 2 and Alg. 3. Now, we show that for the less powerful Level 3 teacher, finding METaL of a particular instance is NP-hard. In particular, it is as hard as the Asymmetric TSP problem.

Definition 5. *An Asymmetric TSP problem [33], characterized by a directed graph $G = (V, E)$ and a starting vertex $v \in V$, is defined as finding the minimum length path that starts from v and visits all vertices $v' \in V$ at least once.*

Theorem 10. *Finding the METaL of a Level 3 teaching problem instance is at least as hard as the Asymmetric Traveling Salesman Problem(ATSP), which is NP-hard; This also means that the best polynomial-time approximation algorithm can only achieve a constant-factor approximation.*

Proof. We show a polynomial-time reduction from ATSP problem to a Level 3 METaL problem. Specifically, we show that for every ATSP problem instance $G = (V, E)$, there exists a Level 3 METaL problem instance $(M, L, Q_0, \pi^\dagger)$ such that the ATSP problem instance has a solution l if and only if the corresponding METaL instance has a solution l.

The reduction is as follows. Given an ATSP problem instance $\{\text{Graph } G = (V, E),\ \text{start vertex} = s_0\}$, we provide a construction to a level 3 METal problem instance $(M, L, Q_0, \pi^\dagger)$. We start by constructing the MDP first. The vertex set V forms the state space of the MDP. Each state s has exactly two actions $a^{(0)}$ and $a^{(1)}$. The support of the transition probability distributions $P(\cdot \mid s, a^{(0)})$ and $P(\cdot \mid s, a^{(1)})$ are the same: they are the outgoing edges of s in the graph G. The exact value of these probabilities and the reward function does not matter, since a level 3 teacher has the power to override them. The initial state distribution μ_0 is concentrated on s_0. We construct a Q_0 that favors action $a^{(0)}$ in each state, and the target policy $\pi^\dagger(s) = a^{(1)}$ for each state $s \in \mathcal{S}$. The horizon is $H = D^2$, where D is the diameter of the graph G. The learner is in $\mathcal{L}$.

Claim 1: If an ATSP problem instance $\{G = (V, E), s_0\}$ has a solution l, then the level 3 METaL problem instance $(M, L, Q_0, \pi^\dagger)$ has a solution l.

To verify Claim 1, note the teacher needs to make the learner visit every state exactly once to teach the target action $a^{(1)}$ in that state. This is because initially every state is untaught (by construction Q_0 prefers $a^{(0)}$). Further, each state s has exactly two actions and no matter which action the learner takes, the teacher can provide a suitable reward to push the target action $a^{(1)}$ to the top of Q-value ordering. If the ATSP problem has a solution $s_{i_0} = s_0 \rightarrow s_{i_1} \rightarrow \cdots s_{i_{l-1}}$, it is possible for the teacher to provide the state transitions $s_{i_0} = s_0 \rightarrow s_{i_1} \rightarrow \cdots s_{i_{l-1}}$ that visits all the states in the least number of time steps and thus teach the target policy optimally. This is because for every edge $s_i \rightarrow s_j$ in the graph, the transition $P(\cdot \mid s_i, a)$ supports s_j for both the actions.

Claim 2: If the level 3 METaL problem instance $(M, L, Q_0, \pi^\dagger)$ has a solution l, then the ATSP problem instance $\{G = (V, E), s_0\}$ has a solution l.

We prove this by contradiction. Let say the METal problem instance $(M, L, Q_0, \pi^\dagger)$ has a solution l. Clearly, all states must have been visited in this optimal teaching length l at least once. So, the corresponding ATSP problem instance must have a solution $\leq l$. But if ATSP has a solution $m < l$, by Claim 1, the METaL problem instance will have a solution $m < l$, thus a contradiction. Hence, the ATSP problem has a solution l.

By establishing this reduction, we prove that the METaL problem for a level 3 teacher is at least as hard as ATSP problem which is itself NP-hard. ■

B Level 1: Algorithm and Proof

Proof of Theorem 1. For a level 1 teacher, the worst-case teaching problem instance is the one in which for all states $s \in \mathcal{S}$, the target action $\pi^\dagger(s)$ is not the top action in the $Q_0(s, \cdot)$. In that case, the teacher would need to make the learner visit each state s at least once so that the learner has a chance to learn $\pi^\dagger$ as s, i.e. to produce and maintain the eventual condition $Q_T(s, \pi^\dagger(s)) > \max_{a \neq \pi^\dagger(s)} Q_T(s, \cdot)$. Thus, $TDim \geq S$. On the other hand, a level-1 teacher can teach a state in just a single visit to it by replacing the agent chosen action with the target action and rewarding it with a sufficiently high reward (step 8 in the algorithm). Further, at any time step, it can also make the agent transition to an untaught state to teach the target action in that state. Thus, for the worst teaching problem instance, the level-1 teacher can teach the target policy in S steps and hence $TDim = S$. ■

Algorithm 2 Optimal Level 1 Teaching Algorithm

```
def Teach($M, L, Q_0, \pi^\dagger$):
 1: A state $s$ needs to be taught if $Q_0(s, \pi^\dagger(s)) \le \max_{a \ne \pi^\dagger(s)} Q_0(s, a)$. Terminate if the MDP has no state to be taught.
    Otherwise arbitrarily order all MDP states that need to be taught as $s^{(0)}, s^{(1)}, \cdots, s^{(n)}$ where $0 \le n \le S - 1$.
 2: The teacher provides the state $s_0 \leftarrow s^{(0)}$.
 3: for $t = 0, 1, \cdots, n$ do
 4:    The agent performs an action according to its current behavior policy $a_t \leftarrow \pi_t(s_t)$.
 5:    The teacher replaces the chosen action with target action $a_t \leftarrow \pi^\dagger(s_t)$.
 6:    The teacher provides the reward $r_t$, and next state $s_{t+1}$
 7:       where $s_{t+1} \leftarrow s^{(\min(t+1,n))}$
 8:             $r_t : Q_{t+1}(s_t, a_t) > \max_{a \ne a_t} Q_{t+1}(s_t, a)$.
 9:    The agent performs an update $Q_{t+1} \leftarrow f(Q_t, e_t)$ using experience $e_t = (s_t, a_t, r_t, s_{t+1})$
```

C Level 2: Algorithm and Proof

Algorithm 3 Optimal Level 2 Teaching Algorithm

```
def Teach($M, L, Q_0, \pi^\dagger$):
 1: A state $s$ needs to be taught if $Q_0(s, \pi^\dagger(s)) \le \max_{a \ne \pi^\dagger(s)} Q_0(s, a)$. Terminate if the MDP has no state to be taught.
    Otherwise arbitrarily order all MDP states that need to be taught as $s^{(0)}, s^{(1)}, \cdots, s^{(n)}$ where $0 \le n \le S - 1$.
 2: $t \leftarrow 0, i \leftarrow 0$, the teacher provides initial state $s_0 \leftarrow s^{(0)}$
 3: while $i \le n$ do
 4:    The agent picks a randomized action $a_t \leftarrow \pi_t(s_t)$.
 5:    if $a_t = \pi^\dagger(s_t)$ then
 6:       $s_{t+1} \leftarrow s^{(\min(i+1,n))}$
 7:       $i \leftarrow i + 1$  // move on to the next state
 8:       $r_t : Q_{t+1}(s_t, a_t) > \max_{a \ne a_t} Q_{t+1}(s_t, a)$  //promote action $\pi^\dagger(s_t)$ to top
 9:    else
10:       if {a: $Q_t(s_t, a) \ge Q_t(s_t, \pi^\dagger(s_t))\} = \{a_t, \pi^\dagger(s_t)\}$ then
11:          $s_{t+1} \leftarrow s^{(\min(i+1,n))}$
12:          $i \leftarrow i + 1$  // move on to the next state
13:       else
14:          $s_{t+1} \leftarrow s^{(i)}$  // stay at this state
15:       $r_t : Q_{t+1}(s_t, a_t) < \min_{a \ne a_t} Q_{t+1}(s_t, a)$  // demote action $a_t$ to bottom
16:    The agent performs an update $Q_{t+1} \leftarrow f(Q_t, e_t)$ with experience $e_t = (s_t, a_t, r_t, s_{t+1})$
17:    $t \leftarrow t + 1$
```

Remark: Line 10 checks whether a_t is the only no-worse action than $\pi^\dagger(s_t)$: if it is, its demotion also completes teaching at s_t.

Proof of Lemma 2. We focus on teaching the target action $\pi^\dagger(s)$ at a particular state s. In general let there be $n \in \{1, \dots, A-1\}$ other actions better than $\pi^\dagger(s)$ in $Q(s, \cdot)$. For simplicity, we assume no action is tied with $\pi^\dagger(s)$, namely

$$Q(s, a_{i_1}) \ge \cdots \ge Q(s, a_{i_n}) > Q(s, a_{i_{n+1}} = \pi^\dagger(s)) > Q(s, a_{i_{n+2}}) \ge \cdots \ge Q(s, a_{i_A}). \tag{5}$$

Define the upper action set $U := \{a_{i_1} \cdots a_{i_n}\}$ and the lower action set $U := \{a_{i_{n+2}} \cdots a_{i_A}\}$. Define $T(n)$ to be the expected number of visits to s to teach the target action $\pi^\dagger(s)$ at state s, given that initially there are n other actions better than $\pi^\dagger(s)$. By "teach" we mean move the n actions from U to L. When the agent visits s it takes a randomized action according to $a_t \leftarrow \pi_t(s)$, which can be any of the A actions. We consider three cases:

Case 1: $a_t \in U$, which happens with probability $1 - \varepsilon + (n-1)\frac{\varepsilon}{A-1}$. The teacher provides a reward to demote this action to the bottom of $Q(s, \cdot)$. Therefore, U has one less action after this one teaching step, and recursively needs $T(n-1)$ expected steps in the future.

Case 2: $a_t = \pi^\dagger(s)$, which happens with probability $\frac{\varepsilon}{A-1}$. The teacher provides a reward to promote a_t to the top of $Q(s, \cdot)$ and terminates after this one teaching step (equivalently, $T(0) = 0$).

Case 3: $a_t \in L$, which happens with probability $(A-n-1)\frac{\varepsilon}{A-1}$. The teacher can do nothing to promote the target action $\pi^\dagger(s)$ because a_t is already below $\pi^\dagger(s)$. Thus, the teacher provides a reward that keeps it that way. In the future, it still needs $T(n)$ steps.

Collecting the 3 cases together we obtain

$$T(n) = 1 + \left[\left(1-\varepsilon+(n-1)\frac{\varepsilon}{A-1}\right)T(n-1) + \frac{\varepsilon}{A-1}T(0) + (A-n-1)\frac{\varepsilon}{A-1}T(n)\right]. \tag{6}$$

Rearranging,

$$\left(1-\frac{A-n-1}{A-1}\varepsilon\right)T(n) = 1 + \left(1-\frac{A-n}{A-1}\varepsilon\right)T(n-1). \tag{7}$$

This can be written as

$$\left(1-\frac{A-1-n}{A-1}\varepsilon\right)T(n) = 1 + \left(1-\frac{A-1-(n-1)}{A-1}\varepsilon\right)T(n-1). \tag{8}$$

This allows us to introduce

$$B(n) := \left(1-\frac{A-1-n}{A-1}\varepsilon\right)T(n) \tag{9}$$

with the relation

$$B(n) = 1 + B(n-1). \tag{10}$$

Since $T(0)=0$, $B(0)=0$. Therefore, $B(n)=n$ and

$$T(n) = \frac{n}{1-\frac{A-1-n}{A-1}\varepsilon}. \tag{11}$$

It is easy to show that the worst case is $n=A-1$, where $T(A-1)=A-1$ regardless of the value of ε. This happens when the target action is originally at the bottom of $Q(s,\cdot)$. ■

Proof of Theorem 3 We construct a worst-case RL teaching problem instance. We design Q_0 so that for each state $s\in\mathcal{S}$ the target action $\pi^\dagger(s)$ is at the bottom of $Q_0(s,\cdot)$. By Lemma 2 the teacher needs to make the agent visits each state $A-1$ times in expectation. Thus a total $S(A-1)$ expected number of steps will be required to teach the target policy to the learner. ■

D Level 3 and 4: Algorithm and Proofs

Algorithm 4 The NavTeach Algorithm

```
def Init($M$):
 1: $D \leftarrow \infty$.  // select the initial state with the shortest tree
 2: for $s$ in $\{s \mid \mu_0(s) > 0\}$ do
 3:     Construct a minimum depth directed tree $T(s)$ from $s$ to all states in the underlying directed graph of $M$, via breadth first
        search from $s$. Denote its depth as $D(s)$.
 4:     if $D(s) < D$ then
 5:         depth $D \leftarrow D(s)$;  root $s^S \leftarrow s$.
 6: Let $s^1, \ldots, s^{S-1}, s^S$ correspond to a post-order depth-first traversal on the tree $T(s^S)$.

def NavTeach($M, L, Q_0, \pi^\dagger, \delta$):
 1: $t \leftarrow 0, s_t \leftarrow s^S$, ask for randomized agent action $a_t \leftarrow \pi_t(s_t)$
 2: for $i = 1, \ldots, S$ do
 3:     // subtask $i$: teach target state $s^i$ with the help of navigation path $p^i$
 4:     Let $p^i \leftarrow [s_{i_0} = s^S, s_{i_1}, ..., s_{i_d} = s^i]$ be the ancestral path from root $s^S$ to $s^i$ in tree $T(s^S)$
 5:     while $\arg\max_a Q_t(s^i, a) \neq \{\pi^\dagger(s^i)\}$ do
 6:         if $s_t = s^i$ then
 7:             // $s_t = s^i$: the current subtask, establish the target policy.
 8:             Randomly pick $s_{t+1} \in \{s' : P(s' \mid s_t, a_t) > 0\}$.
 9:             if $a_t = \pi^\dagger(s_t)$ then
10:                 $r_t \leftarrow$ CarrotStick('promote', $a_t, s_t, s_{t+1}, Q_t, \delta$)
11:             else
12:                 $r_t \leftarrow$ CarrotStick('demote', $a_t, s_t, s_{t+1}, Q_t, \delta$)
13:         else if $s_t \in p^i$ then
14:             // build navigation if MDP allows
15:             if $P(p^i.\text{next}(s_t) \mid s_t, a_t) > 0$ then
16:                 $s_{t+1} \leftarrow p^i.\text{next}(s_t)$.
17:                 $r_t \leftarrow$ CarrotStick('promote', $a_t, s_t, s_{t+1}, Q_t, \delta$).
18:             else
19:                 Randomly pick $s_{t+1} \in \{s' : P(s' \mid s_t, a_t) > 0\}$.
20:                 $r_t \leftarrow$ CarrotStick('demote', $a_t, s_t, s_{t+1}, Q_t, \delta$).
21:         else
22:             // $s_t$ is off subtask $i$ or an already taught state, maintain the $Q(s_t, a_t)$
23:             Randomly pick $s_{t+1} \in \{s' : P(s' \mid s_t, a_t) > 0\}$.
24:             $r_t \leftarrow$ CarrotStick('maintain', $a_t, s_t, s_{t+1}, Q_t, \delta$).
25:         Give experience $e_t \leftarrow (s_t, a_t, r_t, s_{t+1})$ to the agent.
26:         $t \leftarrow t + 1$
27:         if $t\%H = H - 1$ then
28:             $s_t \leftarrow s^S$.  // episode reset
29:         Ask for randomized agent action $a_t \leftarrow \pi_t(s_t)$

def CarrotStick($g, a, s, s', Q, \delta$):
 1: // make $a$ unambiguously the worst action or the best action (with margin $\delta$)or keep it as it is.
 2: if $g$ = 'promote' then
 3:     Return $r \geq 0$ such that $Q_{t+1}(s, a) = \arg\max_{b \neq a} Q_{t+1}(s, b) + \delta$ after $Q_{t+1} = f(Q_t, (s, a, r, s'))$.
 4: else if $g$ = 'demote' then
 5:     Return $r \leq 0$ such that $Q_{t+1}(s, a) = \arg\min_{b \neq a} Q_{t+1}(s, b) - \delta$ after $Q_{t+1} = f(Q_t, (s, a, r, s'))$.
 6: else
 7:     Return $r$ such that $Q_{t+1}(s, a) = Q_t(s, a)$ after $Q_{t+1} = f(Q_t, (s, a, r, s'))$.
```

Proof of Tighter Lower and Upper bound for Level 3 Teacher. We hereby prove the claimed matching $\Theta\left((S-D)AH(1-\varepsilon)^{-D} + H\frac{1-\varepsilon}{\varepsilon}[(1-\varepsilon)^{-D} - 1]\right)$ lower and upper bounds for Level 3 Teacher. The key observation is that for an MDP with state space size S and diameter D, there must exist D states whose distance to the starting state is

$0, 1, \ldots, D-1$, respectively. As a result, the total time to travel to these states is at most

$$\sum_{d=0}^{D-1} H(\frac{1}{1-\varepsilon})^d = H\frac{1-\varepsilon}{\varepsilon}[(\frac{1}{1-\varepsilon})^D - 1] \tag{12}$$

In the lower bound proof of Theorem [4], we only count the number of teaching steps required to teach the tail states. Now, if we assume in addition that the neck states also need to be taught, and the target actions are similarly at the bottom of the $Q_0(s,a)$, then it requires precisely an additional $H\frac{1-\varepsilon}{\varepsilon}[(\frac{1}{1-\varepsilon})^D - 1]$ steps to teach, which in the end gives a total of

$$(S-D-1)(A-1)H(\frac{1}{1-\varepsilon})^D + H\frac{1-\varepsilon}{\varepsilon}[(\frac{1}{1-\varepsilon})^D - 1] \tag{13}$$

steps.

In the upper bound proof of Theorem [5] we upper bound the distance from s_0 to any state by D. However, based on the observation above, at most $S-D$ states can have distance D from s_0, and the rest D states must have distance $0, 1, ..., D-1$. This allows us to upperbound the total number of teaching steps by

$$(2S-1-2D)(A-1)H(\frac{1}{1-\varepsilon})^D + H\frac{1-\varepsilon}{\varepsilon}[(\frac{1}{1-\varepsilon})^D - 1] \tag{14}$$

These two bounds matches up to a constant of 2, and thus gives a matching $\Theta\left((S-D)AH(1-\varepsilon)^{-D} + H\frac{1-\varepsilon}{\varepsilon}[(1-\varepsilon)^{-D}-1]\right)$ lower and upper bound. Setting $\varepsilon = 0$ (requires taking the limit of $\varepsilon \to 0$) induces Corollary [6]. ■

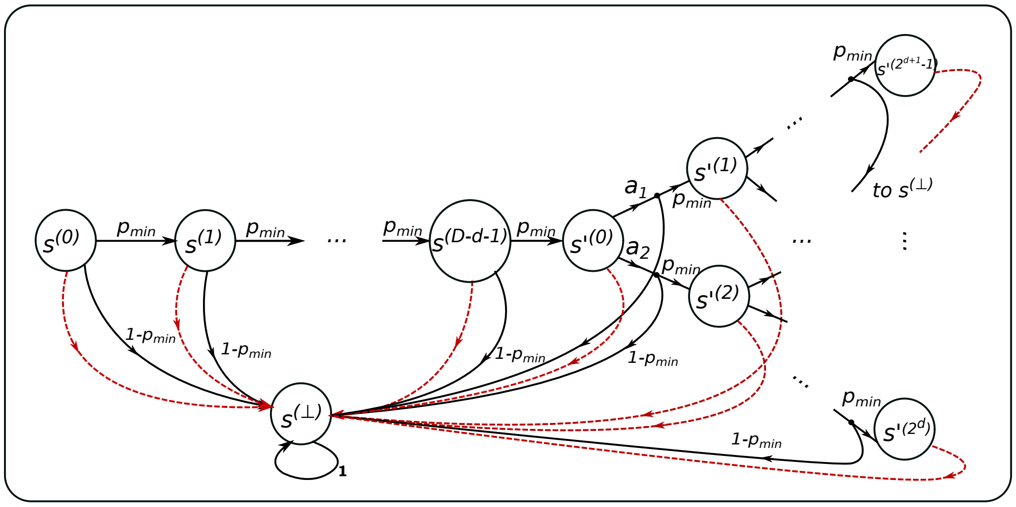


Figure 3: The "peacock tree" MDP

Proof of Theorem [7]. We construct a hard level 4 teaching problem instance, very similar to "peacock MDP" and call it "peacock tree MDP". We then show that this MDP admits the given lower bound. The "peacock tree MDP" has a linear chain of length $D-d-1$(the "neck") and a d depth binary tree(the "tail") attached to the end of the neck. For a given (S, D), we can always find d such that $2^d + (D-d+1) \le S \le 2^{d+1} + (D-d)$. Note that the depth of this MDP is D. To simplify the analysis of the proof, from now on, we will assume that the binary tree is complete and full, i.e. $S = 2^{d+1} + (D-d)$.

As in the case of "peacock MDP", every state has A actions. The action a_1 in the chain transits to next state with probability p_{min} and to the absorbing state $s^{(\perp)}$ with probability $1-p_{min}$. The action a_1 in the non-leaf states of the binary tree transits to its top child with probability p_{min} and to $s^{(\perp)}$ with probability $1-p_{min}$, the action a_2 there transits to the bottom child with probability p_{min} and to $s^{(\perp)}$ with probability $1-p_{min}$. All other $A-1$ actions in the non-leaf states and the chain states lead to $s^{(\perp)}$ with probability 1. Further, all A actions in the leaf states lead to $s^{(\perp)}$ with probability 1. The target policy is to select a_1 at every state. We consider an initial Q_0 which favors the target policy at all non-leaf and chain states. For all the leaf states s the target action a_1 is $\arg\min_a Q_0(s,a)$, namely at the bottom and needs to be taught.

For the lower bound analysis, we consider teaching each leaf state when the traversal path to it is already optimal (Note that in reality, the path has to be taught for each leaf states, but that will eventually add to the lower bound, so we omit it for this analysis). For a leaf state s, there exists a path from the root to it. This requires the teacher to provide the correct transition to the next state along the path, and the learner to choose actions a_1 all along the chain and then a combination of a_1 and a_2 actions to reach that leaf s. Given that the traversal path to the leaf is already optimal, a successful episode consists of the learner choosing the greedy action at each step and the teacher transitioning the learner to the correct next state on the path to the leaf, which happens with a probability of $(p_{min}(1-\varepsilon))^D$. Thus, the expected number of episodes required to make the learner visit the leaf and teach it once there is $(\frac{1}{p_{min}(1-\varepsilon)})^D$. Note that in a successful episode, the learner takes D steps to reach the leaf and the rest of the steps in that episode is wasted, thus accounting for a total of H steps. Similarly, any failed episode wastes a total of H steps. Hence, the expected number of steps required to visit and teach a leaf state once is at least $H(\frac{1}{p_{min}(1-\varepsilon)})^D$. The teacher has to make the learner visit all 2^d leaf states $A-1$ times in expectation (since by our construction, the target action of each leaf is at the bottom of the Q-value ordering). Collectively, this would require at least $2^d(A-1)H(\frac{1}{(p_{min}(1-\varepsilon)})^D$ steps. We note that, $S = 2^{d+1} + (D-d) \leq 2^{d+1} + D = 2\cdot 2^d + D \implies 2^d \geq \frac{1}{2}(S-D)$. Thus, the expected number of steps to teach the target policy is $\geq \frac{1}{2}(S-D)(A-1)H(\frac{1}{p_{min}(1-\varepsilon))^D}) \implies TDim \geq \Omega((S-D)AH(\frac{1}{p_{min}(1-\varepsilon))^D}))$. ■

Proof of Theorem 8. The proof follows similarly to the upper bound proof for the teaching dimension of a level 3 teacher and uses NavTeach algorithm Alg. 4. For a given MDP, the teacher first creates a breadth-first tree and then starts teaching the states in a post-order depth-first traversal. Note that the breadth-first tree is still constructed using the transition edges that are supported by the underlying MDP. A level 4 teacher, while transitioning out from a particular state, can only choose a desired transition-edge with a probability $\geq p_{min}$. Thus, the probability that the teacher can make the learner transit from one state to another using a greedy action chosen by the learner is at least $p_{min}(1-\varepsilon)$.

The teaching goal is broken into S subtasks, one for each state. The sub-task for a state further consists of teaching a navigation path to reach that state and then teaching the target action in that state. Because of the post-order depth-first teaching strategy, a large part of the navigation path is shared between two subtasks. Also, this strategy requires a navigation action at each non-leaf state to be taught just once. We further note that in depth-first teaching strategy, a navigation action from a parent state s_i to a child state s_j is taught only after a navigation path to the parent s_i is laid. Similarly, the target action at a state s_j is taught only after a navigation path to it is laid. Thus, the expected number of steps required to reach a state at depth i and teach once there is at most $(\frac{1}{p_{min}(1-\varepsilon)})^i$. For a simpler analysis, we assume that once the agent falls off the path leading to the target state, the remaining steps in that episode are wasted. Similarly, once an agent reaches a target state and is taught by the teacher, the remaining episode steps are wasted. Thus, the expected number of steps required to visit a state at depth i and teach the navigation action there is $(A-1)H(\frac{1}{p_{min}(1-\varepsilon)})^i \leq (A-1)H(\frac{1}{p_{min}(1-\varepsilon)})^D$. Noting the fact that there are at most $S-1$ non-leaf states and the teacher needs to teach the navigation action at each of them exactly once, the expected number of steps required to teach all the navigation actions is at most

$$(S-1)(A-1)H\Big(\frac{1}{p_{min}(1-\varepsilon)}\Big)^D. \tag{15}$$

Similarly, the expected number of steps required to visit a state at depth i and teach the target action there is $(A-1)H(\frac{1}{p_{min}(1-\varepsilon)})^i \leq (A-1)H(\frac{1}{p_{min}(1-\varepsilon)})^D$. Adding it up, the expected number of steps required to teach the target action at all states is at most

$$S(A-1)H\Big(\frac{1}{p_{min}(1-\varepsilon)}\Big)^D. \tag{16}$$

Combining 15 and 16, we conclude that the expected number of steps required to teach the target policy using Alg. 4 is at most

$$(2S-1)(A-1)H(\frac{1}{p_{min}(1-\varepsilon)})^D \implies TDim \leq O(SAH\Big(\frac{1}{p_{min}(1-\varepsilon)}\Big)^D). \tag{17}$$

■

Remark: A more careful analysis that leads to a tight lower and upper bound is also possible for the level 4 teacher, but the calculation and the eventual bound one gets become much more complicated, and thus we defer it to future works.

E Generalization to SARSA

SARSA is different from standard Q-learning in that its update is delayed by one step. In time step t, the agent is updating the (s_{t-1}, a_{t-1}) entry of the Q table, using experience $e_t = (s_{t-1}, a_{t-1}, r_{t-1}, s_t, a_t)$. This delayed update makes the student learn slowly. In particular, we show that it can take twice as many visits to a state to enforce the target action compared to Q-learning.

Lemma 11. *For a Level 2 Teacher, any SARSA learner, and an MDP family $\mathcal{M}$ with action space size A, it takes at most $2A-2$ visits in expectation to a state s to teach the desired action $\pi^\dagger(s)$ on s.*

Algorithm 5 Machine Teaching Protocol for SARSA

```
Entities: MDP environment, learning agent with initial Q-table $Q_0$, teacher.
1: MDP draws $s_0 \sim \mu_0$ after each episode reset. But the teacher **may** override $s_0$.
2: for $t = 0, ..., H-1$ do
3:     The agent picks an action $a_t = \pi_t(s_t)$ with its current behavior policy $\pi_t$. But the teacher **may** override $a_t$ with a
       teacher-chosen action.
4:     if $t = 0$ then
5:         The agent updates $Q_{t+1} = Q_t$.
6:     else
7:         The agent updates $Q_{t+1} = f(Q_t, e_t)$ from experience $e_t = (s_{t-1}, a_{t-1}, r_{t-1}, s_t, a_t)$.
8:     The MDP evolves from $(s_t, a_t)$ to produce immediate reward $r_t$ and the next state $s_{t+1}$. But the teacher **may** override $r_t$
       or move the system to a different state $s_{t+1}$.
```

Proof Sketch: The key in proving Lemma 11 is to see that if the agent visits the same state two times in a row, then the lesson provided by the teacher during the first visit has not been absorbed by the learner, and as a result, during the second visit, the learner will still prefer the same (undesirable) action. This, in the worst case ($\varepsilon = 0$), will be a completely wasted time step, which implies that the total number of visits required will double compared to Q-learning, giving us $2A - 2$. ■

The wasted time step in Lemma 11 will only occur when the agent visits one state twice in a roll. This can be avoided in Level 1 and 2 teachers as long as $S \geq 2$. Therefore, the teaching dimension for level 1 and 2 teachers will only increase by 1 due to the delayed update of the learner. For Level 3 and Level 4 teacher, the new Lemma 11 only results in at most 2 times increase in the teaching dimension, which does not change the order of our results. Therefore, Level 3 and Level 4 results still hold for SARSA agents.