Reinforcement Learning from Reachability Specifications: PAC Guarantees with Expected Conditional Distance

Jakub Svoboda [1] **Suguman Bansal** [2] **Krishnendu Chatterjee** [1]

Abstract

Reinforcement Learning (RL) from temporal logical specifications is a fundamental problem in sequential decision-making. One of the basic and core specification is the reachability specification that requires a target set to be eventually visited. Despite strong empirical results for RL from such specifications, the theoretical guarantees are bleak, including the impossibility of Probably Approximately Correct (PAC) guarantee for reachability specifications. Given the impossibility result, in this work we consider the problem of RL from reachability specifications along with the information of *expected conditional distance (ECD)*. We present (a) lower bound results which establish the necessity of ECD information for PAC guarantees and (b) an algorithm that establishes PAC-guarantees given the ECD information. To the best of our knowledge, this is the first RL from reachability specifications that does not make any assumptions about the underlying environment to learn policies with guarantees.

1. Introduction

Reinforcement learning (RL) (Sutton & Barto, 2018) is a promising approach to learn policies for challenging tasks such as walking (Collins et al., 2005) and grasping (Andrychowicz et al., 2020), control of multi-agent systems (Inala et al., 2021; Jothimurugan et al., 2022; Lowe et al., 2017), and control from visual inputs (Levine et al., 2016). A key benefit of RL is that it learns optimal policies for tasks in *an unknown environment*, making it ideal for practical applications with unavailable, noisy, or hard-to-model environments.

[1]Institute of Science and Technology, Austria [2]Georgia Institute of Technology, USA. Correspondence to: Jakub Svoboda <jakub.svoboda@ist.ac.at>, Suguman Bansal <suguman@gatech.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Recent years have seen an emergence of *RL from temporal logical specifications* (Aksaray et al., 2016; Brafman et al., 2018; De Giacomo et al., 2019; Hasanbeig et al., 2018; Littman et al., 2017b; Hasanbeig et al., 2019; Yuan et al., 2019; Hahn et al., 2019; Xu & Topcu, 2019; Jiang et al., 2020; Li et al., 2017; Icarte et al., 2018; Jothimurugan et al., 2021). In this approach, the *desired task* is expressed in the form of high-level temporal logical specifications rather than low-level *rewards*. Logic specifications have found a place in the plethora of RL algorithms because of (a) the relative ease to express complex non-Markovian tasks compared to rewards and (b) the impressive ability of RL from logical specification algorithms to efficiently scale learning to long-horizon tasks. Potential applications include learning policies for long-horizon tasks such as navigation and manipulation in robotics and cyber-physical systems.

Despite potential applications, their utility in safety-critical applications has been limited since existing theoretical guarantees of RL from logical specifications have been bleak. *Probably approximately correct (PAC)* learning (Valiant, 1984) is a framework for formalizing guarantees of a learning algorithm: Given an approximation parameter $\varepsilon > 0$ and a confidence parameter $\delta > 0$, a learning algorithm is said to be PAC if it returns a solution that is ε-close to optimal with probability at least $1-\delta$ within a finite (bounded) number of samples. In RL, PAC learning algorithms are known to exist for reward-based specifications (Brafman & Tennenholtz, 2002; Kakade, 2003). However, PAC guarantees from logical specifications have been proven to be impossible (Alur et al., 2022; Yang et al., 2021). The impossibility already appears for *reachability specifications* that require a target state set in the environment to be eventually visited. The PAC impossibility hinges on the inherent *non-robustness* of infinite-horizon logical specifications due to which small perturbations in the environment do not preserve optimal or near-optimal policies (Littman et al., 2017a). As a result, obtaining an optimal or near-optimal policy on a close approximation of the environment does not guarantee a near-optimal policy in the true environment.

In light of PAC impossibility for reachability, our central result is a PAC learning algorithm for reachability specifications using a novel parameter called the *Expected Condi-*

tional Distance (ECD). For a policy, the ECD computes the expected length of trajectories that satisfy the reachability specification. ECD differs from the well-known *Shortest Stochastic Path (SSP)* as the SSP additionally accounts for the expected length of trajectories that do not satisfy the reachability specification. More importantly, ECD serves as an *external* parameter, meaning that assigning a value to ECD does not impose any assumptions on the underlying environment. This sets it apart from previous work focused on achieving PAC guarantees through additional parameterization (as discussed in Section 1.1) where the parameter is *internal* to the environment, requiring additional assumption about the underlying environment. Not only does the use of internal parameters violate the fundamental assumption of an unknown environment in RL, but it also limits the practical applicability of theoretical results because guarantees only hold when the parameterization aligns with the ground truth in the environment. Finally, our algorithm is efficient, i.e. it is polynomial in the size of the input parameters.

Our contributions are as follows. First, we present a lower bound that establishes the necessity of ECD to obtain PAC guarantees on learning from reachability specifications. Second, we present an *efficient* learning algorithm that establishes PAC guarantees given the ECD, i.e., the sample complexity of our PAC learning algorithm is polynomial in the input parameters of the environment $|S|$ and $|A|$ (number of states and actions), the number of target states $|T|$ PAC parameters $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$, and the ECD ℓ. Moreover, we discuss the implications of our result for more general temporal logic specifications (Remark 2.1). To the best of our knowledge, our parametrization by ECD is the first PAC-possible result that does not make any assumptions on the environment.

Outline. Section 2 introduces necessary preliminaries. Our parameter ECD is introduced in Section 3. The necessity of ECD in learning PAC guarantees (lower bound) is established in Section 4. Finally, our sample efficient PAC learning algorithm is presented in Section 5.

1.1. Related Work

Our PAC-possibility result contrasts with existing works that obtain PAC guarantees using additional parameterization. Our result makes no assumptions about the underlying environment. A PAC learning algorithm for linear temporal logic (LTL) specification (Pnueli, 1977) was introduced in (Fu & Topcu, 2014). However, this requires knowledge of the topology of the environment, i.e., it requires to know all transitions with non-zero probability. (Ashok et al., 2019) improves upon this result by only requiring to know the value of the minimum non-zero transition probability in the environment. Most recently, (Perez et al., 2023) proves PAC with LTL specifications given the *mixing time* of the environment.

The only logical specifications for which RL is PAC-learnable are *finitary* specifications which assign a bound on the length of the trajectory. In this case, learning over infinite-horizon tasks reduces to learning on a (fixed) finite horizon (Yang et al., 2021). Alternately, quantitative semantics for logical specifications have also been explored to alleviate the non-robustness of logical specifications. A recent work proves PAC guarantees for LTL specifications under a discounted-sum semantics (Alur et al., 2023).

2. Preliminaries

2.1. Reachability in Markov Decision Process.

Markov Decision Process (MDP). A *Markov Decision Process* (MDP) is a tuple $\mathcal{M} = (S, A, s_0, P)$, where S is a finite set of states and $|S| = n$, s_0 is the initial state,[1] A is a finite set of actions, and $P : S \times A \times S \to [0, 1]$ is the transition probability function, where $\sum_{s' \in S} P(s, a, s') = 1$ for all $s \in S$ and $a \in A$. Given an MDP $\mathcal{M}$ we denote by $G(\mathcal{M}) = (S, \mathcal{E})$ the associated graph structure with set S of states and the set of edges $\mathcal{E} = \{(s, s') \mid \exists a \in A.P(s, a, s') > 0\}$ consists of transitions with positive probabilities. In other words, $G(\mathcal{M})$ represents the MDP structure but not the precise probabilities.

Runs. An *infinite run* $\zeta \in (S \times A)^\omega$ is a sequence $\zeta = s_0 a_0 s_1 a_1 \ldots$, where $s_i \in S$ and $a_i \in A$ for all $i \in \mathbb{N}$. Similarly, a *finite run* $\zeta \in (S \times A)^* \times S$ is a finite sequence $\zeta = s_0 a_0 s_1 a_1 \ldots a_{t-1} s_t$. For any run ζ of length at least j and any $i \leq j$, we let $\zeta_{i:j}$ denote the subsequence $s_i a_i s_{i+1} a_{i+1} \ldots a_{j-1} s_j$. We use $\mathsf{Runs}(\mathcal{M}) = (S \times A)^\omega$ and $\mathsf{Runs}_f(\mathcal{M}) = (S \times A)^* \times S$ to denote the set of infinite and finite runs, respectively.

Policies. Let $\mathcal{D}(A) = \{\Delta : A \to [0, 1] \mid \sum_{a \in A} \Delta(a) = 1\}$ denote the set of all distributions over actions. A policy $\pi : \mathsf{Runs}_f(S, A) \to \mathcal{D}(A)$ maps a finite run $\zeta \in \mathsf{Runs}_f(S, A)$ to a distribution $\pi(\zeta)$ over actions. We denote by $\Pi(S, A)$ the set of all such policies. A policy π is *positional* if $\pi(\zeta) = \pi(\zeta')$ for all $\zeta, \zeta' \in \mathsf{Runs}_f(S, A)$ with $\mathsf{last}(\zeta) = \mathsf{last}(\zeta')$ where $\mathsf{last}(\zeta)$ denotes the last state in the run ζ. A policy π is deterministic if, for all finite runs $\zeta \in \mathsf{Runs}_f(S, A)$, there is an action $a \in A$ with $\pi(\zeta)(a) = 1$.

Probability and expectation measures. Given a finite run $\zeta = s_0 a_0 \ldots a_{t-1} s_t$, the *cylinder* of ζ, denoted by $\mathsf{Cyl}(\zeta)$, is the set of all infinite runs starting with prefix ζ. Given an MDP $\mathcal{M}$ and a policy $\pi \in \Pi(S, A)$, we define the probability of the cylinder set by $\mathcal{D}_\pi^{\mathcal{M}}(\mathsf{Cyl}(\zeta)) = \prod_{i=0}^{t-1} \pi(\zeta_{0:i})(a_i) P(s_i, a_i, s_{i+1})$. It is known that $\mathcal{D}_\pi^{\mathcal{M}}$ can be uniquely extended to a probability measure over the σ-

[1] A distribution η over initial states can be modeled by adding a new state s_0 from which taking any action leads to a state sampled from η.

algebra generated by all cylinder sets. We use $\mathcal{D}_\pi^{\mathcal{M}}$ to denote the distribution over infinite runs in $\mathcal{M}$ induced by the policy π and the associated expectation measure is denoted by $\mathbb{E}_\pi^{\mathcal{M}}$.

Formal Specifications and Reachability. Formal languages can be used to specify properties about runs of an MDP. A language specification $\mathcal{L} \subseteq \mathsf{Runs}$ is a set of *desirable* runs in an MDP.

Given a formal specification $\mathcal{L}$, the value of a policy π w.r.t. specification $\mathcal{L}$ is the probability of generating a sequence in $\mathcal{L}$—i.e.,

$$J_{\mathcal{L}}^{\mathcal{M}}(\pi) = \mathcal{D}_\pi^{\mathcal{M}}\big(\{\zeta \in \mathsf{Runs}(S,A) \mid \zeta \in \mathcal{L}\}\big).$$

Let $\mathcal{J}^*(\mathcal{M},\mathcal{L}) = \sup_\pi J_{\mathcal{L}}^{\mathcal{M}}(\pi)$ denote the maximum value of $J_{\mathcal{L}}^{\mathcal{M}}$ for all policies $\pi \in \Pi(S,A)$. We let $\Pi_{\mathsf{opt}}(\mathcal{M},\mathcal{L})$ denote the set of all optimal policies in $\mathcal{M}$ w.r.t. $\mathcal{L}$—i.e., $\Pi_{\mathsf{opt}}(\mathcal{M},\mathcal{L}) = \{\pi \mid J_{\mathcal{L}}^{\mathcal{M}}(\pi) = \mathcal{J}^*(\mathcal{M},\mathcal{L})\}$. In many cases, it is sufficient to compute an ε-optimal policy $\tilde{\pi}$ with $J_{\mathcal{L}}^{\mathcal{M}}(\tilde{\pi}) \geq \mathcal{J}^*(\mathcal{M},\mathcal{L}) - \varepsilon$; we let $\Pi_{\mathsf{opt}}^{\varepsilon}(\mathcal{M},\mathcal{L})$ denote the set of all ε-optimal policies in $\mathcal{M}$ w.r.t. $\mathcal{L}$.

Reachability specifications comprise an important class of formal specifications. Given a set of states $T \subseteq S$, let a *reachability specification* $\mathcal{L}(T) = \{\zeta = s_0 a_0 \cdots \in \mathsf{Runs}(\mathcal{M}) \mid \exists i \in \mathbb{N}, s_i \in T\}$ be the set of all runs in $\mathcal{M}$ that visit a state in T. We refer to T by the target states/accepting states.

Remark 2.1 (Significance of reachability specifications). We discuss the significance of reachability specifications.

1. First, reachability is the most basic temporal specification, e.g., it corresponds to the class of open sets in the topological characterization of temporal specifications.
2. Second, if we consider general ω-regular specifications, which subsume LTL specifications, then parity specifications provide a canonical way to express them (Safra, 1988). For MDPs with parity specifications, the optimal value is computed as follows (Courcoubetis & Yannakakis, 1995): (a) the set of states X where the optimal value is 1 (almost-sure winning set) is computed; and (b) the optimal value is the optimal reachability probability to X. The almost-sure winning set of an MDP $\mathcal{M}$ only depends on the associated structure $G(\mathcal{M})$ and not the precise probabilities and can be computed efficiently (in sub-quadratic time) with discrete graph theoretic algorithms (Chatterjee & Henzinger, 2011; 2014). Thus if the structure of an MDP is known, for all ω-regular specifications, the core task is to solve the optimal reachability problem.
3. Third, LTLf goals (De Giacomo et al., 2013) goals are also expressed as reachability goals.

Given that reachability is the basic specification, and it is a core problem as mentioned above, in the sequel we only focus on reachability specifications.

2.2. Reinforcement Learning

We define a *reinforcement learning task* to be a pair $(\mathcal{M},\mathcal{L})$ where $\mathcal{M}$ is an MDP and $\mathcal{L}$ is a specification for $\mathcal{M}$. The goal of an RL task $(\mathcal{M},\mathcal{L})$ is to use a *learning algorithm* to produce an optimal or near-optimal policy w.r.t. $\mathcal{L}$ in a *simulator* of $\mathcal{M}$. The key components are described below.

Simulator. In reinforcement learning, the standard assumption is that the set of states S, the set of actions A, and the initial state s_0 are known but the transition probability function P is unknown. The learning algorithm has access to a simulator $\mathbb{S}$ which can be used to sample runs of the system $\zeta \sim \mathcal{D}_\pi^{\mathcal{M}}$ using any policy π. The simulator can also be the real system, such as a robot, that $\mathcal{M}$ represents. Internally, the simulator stores the current state of the MDP which is denoted by $\mathbb{S}.\texttt{state}$. It makes the following functions available to the learning algorithm.

$\mathbb{S}.\texttt{reset()}$: This function sets $\mathbb{S}.\texttt{state}$ to the initial state s_0.

$\mathbb{S}.\texttt{step}(a)$: Given as input an action a, this function samples a state $s' \in S$ according to the transition probability function P—i.e., the probability that a state s' is sampled is $P(s,a,s')$ where $s = \mathbb{S}.\texttt{state}$. It then updates $\mathbb{S}.\texttt{state}$ to the newly sampled state s' and returns s'.

Learning algorithm. A learning algorithm $\mathcal{A}$ is an iterative process that in each iteration (i) either resets the simulator or takes a step in $\mathcal{M}$, and (ii) outputs its current estimate of an optimal policy π. A learning algorithm $\mathcal{A}$ induces a random sequence of output policies $\{\pi_n\}_{n=1}^{\infty}$ where π_n is the policy output in the n^{th} iteration.

A learning algorithm is *Probably Approximately Correct* (PAC-MDP) (Kakade, 2003) if it is guaranteed to learn a near-optimal policy with high confidence within a finite number of iterations.

Definition 2.2 (PAC-MDP)**.** A learning algorithm $\mathcal{A}$ is said to be PAC-MDP for $\mathcal{L}$ if, there is a function h such that for all $p > 0$, $\varepsilon > 0$, and all RL tasks $(\mathcal{M},\mathcal{L})$ with $\mathcal{M} = (S,A,s_0,P)$, taking $N = h(|S|,|A|,|\mathcal{L}|,\frac{1}{p},\frac{1}{\varepsilon})$, with probability at least $1-p$, we have

$$\left|\left\{n \mid \pi_n \notin \Pi_{\mathsf{opt}}^{\varepsilon}(\mathcal{M},\mathcal{L})\right\}\right| \leq N.$$

Efficiency in PAC. We say a PAC-MDP algorithm is *efficient* if the *sample complexity* function h is polynomial in $|S|, |A|, \frac{1}{p}$ and $\frac{1}{\varepsilon}$.

Prior work has shown that there exists a PAC-MDP learning algorithm for a specification $\mathcal{L}$ iff $\mathcal{L}$ is *finitary* (Yang et al., 2021). A language $\mathcal{L}$ is said to be finitary if it can be represented in LTL (Pnueli, 1977) using the Next operator only. All finitary languages can be expressed as a reachability specification, but not vice versa. i.e., reachability specifications are strictly more expressive than finitary specifications. In particular, reachability is not a finite-horizon specification. In the sequel we focus on reachability specifications with the target set T, and often use T to denote $\mathcal{L}(T)$.

3. Expected Conditional Distance

This section introduces our parameter *Expected Conditional Distance (ECD)* which we show renders PAC learnability under reachability specifications.

We begin with some useful notation. Given an MDP $\mathcal{M} = (S, A, s_0, P)$ with target states $T \subseteq S$, we define the $\mathsf{Len}_T : \mathsf{Runs} \to \mathbb{N}$ such that $\mathsf{Len}_T(\zeta)$ for $\zeta = s_0 a_0 s_1 \cdots$ be k such that for all $i < k$, $s_i \notin T$ and $s_k \in T$ if ζ visits T, and 0 otherwise (if ζ does not visit T).

Then, the Expected Conditional Distance (ECD) is defined as follows:

Definition 3.1 (Expected Conditional Distance (ECD))**.** Given an MDP $\mathcal{M} = (S, A, s_0, P)$, target states $T \subseteq S$, and a policy π, the *expected conditional distance (ECD)*, denoted $\mathsf{ECD}_T^{\mathcal{M}}(\pi)$, is the expectation of Len_T on all trajectories produced from π in $\mathcal{M}$, i.e.,

$$\mathsf{ECD}_T^{\mathcal{M}}(\pi) = \mathbb{E}_{\pi}^{\mathcal{M}}[Len_T(\zeta)]$$

Similarly, we have

$$\mathsf{ECD}_T^{\mathcal{M}} = \mathsf{inf}_{\pi \in \Pi_{\mathsf{opt}}(\mathcal{M},T)} \mathsf{ECD}_T^{\mathcal{M}}(\pi)$$

i.e., ECD of the MDP is the minimal ECD achieved among all optimal policies for the reachability specifications.

Remark 3.2 (Comparison of ECD vs SSP). The ECD concept is related to the well-known *shortest stochastic path (SSP)* (Bertsekas & Tsitsiklis, 1991), however, there is a key difference. ECD treats infinite trajectories that do not reach the target with weight 0, in contrast, SSP treats them as ∞. More precisely, if SSP is finite, then ECD and SSP coincide, otherwise SSP can be infinite, but ECD is finite.

Key intuition. The intuition behind using ECD as a parameter in learning with reachability specifications is that when learning a policy with ECD less than or equal to a given value ℓ, a learning algorithm need not sample trajectories much longer than ℓ that do not visit a state in the target set. This is because, if a trajectory significantly longer than ℓ has not reached the target state, the ECD suggests a low probability of it reaching the target in the future. Such an inference cannot be made when learning reachability specifications without ECD.

Interpretation of ECD. Further, note that ECD is neither an internal parameter of the MDP nor the specification. A user attempting to learn a policy could supply a desirable ECD value as an input *w*ithout any knowledge of the underlying environment. Another interpretation of the ECD is that it is an external value that a user could provide when they are keen on learning a policy that either visits a target state within $\sim \ell$ steps or does not visit the target at all.

Optimal policies $\Pi_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ and near-optimal policies $\Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ with ECD. We now define optimal and ε-optimal policies among those that meet an ECD requirement. Let $\Pi_{\leq \ell, T}$ denote the set of policies for which ECD w.r.t. the target states T is less than or equal to ℓ, i.e., $\Pi_{\leq \ell, T} = \{\pi \in \Pi \mid \mathsf{ECD}_T^{\mathcal{M}}(\pi) \leq \ell\}$. Let

$$\mathcal{J}^*(\mathcal{M}, T, \ell) = \sup_{\pi \in \Pi_{\leq \ell, T}} J_T^{\mathcal{M}}(\pi)$$

denote the maximum value of $J_T^{\mathcal{M}}(\pi)$ such that $\pi \in \Pi_{\leq \ell, T}$. We let $\Pi_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ denote the set of all optimal policies in $\mathcal{M}$ w.r.t. T with ECD at most ℓ—i.e.,

$$\Pi_{\mathsf{opt}}(\mathcal{M}, T, \ell) = \{\pi \in \Pi_{\leq \ell, T} \mid J_T^{\mathcal{M}}(\pi) = \mathcal{J}^*(\mathcal{M}, T, \ell)\}.$$

In many cases, it is sufficient to compute an ε-optimal policy $\tilde{\pi} \in \Pi_{\leq \ell, T}$ with $J_T^{\mathcal{M}}(\tilde{\pi}) \geq \mathcal{J}^*(\mathcal{M}, T, \ell) - \varepsilon$. We let $\Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ denote the set of ε-optimal policies in $\mathcal{M}$ w.r.t. T with ECD at most ℓ—i.e.,

$$\Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell) = \{\pi \in \Pi_{\leq \ell, T} \mid J_T^{\mathcal{M}}(\pi) \geq \mathcal{J}^*(\mathcal{M}, T, \ell) - \varepsilon\}.$$

We now define the PAC-MDP problem with ECD.

Definition 3.3 (PAC-MDP with ECD)**.** A learning algorithm $\mathcal{A}$ is said to be *PAC-MDP with ECD* of length ℓ if there is a function h such that for any $p > 0$, $\varepsilon > 0$, and a reachability RL task $(\mathcal{M}, \mathcal{L}(T))$ with $\mathcal{M} = (S, A, s_0, P)$ and $T \subseteq S$, taking $N = h(|S|, |A|, |T|, \frac{1}{p}, \frac{1}{\varepsilon}, \ell)$, with probability at least $1 - p$, we have

$$\left| \left\{ n \mid \pi_n \notin \Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell) \right\} \right| \leq N.$$

Efficiency in PAC. The function h computes the *sample complexity* of the PAC-MDP with ECD algorithm. We say a PAC-MDP with ECD algorithm is *efficient* if h is polynomial in $|S|, |A|, |T|, \frac{1}{p}, \frac{1}{\varepsilon}$, and ℓ.

Remark 3.4 (Connection of Definition 2.2 and Definition 3.3). Given an MDP $\mathcal{M}$ with target set T, if the input parameter ℓ satisfies that $\ell \geq \mathsf{ECD}_T^{\mathcal{M}}$, then every policy in $\Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ is also in $\Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T)$ (i.e., this is an ε-optimal policy in $\mathcal{M}$). Hence in this case Definition 3.3 coincides with Definition 2.2.

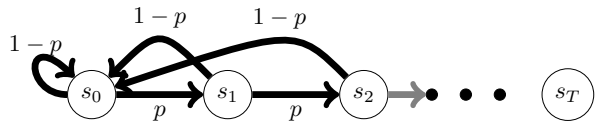

Figure 1: An MDP with reachability for $T = \{s_T\}$, where ECD is $p^{-|S|+1}$.

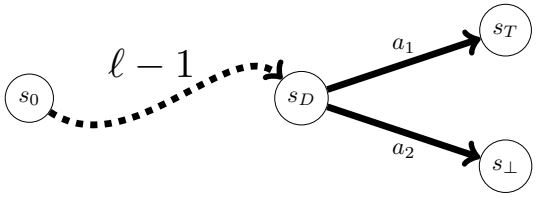

Figure 2: Structure of an MDP with one state with two actions where PAC-MDP is not efficient.

4. Lower Bound for PAC-MDP with ECD

In this section, we present a lower bound that establishes that the sample complexity to learn a reachability specification must be proportional to the ECD.

Key ideas. The key idea for the lower bound is as follows: We first show that there exists an MDP $\mathcal{M}$ and target states T such that the ECD is exponential in the smallest non-zero transition probability in $\mathcal{M}$. This construction shows that ECD can be large even for simple MDP. We use the construction to create MDP $\mathcal{M}$ where we show the impossibility of learning in fewer steps than ECD.

Lemma 4.1. *There exists $\mathcal{M}$ with target set T such that for every π we have*

$$\mathsf{ECD}_T^{\mathcal{M}}(\pi) \geq p^{-|S|+1}\,,$$

where p is the smallest transition probability.

Figure 1 shows the structure of $\mathcal{M}$, for proof, see Appendix A.

Theorem 4.2 (Lower bound for PAC-MDP with ECD)**.** *There exists an MDP $\mathcal{M} = (S, A, s_0, P)$ with reachability RL task $(\mathcal{M}, \mathcal{L}(T))$ for $T \subseteq S$, where for all $\pi \in \Pi_{\mathsf{opt}}(\mathcal{M}, T)$ we have $\mathsf{ECD}_T^{\mathcal{M}}(\pi) \geq \ell$, then for all $p < \frac{1}{2}$, $\varepsilon < \frac{1}{2}$, and all learning algorithms $\mathcal{A}$ with probability at least p we have*

$$\left|\left\{n \mid \pi_n \notin \Pi_{\mathsf{opt}}^{\varepsilon}(\mathcal{M}, T)\right\}\right| \geq \frac{1}{2}(1-2p)(\ell-2)\,.$$

This result holds even when the structure $G(\mathcal{M})$ of the MDP is known and the MDP has only one state with two actions and all other states have a single action.

Proof. We construct the MDP $\mathcal{M}$, the sketch is on Figure 2. The starting state is s_0, and every state has only a single action, except state s_D which has two actions. The structure from s_0 to s_D is not important, states create a Markov Chain that in $\ell - 1$ steps in expectation reaches s_D. Moreover, the state s_0 is the furthest away from s_D. From the decision state s_D, there are two actions a_1 and a_2 (assigned randomly), one leads to the target s_T and the other leads to a state $s_\perp$ that cannot be escaped. The objective of the learning algorithm is to determine whether a_1 or a_2 leads to s_T.

When s_D is visited, the learning algorithm plays one action and from the result, it knows which action leads to s_T. We show that with probability $2p$ in $(1-2p)(\ell-2)$ steps, the state s_D is not reached. For the sake of contradiction, suppose that with probability $1-2p$, the path to s_D takes at most $(1-2p)(\ell-2)$ steps. We compute the upper bound on the expected length E. With probability $1-2p$, the path length is at most $(1-2p)(\ell-2)$, otherwise, at worst the process starts from s_0 again. We obtain upper bound on E as follows

$$E \leq (1-2p)(1-2p)(\ell-2) + 2p(E + (1-2p)(\ell-2))\,,$$

which gives $E \leq \ell - 2$, a contradiction for the construction of the Markov Chain.

Now, we know that with probability at least $2p$, the algorithm $\mathcal{A}$ needs at least $(1-2p)(\ell-2)$ steps to reach s_D. Before s_D is reached $\mathcal{A}$ has to choose a policy blindly. Since $\varepsilon < \frac{1}{2}$, at least one action ($a_1$ or a_2) needs to be selected with probability at least $1-\varepsilon$, otherwise the policy is not optimal trivially. That means $\mathcal{A}$ needs to commit to one action (that is to choose the action with probability at least $1-\varepsilon$). However, in $(1-2p)(\ell-2)$ steps, any learning algorithm needs to commit to one of the two actions at least $\frac{1}{2}(1-2p)(\ell-2)$ times, with probability $\frac{1}{2}$ this is the wrong action.

That means with probability $2p$, the correct action is not learned within $(1-p)(\ell-2)$ steps and with probability $\frac{1}{2}$, at least half of the steps the learning algorithm outputted a wrong policy. □

Remark 4.3 (Impossibility of PAC-MDP without ECD). The above lower bound for PAC-MDP with ECD has the following significance of PAC-MDP: it shows the impossibility for efficient PAC-MDP if $\mathsf{ECD}_T^{\mathcal{M}}$ is bigger than polynomial in the input parameters of Definition 2.2, and Lemma 4.1 shows that indeed $\mathsf{ECD}_T^{\mathcal{M}}$ can be exponential. In general efficient PAC-MDP is impossible without ECD information.

5. Algorithm for PAC-MDP with ECD

This section presents our PAC-MDP with ECD algorithm for reachability specifications. The section is organized as

follows: Section 5.1 presents an algorithm to learn an approximation of an unknown MDP. Section 5.2 presents an algorithm to find a near-optimal policy w.r.t reachability goals with bounded ECD in a *known* MDP. Finally, Section 5.3 combines the above two results to yield our PAC-MDP algorithm for reachability goals with ECD.

5.1. Learning an MDP approximation

We introduce ApproximateMDP (Algorithm 1) which constructs an approximation of the MDP such that the transition probabilities of all states reachable within some number of steps from the s_0 by any policy. We say that some states with a good approximation of the transition probabilities are *explored*. The algorithm returns MDP $\mathcal{M}'$ which is the approximation of $\mathcal{M}$ on these explored states.

Algorithm overview. The inputs to algorithm ApproximateMDP are a simulator $\mathbb{S}$ of $\mathcal{M} = (S, A, s_0, P)$ with set of states S, ECD ℓ, and parameters ε and p for error and confidence, respectively. The goal is to build an estimated MDP $\mathcal{M}' = (S, A, s_0, P')$ such that with high probability (i.e. with probability at least $1 - p$), $\mathcal{M}'$ is a *good approximation* of $\mathcal{M}$, i.e., the difference between corresponding transitions between the *explored states* $E \subseteq S$ in the two MDPs is bounded w.r.t. parameters ℓ and ε. The algorithm returns the estimated model $\mathcal{M}'$ and the set of explored states E.

The algorithm estimates the MDP $\mathcal{M}'$ by calling subroutine Explore on states of the MDP in order to estimate the transition probabilities of all actions from these states. Since the original MDP $\mathcal{M}$ is unknown, ApproximateMDP creates $\mathcal{M}'$ by sampling states using the simulator $\mathbb{S}$ of $\mathcal{M}$. Initially, all states in $\mathcal{M}'$ are unexplored and all transition probability distributions are unknown. The algorithm begins by calling Explore on the initial state s_0. While exploring a state, Explore samples each action *for sufficiently many number of times* and records the estimated transition probability distribution for each state-action pair. It continues to explore states until all the unexplored states are *far* from s_0, by the notion of distance defined below.

Algorithm Details. We say that a state-action pair (s, a) is *explored* if action a has been sampled in state s (i.e. $\mathbb{S}$.`step(a)` is called from state s) $\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times. A state s is said to be *explored* if for all actions $a \in A$, the state-action pair (s, a) has been explored. A state is said to be *unexplored* otherwise.

Let MDP $\mathcal{M}'$ be an estimate of MDP $\mathcal{M} = (S, s_0, A, P)$ s.t. $E \subseteq S$ is the set of explored states in $\mathcal{M}'$. Let s_0 be the initial state in both. The *path probability* of state s from s_0 in $\mathcal{M}'$, denoted $p_{\mathcal{M}'}(s_0, s, d)$, is the highest probability with which any policy can reach s from s_0 within d steps.

Algorithm 1 ApproximateMDP$(\mathbb{S}, S, \ell, \varepsilon, p)$
Approximates the MDP $\mathcal{M}$ on $E \subseteq S$.

```
Explore(s_0)
E ← {s_0}
while ∃s ∈ S \ E : p_{M'}(s_0, s, 6ℓ/ε) ≥ ε/(7n) do
  Explore(s)
  E ← E ∪ {s}
end while
Return M', E
```

The subroutine Explore(s) operates on an unexplored state s as follows: It ensures, with high probability, that for all actions $a \in A$, the pair (s, a) is sampled $\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times and uses these to estimate the transition probability distribution on (s, a). To do so, Explore computes a policy π to reach state s from s_0 in $\mathcal{M}'$ with high probability in $6\ell/\varepsilon$ steps, then samples trajectories in $\mathcal{M}$ using π. Appendix B shows that, with high probability, sampling several such trajectories ensures that each state-action pair (s, a) can be sampled $\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times.

Theoretical Guarantees. We say that an MDP $\mathcal{M}'$ with explored states E is a *good approximation* of an MDP $\mathcal{M}$ w.r.t. error parameter $\varepsilon > 0$ and ECD $\ell > 0$ if for all states $s \in E$, actions $a \in A$, and states $s' \in S$, we have

$$|P(s, a, s') - P'(s, a, s')| < \frac{1}{n} \frac{\varepsilon^2}{72n\ell}$$

where $n = |S|$ is the number of states in $\mathcal{M}$. i.e., for all outgoing transitions from explored states, the difference between the transition probabilities in $\mathcal{M}$ and $\mathcal{M}'$ is bounded.

We prove that Algorithm 1 generates a good approximation of the input MDP with high probability:

Lemma 5.1 (Good Approximation)**.** *Given MDP $\mathcal{M} = (S, A, s_0, P)$ with simulator $\mathbb{S}$, let $\mathcal{M}'$ and E be the estimated MDP and set of explored states, respectively, obtained by running* ApproximateMDP *on $\mathbb{S}$ with ECD ℓ, error $\varepsilon > 0$, and confidence $p > 0$. Then, with probability at least $1 - p/2$, $\mathcal{M}'$ is a good approximation of $\mathcal{M}$.*

Proof. Algorithm 1 constructs $\mathcal{M}'$ by estimating outgoing transition probabilities for all explored states by sampling each state-action pair in these states $\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times. We use Chernoff bounds to establish that these many samples are sufficient to ensure that $\mathcal{M}'$ is a good approximation with high probability.

Let $s \in E$ be an explored state and $a \in A$ be an action. We are interested in bounding the error on the estimated transition probability $P'(s, a, s')$ for state $s' \in S$. When s is explored, state-action pair (s, a) has to be sampled

$\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times. Let X_i be the random variable that sampling action a in state s results in visiting state s' in the i-th sample. Clearly, all X_is are independent.

Then, let random variable X be the sum of all X_is. Then, let $\mu = E[X] = P'(s,a,s') \cdot T$ where T is the number of samples. Then, from Chernoff bounds, we get that for all $0 < \delta < 1$,

$$\Pr\left(|X - \mu| \geq \delta\mu\right) \leq 2e^{-\frac{\delta^2\mu}{3}}.$$

Let $p' = P(s,a,s')$. Since (s,a) is sampled $\Theta\left(n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$ times, we have $\mu = \Theta\left(p' n^4 \frac{\ell^2}{\varepsilon^4} \log \frac{n^2|A|}{p}\right)$. Then, by setting $\delta = \frac{1}{np'} \frac{\varepsilon^2}{72n\ell}$, we get

$$\Pr\left(|X - \mu| \geq \frac{1}{np'} \frac{\varepsilon^2}{72n\ell} \cdot \mu\right) \leq e^{-\Theta\left(\frac{1}{p'} \log \frac{n^2|A|}{p}\right)} \leq \frac{p}{2n^2|A|}.$$

In other words, $|P(s,a,s') - P'(s,a,s')| \geq \frac{1}{n} \frac{\varepsilon^2}{72n\ell}$ with probability at most $\frac{p}{2n^2|A|}$. Alternately, $|P(s,a,s') - P'(s,a,s')| < \frac{1}{n} \frac{\varepsilon^2}{72n\ell}$ with probability at least $1 - \frac{p}{n^2|A|}$.

We are interested in computing the probability that the error on all transition probabilities from explored states is bounded. The above shows that for each (s,a,s'), the probability that the error is unbounded is at most $\frac{p}{2n^2|A|}$. Since there are $n^2|A|$ such tuples, by union bound, the probability that the error in at least one tuple is unbounded is $p/2$. Therefore, the probability that the error is bounded across all (s,a,s') tuples is at least $1 - p/2$. □

An immediate corollary is that for all $s \in E$ and $a \in A$, we have

$$||P(s,a) - P'(s,a)||_1 \leq \frac{\varepsilon^2}{72n\ell}$$

with probability at least $1 - p/2$. This is obtained by simply applying union-bound to the error margins.

In the following lemma, we show that if a policy has a high chance of reaching T after leaving E, ECD of the policy is above ℓ.

Lemma 5.2. *Let $\mathcal{M}', E$ be the output of running Algorithm 1 on MDP $\mathcal{M}$ with ECD ℓ. Let $\mathcal{M}'$ be a good approximation of $\mathcal{M}$. Let $T \subseteq S$ be the target states. Let policy π in $\mathcal{M}'$ be such that π leaves E and then reach target T with probability at least $\varepsilon/3$, then $\mathsf{ECD}^T_{\mathcal{M}}(\pi) > \ell$.*

Proof. For the sake of contradiction, let us assume that there exists π that leaves E and then reaches the target with probability at least $\varepsilon/3$, and $\mathsf{ECD}^T_{\mathcal{M}}(\pi) \leq \ell$. Let us look only at trajectories that leave E and reach T, then the expected length of these trajectories is at most $\frac{3\ell}{\varepsilon}$, otherwise we have $\mathsf{ECD}^T_{\mathcal{M}}(\pi) > \ell$ trivially from the definition of ECD. Still looking at trajectories that leave E and reach T, from Markov's inequality, we have that after $\frac{6\ell}{\varepsilon}$ steps, at least half of them had to reach the target already. This means, following π for $6\ell/\varepsilon$ steps, with probability at least $\varepsilon/6$, we need to leave E (and also reach T).

Now, let us look at $\mathcal{M}'$ and follow π there. Since $\mathcal{M}'$ is a good approximation of $\mathcal{M}$, all estimated transition probabilities are close to the real transition probability. The distribution after one step in $\mathcal{M}'$ differs by at most $\frac{\varepsilon^2}{72n\ell}$ from the distribution in $\mathcal{M}$. From union bound, we have that the distribution distribution changes by at most $t\frac{\varepsilon^2}{72n\ell}$ after t steps. Setting $t = 6\ell/\varepsilon$, we get the change at most $\frac{\varepsilon}{12n}$. Since the reachability probability was at least $\frac{\varepsilon}{6}$ before, now it it at least $\frac{\varepsilon}{7}$ (for $n \geq 4$).

Therefore, following π in $\mathcal{M}'$ for $6\ell/\varepsilon$ steps reaches a state outside E with probability at least $\varepsilon/7$. There are at most n states outside E, which means, from Dirichlet's principle, one state s' is reached with probability at least $\frac{\varepsilon}{7n}$. However, from path probability, we have that any state with $p_{\mathcal{M}'}(s_0, s', 6\ell/\varepsilon) \geq \frac{\varepsilon}{7n}$ was explored, which is a contradiction. □

Algorithm 2 Algorithm to find $\pi \in \Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)$.

Input: $\mathcal{M}, T, \varepsilon, \ell$
Output: $v_{0,0,\ell}$
$\delta \leftarrow \frac{\varepsilon^2}{n\ell}$
$v_{i,j,k} = 0$ for $i \in [|S|], \quad j \in [4\ell/\varepsilon], \quad k \in \delta[\frac{4\ell}{\varepsilon\delta}]$
{Index k goes from 0 to $4\ell/\varepsilon$ in increments of δ.}
$v_{i,j,k} = 1$ for $s_i \in T$
for $j = 4\ell/\varepsilon - 1; \quad j \geq 0; \quad j = j - 1$ **do**
  for $i = 0; \quad i < |S|; \quad i = i + 1$ **do**
    for $t = 0; \quad t < |A|; \quad t = t + 1$ **do**
      $S_{i,j+1,t} = \sum_{s_{i'} \in S} p_{i,t,i'} v_{i',j+1}$
      {$S_{i,j+1,t}$ is a vector and sum performed by Algorithm 3.}
    end for
    $v_{i,j} = \text{Shift}(\sum_{t \in A} S_{i,j+1,t})$
    {$v_{i,j}$ is a vector and sum is performed by Algorithm 4.}
  end for
end for

5.2. Approximation algorithm for ECD

We describe a parametrized algorithm to find the ε-approximation to the optimal policy in $\Pi_{\leq \ell, T}$ in a *known MDP*. We suppose the policy can terminate at any time, thus contributing 0 to the ECD.

Algorithm Description. Algorithm 2 computes the values $v_{i,j,k}$, defined as the reachability probability from state s_i after j steps with ECD at most k. The algorithm can also recover the policy by remembering optimal decisions leading to the value. These values are computed for the policy that ends after $4\ell/\varepsilon$ steps. After initialization of the target states to 1, the algorithm computes $v_{i,j,k}$ from $j = 4\ell/\varepsilon - 1$ down to $j = 0$ for all i ($s_i \in S$) and for all k that is the discretization of the actual ECD. The algorithm uses three subroutines described in Appendix C. Algorithm 3 computes the (discretized) function $S_{i,j+1,t}$ that tracks the expected reachability after choosing action a_t in state s_i in the j-th step. Algorithm 4 computes the function $v_{i,j}$ that tracks the expected reachability after selecting optimal weights of possible actions (combination of $S_{i,j+1,t}$). Algorithm Shift takes into account that one step was taken.

Lemma 5.3 (Key lemma for approximation)**.** *Given a known MDP $\mathcal{M}$, target set T, parameter ε, and length ℓ, in time $\mathcal{O}(n^4|A|\ell^7\varepsilon^{-10})$ Algorithm 2 finds a policy π s.t.*

$$\pi \in \Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)\,.$$

Proof sketch. For the detailed proof, see Appendix C. Here, we describe the main ideas for the proof and algorithm. First, we consider paths that reach the target within $4\ell/\varepsilon$ steps, this decreases the reachability probability by at most $\varepsilon/4$.

We define a function $v_{i,j}(k)$ that tracks the reachability probability for state s_i after j steps where the ECD from this position is at most k. We initialize $v_{i,j}(k) = 0$ if $j + k > 4\ell/\varepsilon$ and $v_{i,j}(k) = 1$ for all $s_i \in T$. We can express $v_{i,j}(k)$ as recursive maximization over two sets $a = \{a_1, a_2, \ldots, a_{|A|}\}$ and $K = \{k_{t,i'} \mid t \in A; s_{i'} \in S\}$ as follows:

$$v_{i,j}(k) = \max_{a,K} \sum_{t\in A} a_t \sum_{s'_i \in S} p_{i,t,i'} v_{i',j+1}(k_{t,i'})\,,$$

where the set a is a distribution over actions (so $\sum_{i=1}^{|A|} a_i = 1$) and the set K is the set of examined lengths, and the following holds: $\sum_{t=0}^{|A|-2} a_t \left(1 + \sum_{s_{i'}\in S} p_{i,t,i'} k_{t,i'}\right) \le k$, where the last action is the one that ends the process without reaching. This gives us the values of the optimal policy.

We compute the optimal values using Algorithm 2 that discretizes the the functions $v_{i,j}$ to increments of $\delta = \frac{\varepsilon^2}{n\ell}$. Then, since all the functions are concave, we can compute good approximation of the values. □

5.3. Final Algorithm

We finally describe a learning algorithm that is PAC-MDP with ECD.

Algorithm Description. Our final algorithm is described as follows: Given inputs $\mathbb{S}, S, \ell, \varepsilon, p, T$, first run Algorithm 1 on inputs $(\mathbb{S}, S, \ell, \varepsilon, p)$ to obtain an approximate $\mathcal{M}'$ with states E. Next, run Algorithm 2 on inputs $\mathcal{M}', T, \varepsilon/3, \ell$. Let its output be π. Then the following holds:

Theorem 5.4. *Given $\mathbb{S}, S, \ell, \varepsilon, p, T$, let π be the output of the above algorithm. Then with probability at least $1 - p$,*

$$\pi \in \Pi^{\varepsilon}_{\mathsf{opt}}(\mathcal{M}, T, \ell)\,.$$

Moreover, the expected sample complexity of the algorithm is $\mathcal{O}\left(n^6|A|\ell^3\varepsilon^{-6}\log\frac{n^2|A|}{p}\log\frac{n|A|\ell}{\varepsilon p}\right)$.

Proof. We show that π is ε-optimal by modifying some optimal policy $\pi_{\mathsf{opt}} \in \Pi_{\mathsf{opt}}(\mathcal{M}, T, \ell)$ in the following way: (a) We restrict the policy to the set of explored states $E \subseteq S$. (b) We consider an approximation of the MDP $\mathcal{M}'$ instead of the original MDP $\mathcal{M}$. (c) We compute an approximation of the optimal policy in $\mathcal{M}'$. (d) We show that an approximation of the optimal policy in $\mathcal{M}'$ is also an approximation of the optimal policy in $\mathcal{M}$.

With probability at least $1 - p$ the MDP $\mathcal{M}'$ is good approximation of $\mathcal{M}$ and Explore did not fail (see Lemma 5.1 and Lemma B.1). In the rest of the proof, we suppose that all pairs (s, a) for $s \in E$ and $a \in A$ the following holds: $||P(s,a) - P'(s,a)||_1 \le \frac{\varepsilon^2}{72n\ell}$.

Given $\pi_{\mathsf{opt}} \in \Pi_{\mathsf{opt}}(\mathcal{M}, T, \ell)$, we can create a policy π' such that if the policy leaves E (the set of explored states), it is considered that the condition is not satisfied. Since π' sometimes ends which decreases the reachability probability, we have $\mathsf{ECD}(\pi') \le \mathsf{ECD}(\pi_{\mathsf{opt}})$. Since $\mathsf{ECD}^T_{\mathcal{M}}(\pi_{\mathsf{opt}}) \le \ell$, we know from Lemma 5.2 that with probability less than $\varepsilon/3$ the policy π_{opt} leaves E and then reaches the target, therefore $J^{\mathcal{M}}_T(\pi') \ge J^{\mathcal{M}}_T(\pi_{\mathsf{opt}}) - \varepsilon/3$.

In $\mathcal{M}$ the policy π' achieves reachability $J^{\mathcal{M}}_T(\pi')$. We examine $J^{\mathcal{M}'}_T(\pi')$, the payoff of policy π' on the approximated MDP $\mathcal{M}'$. We can group all runs with $Len(\zeta) = i$ and denote p_i their probability mass. We have $\mathsf{ECD}_{\mathcal{M}}(\pi') = \sum_{i=0}^{\infty} i \cdot p_i \le \ell$. After one step, the distribution of outcomes differs by at most $\frac{\varepsilon^2}{72n\ell}$ which is the maximal error in one step. From union bound, after i steps, the error is at most $i \cdot \frac{\varepsilon^2}{72n\ell}$. In other words, the error that we get from all runs for which $Len(\zeta) = i$ is at most $p_i \cdot i\frac{\varepsilon^2}{72n\ell}$, now we bound the error

$$\sum_{i=0}^{\infty} i\frac{\varepsilon^2}{72n\ell} \cdot p_i < \frac{\varepsilon^2}{72n}\,.$$

By using $\mathcal{M}'$, instead of $\mathcal{M}$ restricted on E, the reachability probability decreases by at most $\frac{\varepsilon^2}{72n}$. We know the MDP $\mathcal{M}'$ and from Lemma 5.3, we can find a policy $\pi \in \Pi^{\varepsilon/3}_{\mathsf{opt}}(\mathcal{M}', T, \ell)$ in polynomial time.

Finally, we need to examine how π performs on $\mathcal{M}$. Again, $\mathcal{M}'$ is a good approximation, so by the same argument as

going from $\mathcal{M}$ to $\mathcal{M}'$ we get an error at most $\sum_{i=0}^{\infty} i\frac{\varepsilon^2}{72n\ell} \cdot p_i < \frac{\varepsilon^2}{72n}$. Here, π can have a higher ECD, where the mistake might be instead of decreasing reachability, it is delayed, but we can treat these paths also as not reaching T.

This means that policy π is by at most $\frac{\varepsilon^2}{72n}$ worse on $\mathcal{M}$ than on $\mathcal{M}'$ where it is by at most $\varepsilon/3$ worse than the optimal policy $\mathcal{M}'$, which is by at most $\frac{\varepsilon^2}{72n}$ worse than the optimal policy on $\mathcal{M}$ restricted on E, which is by at most $\varepsilon/3$ worse than the optimal policy on $\mathcal{M}$. This gives the difference $J^T_{\mathcal{M}}(\pi_{\mathsf{opt}}) - \varepsilon < J^T_{\mathcal{M}}(\pi)$.

Every state-action pair for state in E is examined by Explore at least $n^4\frac{\ell^2}{\varepsilon^4}\log\frac{n^2|A|}{p}$. Moreover, every state needs to be reached. Lemma B.1 gives the upper bound on the number of calls on the simulator, $\mathcal{O}\left(\frac{n\ell}{\varepsilon^2}\log\frac{n\ell|A|}{\varepsilon p}\right)$. The product gives the final expected sample complexity $\mathcal{O}\left(n^6|A|\ell^3\varepsilon^{-6}\log\frac{n^2|A|}{p}\log\frac{n|A|\ell}{\varepsilon p}\right)$.

The algorithm is in polynomial time. Algorithm 1 repeatedly explores and needs to compute the path probability, which can be computed by Algorithm 2. Lemma 5.3 shows that the approximate policy can be computed by Algorithm 2 in polynomial time. □

Corollary 5.5. *Reachability specifications are efficient PAC-MDP learnable with ECD.*

Remark 5.6 (Significance). The above result along with Remark 3.4 shows that with the ECD information efficient PAC-MDP is possible. Along with our impossibility result (Remark 4.3), our results present a tight characterization of efficient PAC-MDP wrt to the ECD information. More precisely, our results show that ECD information provides a necessary and sufficient parametrization for efficient PAC-MDP problem.

6. Concluding remarks

This work presents the first PAC-possible result in RL from reachability specifications that is purely environment-agnostic. The additional parameter used to obtain the guarantee, the expected conditional distance (ECD), is user-defined external parameter. In contrast, the additional parameterization in prior PAC possibility results is an internal parameter of the environment, hence requiring additional knowledge of the environment to obtain the PAC guarantee. Remark 5.6 from above further elaborates on the necessity and sufficiency of the ECD to obtain efficient PAC guarantee under reachability objectives.

This work opens up several future directions, including (a). examination of ECD parameterization under richer classes of qualitative specifications such as ω-regular objectives, safety objectives, LTL, etc, (b). model-free PAC algorithms with ECD, and (c). empirical evaluations to improve the practical feasibility of these approaches.

Impact Statement

This paper presents work whose goal is to advance the trustworthy Machine Learning, in particular the theory of RL under qualitative specifications. The contributions of this work are primarily theoretical and do not warrant any immediate societal consequence, to the best of our knowledge.

References

Aksaray, D., Jones, A., Kong, Z., Schwager, M., and Belta, C. Q-learning for robust satisfaction of signal temporal logic specifications. In *Conference on Decision and Control (CDC)*, pp. 6565–6570. IEEE, 2016.

Alur, R., Bansal, S., Bastani, O., and Jothimurugan, K. A framework for transforming specifications in reinforcement learning. In *Principles of Systems Design: Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday*, pp. 604–624. Springer, 2022.

Alur, R., Bastani, O., Jothimurugan, K., Perez, M., Somenzi, F., and Trivedi, A. Policy synthesis and reinforcement learning for discounted ltl. *Intenational Conference on Computer-Aided Verification*, 2023.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Ashok, P., Křetínský, J., and Weininger, M. Pac statistical model checking for markov decision processes and stochastic games. In *International Conference on Computer Aided Verification*, pp. 497–519. Springer, 2019.

Bertsekas, D. P. and Tsitsiklis, J. N. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Brafman, R., De Giacomo, G., and Patrizi, F. Ltlf/ldlf non-markovian rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

Chatterjee, K. and Henzinger, M. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In Randall, D. (ed.), *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*

2011, San Francisco, California, USA, January 23-25, 2011, pp. 1318–1336. SIAM, 2011.

Chatterjee, K. and Henzinger, M. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *J. ACM*, 61(3):15:1–15:40, 2014.

Collins, S., Ruina, A., Tedrake, R., and Wisse, M. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

Courcoubetis, C. and Yannakakis, M. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

De Giacomo, G., Vardi, M. Y., et al. Linear temporal logic and linear dynamic logic on finite traces. In *Ijcai*, volume 13, pp. 854–860, 2013.

De Giacomo, G., Iocchi, L., Favorito, M., and Patrizi, F. Foundations for restraining bolts: Reinforcement learning with ltlf/ldlf restraining specifications. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pp. 128–136, 2019.

Fu, J. and Topcu, U. Probably approximately correct MDP learning and control with temporal logic constraints. In *Robotics: Science and Systems*, 2014.

Hahn, E. M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., and Wojtczak, D. Omega-regular objectives in model-free reinforcement learning. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 395–412, 2019.

Hasanbeig, M., Abate, A., and Kroening, D. Logically-constrained reinforcement learning. *arXiv preprint arXiv:1801.08099*, 2018.

Hasanbeig, M., Kantaros, Y., Abate, A., Kroening, D., Pappas, G. J., and Lee, I. Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees. In *Conference on Decision and Control (CDC)*, pp. 5338–5343, 2019.

Icarte, R. T., Klassen, T., Valenzano, R., and McIlraith, S. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pp. 2107–2116. PMLR, 2018.

Inala, J. P., Yang, Y., Paulos, J., Pu, Y., Bastani, O., Kumar, V., Rinard, M., and Solar-Lezama, A. Neurosymbolic transformers for multi-agent communication. *arXiv preprint arXiv:2101.03238*, 2021.

Jiang, Y., Bharadwaj, S., Wu, B., Shah, R., Topcu, U., and Stone, P. Temporal-logic-based reward shaping for continuing learning tasks, 2020.

Jothimurugan, K., Bansal, S., Bastani, O., and Alur, R. Compositional reinforcement learning from logical specifications. *Advances in Neural Information Processing Systems*, 34:10026–10039, 2021.

Jothimurugan, K., Bansal, S., Bastani, O., and Alur, R. Specification-guided learning of nash equilibria with high social welfare. In *International Conference on Computer Aided Verification*, pp. 343–363. Springer, 2022.

Kakade, S. M. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Li, X., Vasile, C.-I., and Belta, C. Reinforcement learning with temporal logic rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3834–3839. IEEE, 2017.

Littman, M. L., Topcu, U., Fu, J., Isbell, C., Wen, M., and MacGlashan, J. Environment-independent task specifications via GLTL. *arXiv preprint arXiv:1704.04341*, 2017a.

Littman, M. L., Topcu, U., Fu, J., Isbell, C., Wen, M., and MacGlashan, J. Environment-independent task specifications via GLTL, 2017b.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

Perez, M., Somenzi, F., and Trivedi, A. A pac learning algorithm for ltl and omega-regular objectives in mdps. *arXiv preprint arXiv:2310.12248*, 2023.

Pnueli, A. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pp. 46–57. IEEE, 1977.

Safra, S. On the complexity of omega-automata. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pp. 319–327. IEEE Computer Society, 1988.

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

Valiant, L. G. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Xu, Z. and Topcu, U. Transfer of temporal logic formulas in reinforcement learning. In *International Joint Conference on Artificial Intelligence*, pp. 4010–4018, 7 2019.

Yang, C., Littman, M., and Carbin, M. Reinforcement learning for general ltl objectives is intractable. *arXiv preprint arXiv:2111.12679*, 2021.

Yuan, L. Z., Hasanbeig, M., Abate, A., and Kroening, D. Modular deep reinforcement learning with temporal logic specifications. *arXiv preprint arXiv:1909.11591*, 2019.

Algorithm 3 Sum of v and v' with given weights.

Input: v, v', p, p'
Output: S
for $k = 4\ell/(\delta\varepsilon)$; $k \geq 0$; $k = k - 1$ **do**
  for $k' = 4\ell/(\delta\varepsilon)$; $k' \geq 0$; $k' = k' - 1$ **do**
    $t \leftarrow \frac{1}{\delta} \cdot \lceil \delta(p \cdot k + p' \cdot k') \rceil$
    $S_t = \max(S_t, p \cdot v_k + p' \cdot v'_{k'})$
  end for
end for

A. Proof of Lemma 4.1

Lemma A.1. *There exists $\mathcal{M}$ and $\mathcal{L}(T)$ such for every π holds*

$$\mathsf{ECD}_T^{\mathcal{M}}(\pi) \geq p^{-|S|+1},$$

where p is the smallest transition probability.

Proof. The set of states of $\mathcal{M}$ is denoted $s_0, s_1, \ldots, s_{n-1}$. The set of actions $|A| = 1$ holds, which means there is only one policy. For $i < n-1$ holds $P(s_i, a, s_0) = 1 - p$ and $P(s_i, a, s_{i+1}) = p$. The specification $\mathcal{L}(T)$ is simple reachability with $T = \{s_{n-1}\}$. Figure 1 shows the structure of $\mathcal{M}$.

Let t_i be the ECD starting from s_i. We have $t_{n-1} = 0$, $t_0 = 1 + pt_1 + (1-p)t_0$, and $t_i = 1 + pt_{i+1} + (1-p)t_0$. The solution of the system of equations is $t_0 = \sum_{i=1}^{n-1} p^{-i} \geq p^{-n+1}$, which proves the statement. □

B. Technical details of Explore

The function $\mathrm{Explore}(s)$ uses a policy π that within $6\ell/\varepsilon$ steps reaches s with the highest probability. It follows π for $6\ell/\varepsilon$ steps and either s is reached or restarts and begins from s_0 again. If s is not reached within $\Theta(\frac{n}{\varepsilon} \log \frac{n\ell|A|}{\varepsilon p})$ trials, Explore fails and the algorithm ends. This happens with a very small probability.

Lemma B.1. *One call on* $\mathrm{Explore}(s)$ *requires* $\Theta(\ell/\varepsilon \cdot \frac{n}{\varepsilon} \log \frac{n\ell|A|}{\varepsilon p})$ *calls on the simulator and* Explore *fails with probability at most* $p/2$.

Proof. We know from the proof of Lemma 5.2 that the probability on $\mathcal{M}'$ that is a good approximation of $\mathcal{M}$ changes only by a $\frac{\varepsilon}{12n}$ within in $6\ell/\varepsilon$ steps. That means policy π on $\mathcal{M}$ reaches s within $6\ell/\varepsilon$ steps with probability at least $\frac{\varepsilon}{7n} - \frac{\varepsilon}{12n} > \frac{\varepsilon}{20n}$.

That means after t trials (runs of length $6\ell/\varepsilon$), the state $s \in S \setminus E$ is not reached with probability at least

$$\left(1 - \frac{\varepsilon}{20n}\right)^t.$$

Setting $t = \Theta(\frac{n}{\varepsilon} \log \frac{n\ell|A|}{\varepsilon p})$ gives the failure probability of not reaching s below $p/2$ for all reachabilities that need to be satisfied by Explore. □

C. Approximation of optimal policy with ECD

In this section, we prove in detail Lemma 5.3. We also present Algorithm 3 and Algorithm 4.

We suppose that the MDP has a special state $s_\perp$ and action $a_\perp$ from every state, such that $P(s, a_\perp, s_\perp) = 1$ this state is not in the target set and it allows the policy to give up at any time and shorten its ECD.

Lemma C.1 (Key lemma for approximation)**.** *Given MDP $\mathcal{M}$, target set T, parameter ε, and length ℓ, Algorithm 2 in time $\mathcal{O}(n^4|A|\ell^7\varepsilon^{-10})$ finds policy π s.t.*

$$\pi \in \Pi_{\mathsf{opt}}^{\varepsilon}(\mathcal{M}, T, \ell).$$

Algorithm 4 Weighted sum of two functions v and v'.

Input: v, v'
Output: S
for $k = 4\ell/(\delta\varepsilon)$; $k \geq 0$; $k = k - 1$ **do**
  for $k' = 4\ell/(\delta\varepsilon)$; $k' \geq 0$; $k' = k' - 1$ **do**
    for $t = \max(k, k')$; $t \geq \min(k, k')$; $t = t - 1$ **do**
      $a \leftarrow \frac{t-k}{k-k'}$
      $S_t = \max(S_t, a \cdot v_k + (1-a) \cdot v'_{k'})$
    end for
  end for
end for

Proof. We first argue why we can look only at the finite horizon. Then we describe how to express optimal policy for the finite horizon. Finally, we show that Algorithm 2 finds the correct value of ε-optimal policy, which means we can recover the policy.

Suppose that for policy π holds $\mathsf{ECD}(\pi) \leq \ell$. The total probability that π reaches the target after $4\ell/\varepsilon$ steps is at most $\frac{\varepsilon}{4}$, otherwise, the ECD is above ℓ. That means we can focus only on the first $4\ell/\varepsilon$ steps. This decreases the reachability probability by at most $\varepsilon/4$ and does not increase ECD. We consider optimal policy π (not necessarily positional) with ECD at most ℓ and that ends in $4\ell/\varepsilon$ steps. We define a function $v_{i,j}(k)$ that tracks the reachability probability for state s_i after j steps where the ECD from this position is at most k. We are looking for $v_{0,0}(\ell)$. Moreover, since the policy needs to finish after $4\ell/\varepsilon$ steps, we have $v_{i,j}(k) = 0$ if $j + k \geq 4\ell/\varepsilon$. For state $s_i \in T$ holds $s_{i,j}(k) = 1$ for all j and k. The policy π is not necessarily positional on $\mathcal{M}$, but it can consider only the length of the policy, so knowing, the current state and length, it is positional. We can express $v_{i,j}(k)$ as recursive maximization over two sets $\alpha = \{a_1, a_2, \ldots, a_{|A|}\}$ and $K = \{k_{t,i'} \mid t \in A; s_{i'} \in S\}$ as follows:

$$v_{i,j}(k) = \max_{\alpha,K} \sum_{t \in A} a_t \sum_{s'_i \in S} p_{i,t,i'} v_{i',j+1}(k_{t,i'}) \,,$$

where the set α is a distribution over actions (so $\sum_{i=1}^{|A|} a_i = 1$) and the set K is the set of examined lengths, and the following holds:

$$\sum_{t=0}^{|A|-2} a_t \left(1 + \sum_{s_{i'} \in S} p_{i,t,i'} k_{t,i'}\right) \leq k \,,$$

where the last action is the one that ends the process without reaching.

We prove the recursive equation by induction by j (number of steps already performed). Reachability from the target is 1 in 0 steps, and if the policy ends after $4\ell/\varepsilon$ steps, the variables representing reachability after more steps are 0. That means the initialization was correct. Then, computing $v_{i,j}(k)$ while knowing values of $v_{i,j'}(k)$ for all i, k and $j' > j$ is optimal. We consider all policies from s_i and look at all possible combinations of how long we continue. That means $v_{i,j}(k)$ is also optimal.

Now, to compute approximation of $v_{i,j}(k)$, we discretize k to increments of δ. For all parameters of i, j we have $\frac{4\ell}{\varepsilon\delta}$ variables $v_{i,j,k}$ (k is multiple of δ). As in the exact case, we want a similar recursive equation. Now, $v_{i,j,*}$ is a vector, but we can treat it also as a function.

First, observe that $v_{i,j,k} \geq v_{i,j,k'}$ iff $k \geq k'$. Moreover, if we view $v_{i,j}$ as a function of k, it is concave since for any two points $v_{i,j,k+x}$ and $v_{i,j,k}$ we can create a policy that ends with probability $\frac{k+x}{k}$, otherwise continues with a policy that guarantees $v_{i,j,k+x}$. From that, also $|v_{i,j,k+\delta} - v_{i,j,k}| \leq \delta$.

The recursive equation looks similar

$$v_{i,j,k} = \max_{\alpha,K} \sum_{t \in A} a_t \sum_{s'_i \in S} p_{i,t,i'} v_{i',j+1,k_{t,i'}} \,,$$

with a change that elements of K are multiples of δ.

Algorithm 5 PAC-MDP with ECD Learning Algorithm

1: **Input:** $\mathbb{S}, S, \ell, \varepsilon, p, T$
2: **Output:** π
3: $\mathcal{M}', E \leftarrow$ Algorithm 1$(\mathbb{S}, S, \ell, \varepsilon, p)$
4: $\pi \leftarrow$ Algorithm 2$(\mathcal{M}', T, \varepsilon/3, \ell)$
5: **Return** π

Let us describe how to compute the sum $\sum_{s_{i'} \in S} p_{i,t,i'} v_{i',j+1,k_{t,i'}}$ for all $k_{t,i'}$. Algorithm 3 describes how to sum two functions. To sum up more functions, we apply the function iteratively. We have a function S_* initialized to 0 that stores the results. Given two functions v_* and v'_* with p and p', we compute for all pairs of k, k' the following: $p \cdot v_k + p' \cdot v'_{k'}$ and if the sum is bigger than the current value, we store it into S_*, under index $\frac{1}{\delta} \cdot \lceil \delta(p \cdot k + p' \cdot k') \rceil$, which is $p \cdot k + p' \cdot k'$ rounded up to the nearest multiple of δ. Summing two functions requires $\mathcal{O}(\delta^{-2}\ell^2\varepsilon^{-2})$ steps and every sum is at most $\mathcal{O}(\delta)$ approximation of the perfect convolution. To sum up all possible n results of one action, we require time $\mathcal{O}(\delta^{-2}\ell^2\varepsilon^{-2}n)$ and the error is at most $\mathcal{O}(n\delta)$. We denote the convolution of $\sum_{s'_i \in S} p_{i,t,i'} v_{i',j+1,k_{t,i'}}$ as $S_{i,t,j+1,*}$. Again, this function is concave.

Now, we need to maximize

$$v_{i,j,k} = \max_{\alpha,K} \sum_{t \in A} a_t S_{t,j+1,k_t} \,.$$

Again, we show how to compute the sum of two functions, v and v', to obtain S. Algorithm 4 shows the sum. The functions v and v' are concave. To obtain function S, we again consider v and v' for all indices k and k'. Then, for all lengths t (it makes sense to consider only $\min(k, k') \leq t \leq \max(k, k')$), and a such that $ak + (1-a)k' = t$, we have that S_t can be updated by $a \cdot v_k + (1-a) \cdot v'_{k'}$. This gives us function S which is the optimal mix of v and v' in every point. Therefore, we can evaluate $\max_{\alpha,K} \sum_{t \in A} a_t S_{t,j+1,k_t}$ exactly and in time $\mathcal{O}(\delta^{-3}\ell^3\varepsilon^{-3} \cdot |A|)$.

That means after computing $v_{i,j,k}$ for all j, we get the error at most $\mathcal{O}(n\delta\ell/\varepsilon)$. The time of computing everything is $\mathcal{O}(\frac{\ell}{\varepsilon} n(\delta^{-3}\ell^3\varepsilon^{-3} \cdot |A| + \delta^{-2}\ell^2\varepsilon^{-2} \cdot n))$. Setting $\delta = \mathcal{O}(\frac{\varepsilon^2}{n\ell})$, we get error at most $\varepsilon/2$ and time $\mathcal{O}(n^4|A|\ell^7\varepsilon^{-10})$.

The described algorithm computes the reachability probability, to retrieve policy π, we remember the maximum of every decision. □

D. Algorithm 5

In this section, we present Algorithm 5 for completeness.