Sample-Efficient Reinforcement Learning with Temporal Logic Objectives: Leveraging the Task Specification to Guide Exploration

Yiannis Kantaros, *Member, IEEE*, Jun Wang *Student Member, IEEE* .

***Abstract*—This paper addresses the problem of learning optimal control policies for systems with uncertain dynamics and high-level control objectives specified as Linear Temporal Logic (LTL) formulas. Uncertainty is considered in the workspace structure and the outcomes of control decisions giving rise to an unknown Markov Decision Process (MDP). Existing reinforcement learning (RL) algorithms for LTL tasks typically rely on exploring a product MDP state-space uniformly (using e.g., an ϵ-greedy policy) compromising sample-efficiency. This issue becomes more pronounced as the rewards get sparser and the MDP size or the task complexity increase. In this paper, we propose an accelerated RL algorithm that can learn control policies significantly faster than competitive approaches. Its sample-efficiency relies on a novel task-driven exploration strategy that biases exploration towards directions that may contribute to task satisfaction. We provide theoretical analysis and extensive comparative experiments demonstrating the sample-efficiency of the proposed method. The benefit of our method becomes more evident as the task complexity or the MDP size increases.**

***Index Terms*—Reinforcement Learning, Temporal Logic Planning, Stochastic Systems**

I. Introduction

Reinforcement learning (RL) has been successfully applied to synthesize control policies for systems with highly nonlinear, stochastic or unknown dynamics and complex tasks [1]. Typically, in RL, control objectives are specified as reward functions. However, specifying reward-based objectives can be highly non-intuitive, especially for complex tasks, while poorly designed rewards can significantly compromise system performance [2]. To address this challenge, Linear Temporal logic (LTL) has recently been employed to specify tasks that would have been very hard to define using Markovian rewards [3]; e.g., consider a navigation task requiring to visit regions of interest in a specific order.

Several model-free RL methods with LTL-encoded tasks have been proposed recently; see e.g., [4]–[15]. Common in the majority of these works is that they explore *randomly* a product state space that grows exponentially as the size of the MDP and/or the complexity of the assigned temporal logic task increase. This results in sample inefficiency and slow training/learning process. This issue becomes more pronounced by the fact that LTL specifications are converted into sparse rewards in order to synthesize control policies with probabilistic satisfaction guarantees [9], [14], [16].

Yiannis Kantaros (ioannisk@wustl.edu) and Jun Wang (junw@wustl.edu) are with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO, 63130, USA. This work was supported by the NSF award CNS #2231257.

Sample inefficiency is a well-known limitation in RL, whether control objectives are specified using reward functions directly or LTL. To address this limitation, reward engineering approaches have been proposed augmenting the reward signal [17]–[23]. Such methods often require a user to *manually* decompose the global task into sub-tasks, followed by assigning additional rewards to these intermediate sub-tasks. Nevertheless, this may result in sub-optimal control policies concerning the original task [24], while their efficiency highly depends on the task decomposition (i.e., the density of the rewards) [25]. Also, augmenting the reward signal for temporal logic tasks may compromise the probabilistic correctness of the synthesized controllers [9]. To alleviate these limitations, intelligent exploration strategies have been proposed, such as Boltzmann/softmax [26], [27] and upper confidence bound (UCB) [28] that do not require knowledge or modification of the rewards; a recent survey is available in [29]. Their sample-efficiency relies on guiding exploration using a continuously learned value function (e.g., Boltzmann) which, however, can be inaccurate in early training episodes. Alternatively, they rely on how many times a state-action pair has been visited (e.g., UCB), which might not always guide exploration towards directions contributing to task satisfaction.

Another approach to enhance sample-efficiency is through model-based methods [30], [31]. These works continuously learn an unknown Markov Decision Process (MDP), modeling the system, that is composed with automaton representations of LTL tasks. This gives rise to a product MDP (PMDP). Then, approximately optimal policies are constructed for the PMDP in a finite number of iterations. However, saving the associated data structures for the PMDP results in excessive memory requirements. Also, the quality of the generated policy critically depends on the accuracy of the learned PMDP. Finally, model-based methods require the computation of accepting maximum end components (AMECs) of PMDPs that has a quadratic time complexity in the PMDP size. This computation is avoided in related model-free methods; see e.g., [6].

In this paper, we propose a novel approach to enhance the sample-efficiency of model-free RL methods. Unlike the aforementioned works, the key idea to improve sample efficiency is to leverage the (known) task specification in order to extract promising directions for exploration that contribute to mission progress. We consider robots modeled as unknown MDPs with discrete state and action spaces, modeling uncertainty in the workspace and in the outcome of control decisions, and high-level LTL-encoded control objectives. The proposed algorithm

relies on the following three steps. First, the LTL formula is converted into a Deterministic Rabin Automaton (DRA). Second, similar to [6], the product between the MDP and the DRA is constructed on-the-fly giving rise to a PMDP over which rewards are assigned based on the DRA acceptance condition. We note that the PMDP is not explicitly constructed/strored in our approach. The first two steps are common in related model-free algorithms. Third, a new RL method is applied over the PMDP to learn policies that maximize the expected accumulated reward capturing the satisfaction probability. The proposed RL algorithm relies on a new stochastic policy, called $(\epsilon, \delta)-$ greedy policy, that exploits the DRA representation of the LTL formula to bias exploration towards directions that may contribute to task satisfaction. Particularly, according to the proposed policy, the greedy action is selected with probability $1-\epsilon$ (exploitation phase) while exploration is triggered with probability ϵ, as in the ϵ-greedy policy. Unlike the ϵ-greedy policy, when exploration is enabled, either a random or a biased action is selected probabilistically (determined by δ parameters), where the latter action guides the system towards directions that will most likely result in mission progress. For instance, consider a simple scenario where a robot with uncertain/unknown dynamics is required to eventually safely reach a region of interest. In this case, intuitively, exploration in the vicinity of the shortest dynamically feasible path (that is initially unknown but it is continuously learned) connecting the current robot position to the desired region should be prioritized to accelerate control design. We emphasize that the proposed task-driven exploration strategy does not require knowledge or modification of the reward structure. As a result, it can be coupled with sparse rewards, as e.g., in [9], [13], resulting in probabilistically correct control policies as well as with augmented rewards, as e.g., in [20], [22], [25], to further accelerate the learning phase.

Our approach is inspired by transfer learning algorithms that leverage external teacher policies for 'similar' tasks to bias exploration [32]. To design a biased exploration strategy, in the absence of external policies, we build upon [33], [34] that propose a biased sampling-based strategy to synthesize temporal logic controllers for large-scale, but *deterministic*, multi-robot systems. Particularly, computation of the biased action requires (i) a distance function over the DRA state space, similarly constructed as in [33]–[36], to measure how far the system is from satisfying the assigned LTL task, and (ii) a continuously learned MDP model. The latter renders the proposed exploration strategy model-based. Thus, we would like to emphasize the following key differences with respect to related model-based RL methods discussed earlier. First, unlike existing model-based algorithms, the proposed method does not learn/store the PMDP model to compute the optimal policy. Instead, it learns only the MDP modeling the system, making it more memory efficient. Second, the quality of the learned policy is not contingent on the quality of the learned MDP model, distinguishing it from model-based methods. This is because our approach utilizes the MDP model solely for designing the biased action and, in fact, as it will be discussed in Section III-C, does not even require learning *all* MDP transition probabilities accurately. This is also supported by our numerical experiments where we empirically demonstrate sample efficiency of the proposed method against model inaccuracies. We provide comparative experiments demonstrating that the proposed learning algorithm outperforms in terms of sample-efficiency model-free RL methods that employ random (e.g., [6], [8], [9]), Boltzmann, and UCB exploration. The benefit of our approach becomes more pronounced as the size of the PMDP increases. We also provide comparisons against model-based methods showing that our method, as well as model-free baselines, are more memory-efficient and, therefore, scalable to large MDPs. A preliminary version of this work was presented in [37]. We extend [37] by (i) providing theoretical results that help understand when the proposed approach is, probabilistically, more sample efficient than random exploration methods; (ii) providing more comprehensive comparative experiments that do not exist in [37]; and (iii) demonstrating how the biased sampling strategy can be extended to Limit Deterministic Buchi Automata (LDBA) that have smaller state space than DRA and, therefore, can further expedite the learning process [8], [38], [39]. We also release software implementing our proposed algorithm, which can be found in [40].

Contribution: *First*, we propose a novel RL algorithm to quickly learn control policies for *unknown* MDPs with LTL tasks. *Second*, we provide conditions under which the proposed algorithm is, probabilistically, more sample-efficient than related works that rely on random exploration. *Third*, we show that the proposed exploration strategy can be employed for various automaton representations of LTL formulas such as DRA and LDBA. *Fourth*, we provide extensive comparative experiments demonstrating the sample efficiency of the proposed method compared to related works.

II. Problem Definition

A. Robot & Environment Model

Consider a robot that resides in a partitioned environment with a finite number of states. To capture uncertainty in the robot motion and the workspace, we model the interaction of the robot with the environment as a Markov Decision Process (MDP) of unknown structure, which is defined as follows.

Definition 2.1 (MDP): A Markov Decision Process (MDP) is a tuple $\mathfrak{M} = (\mathcal{X}, x_0, \mathcal{A}, P, \mathcal{AP})$, where $\mathcal{X}$ is a finite set of states; $x_0 \in \mathcal{X}$ is an initial state; $\mathcal{A}$ is a finite set of actions. With slight abuse of notation $\mathcal{A}(x)$ denotes the available actions at state $x \in \mathcal{X}$; $P : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the transition probability function so that $P(x, a, x')$ is the transition probability from state $x \in \mathcal{X}$ to state $x' \in \mathcal{X}$ via control action $a \in \mathcal{A}$ and $\sum_{x' \in \mathcal{X}} P(x, a, x') = 1$, for all $a \in \mathcal{A}(x)$; $\mathcal{AP}$ is a set of atomic propositions; $L : \mathcal{X} \to 2^{\mathcal{AP}}$ is the labeling function that returns the atomic propositions that are satisfied at a state $x \in \mathcal{X}$.

Assumption 2.2 (Fully Observable MDP): We assume that the MDP $\mathfrak{M}$ is fully observable, i.e., at any time step t the current state, denoted by x_t, and the observations $L(x_t) \in 2^{\mathcal{AP}}$ in state x_t are known.

Assumption 2.3 (Static Environment): We assume that the environment is static in the sense that the atomic propositions that are satisfied at an MDP state x are fixed over time.

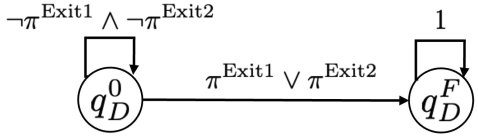

Fig. 1. DRA corresponding to $\phi = \Diamond(\pi^{\text{Exit1}} \vee \pi^{\text{Exit2}})$. There is only one set of accepting pairs defined as $\mathcal{G}_1 = \{q_D^F\}$ and $\mathcal{B}_1 = \{q_D^0\}$. A transition is enabled if the robot generates a symbol satisfying the Boolean formula noted on top of the transitions. All transitions are feasible as per Def. 3.1. The function d_F in (3) is defined as $d_F(q_D^0, \mathcal{F}) = 1$ and $d_F(q_D^F, \mathcal{F}) = 0$.

For instance, Assumption 2.3 implies that obstacles and regions of interest in the environment are static. This assumption can be relaxed using probabilistically labeled MDPs as in [8].

B. LTL-encoded Task Specification

The robot is responsible for accomplishing a task expressed as an LTL formula, such as sequencing, coverage, surveillance, data gathering or connectivity tasks [41]–[47]. LTL is a formal language that comprises a set of atomic propositions $\mathcal{AP}$, the Boolean operators, i.e., conjunction $\wedge$ and negation $\neg$, and two temporal operators, next $\bigcirc$ and until $\cup$. LTL formulas over a set $\mathcal{AP}$ can be constructed based on the following grammar: $\phi ::= \text{true} \mid \pi \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \bigcirc\phi \mid \phi_1 \cup \phi_2$, where $\pi \in \mathcal{AP}$. The other Boolean and temporal operators, e.g., *always* $\square$, have their standard syntax and meaning [3]. An infinite *word* w over the alphabet $2^{\mathcal{AP}}$ is defined as an infinite sequence $w = \pi_0\pi_1\pi_2 \cdots \in (2^{\mathcal{AP}})^\omega$, where ω denotes infinite repetition and $\pi_t \in 2^{\mathcal{AP}}$, $\forall t \in \mathbb{N}$. The language $\{w \in (2^{\mathcal{AP}})^\omega | w \models \phi\}$ is defined as the set of words that satisfy the LTL formula ϕ, where $\models \subseteq (2^{\mathcal{AP}})^\omega \times \phi$ is the satisfaction relation [3]. In what follows, we consider atomic propositions of the form π^i that are true if the robot is in state $x_i \in \mathcal{X}$ and false otherwise.

C. From LTL formulas to DRA

Any LTL formula can be translated into a Deterministic Rabin Automaton (DRA) defined as follows.

Definition 2.4 (DRA [3]): A DRA over $2^{\mathcal{AP}}$ is a tuple $\mathfrak{D} = (\mathcal{Q}_D, q_D^0, \Sigma, \delta_D, \mathcal{F})$, where $\mathcal{Q}_D$ is a finite set of states; $q_D^0 \subseteq Q_D$ is the initial state; $\Sigma = 2^{\mathcal{AP}}$ is the input alphabet; $\delta_D : \mathcal{Q}_D \times \Sigma_D \rightarrow \mathcal{Q}_D$ is the transition function; and $\mathcal{F} = \{(\mathcal{G}_1, \mathcal{B}_1), \ldots, (\mathcal{G}_f, \mathcal{B}_f)\}$ is a set of accepting pairs where $\mathcal{G}_i, \mathcal{B}_i \subseteq \mathcal{Q}_D, \forall i \in \{1, \ldots, f\}$. □

An infinite run $\rho_D = q_D^0 q_D^1 \ldots q_D^t \ldots$ of D over an infinite word $w = \sigma_0\sigma_1\sigma_2 \ldots$, where $\sigma_t \in \Sigma$, $\forall t \in \mathbb{N}$, is an infinite sequence of DRA states q_D^t, $\forall t \in \mathbb{N}$, such that $\delta(q_D^t, \sigma_t) = q_D^{t+1}$. An infinite run ρ_D is called *accepting* if there exists at least one pair $(\mathcal{G}_i, \mathcal{B}_i)$ such that $\texttt{Inf}(\rho_D) \cap \mathcal{G}_i \neq \varnothing$ and $\texttt{Inf}(\rho_D) \cap \mathcal{B}_i = \varnothing$, where $\texttt{Inf}(\rho_D)$ represents the set of states that appear in ρ_D infinitely often; see also Ex. 2.5.

Example 2.5 (DRA): Consider the LTL formula $\phi = \Diamond(\pi^{\text{Exit1}} \vee \pi^{\text{Exit2}})$ that is true if a robot eventually reaches either Exit1 or Exit2 of a building. The corresponding DRA is illustrated in Figure 1.

D. Product MDP

Given the MDP $\mathfrak{M}$ and the DRA $\mathfrak{D}$, we define the product MDP (PMDP) $\mathfrak{P} = \mathfrak{M} \times \mathfrak{D}$ as follows.

Definition 2.6 (PMDP): Given an MDP $\mathfrak{M} = (\mathcal{X}, x_0, \mathcal{A}, P, \mathcal{AP})$ and a DRA $\mathfrak{D} = (\mathcal{Q}_D, q_D^0, \Sigma, \mathcal{F}, \delta_D)$, we define the product MDP (PMDP) $\mathfrak{P} = \mathfrak{M} \times \mathfrak{D}$ as $\mathfrak{P} = (\mathcal{S}, s_0, \mathcal{A}_\mathfrak{P}, P_\mathfrak{P}, \mathcal{F}_\mathfrak{P})$, where (i) $\mathcal{S} = \mathcal{X} \times \mathcal{Q}_D$ is the set of states, so that $s = (x, q_D) \in \mathcal{S}$, $x \in \mathcal{X}$, and $q_D \in \mathcal{Q}_D$; (ii) $s_0 = (x_0, q_D^0)$ is the initial state; (iii) $\mathcal{A}_\mathfrak{P}$ is the set of actions inherited from the MDP, so that $\mathcal{A}_\mathfrak{P}(s) = \mathcal{A}(x)$, where $s = (x, q_D)$; (iv) $P_\mathfrak{P} : \mathcal{S} \times \mathcal{A}_\mathfrak{P} \times \mathcal{S} : [0, 1]$ is the transition probability function, so that $P_\mathfrak{P}(s, a_P, s') = P(x, a, x')$, where $s = (x, q_D) \in \mathcal{S}$, $s' = (x', q_D') \in \mathcal{S}$, $a_P \in \mathcal{A}(s)$ and $q_D' = \delta(q, L(x))$; (v) $\mathcal{F}_\mathfrak{P} = \{\mathcal{F}_i^\mathfrak{P}\}_{i=1}^f$ is the set of accepting states, where $\mathcal{F}_i^\mathfrak{P}$ is a set defined as $\mathcal{F}_i^\mathfrak{P} = \mathcal{X} \times \mathcal{F}_i$ and $\mathcal{F}_i = (\mathcal{G}_i, \mathcal{B}_i)$. □

Given any policy $\boldsymbol{\mu} : \mathcal{S} \rightarrow \mathcal{A}_\mathfrak{P}$ for $\mathfrak{P}$, we define an infinite run $\rho_\mathfrak{P}^{\boldsymbol{\mu}}$ of $\mathfrak{P}$ to be an infinite sequence of states of $\mathfrak{P}$, i.e., $\rho_\mathfrak{P}^{\boldsymbol{\mu}} = s_0 s_1 s_2 \ldots$, where $P_\mathfrak{P}(s_t, \boldsymbol{\mu}(s_t), s_{t+1}) > 0$. By definition of the accepting condition of the DRA $\mathfrak{D}$, an infinite run $\rho_\mathfrak{P}^{\boldsymbol{\mu}}$ is accepting if there exists at least one pair $i \in \{1, \ldots, f\}$ such that $\texttt{Inf}(\rho_\mathfrak{P}^{\boldsymbol{\mu}}) \cap \mathcal{G}_i^\mathfrak{P} \neq \emptyset$, and $\texttt{Inf}(\rho_\mathfrak{P}^{\boldsymbol{\mu}}) \cap \mathcal{B}_i^\mathfrak{P} = \emptyset$.

E. Problem Statement

Our goal is to compute a policy for the PMDP that maximizes the satisfaction probability $\mathbb{P}(\boldsymbol{\mu} \models \phi \mid s_0)$ of an LTL-encoded task ϕ. A formal definition of this probability can be found in [3], [48], [49]. To this end, we first adopt existing reward functions $R : \mathcal{S} \times \mathcal{A}_\mathfrak{P} \times \mathcal{S} \rightarrow \mathbb{R}$ defined based on the accepting condition of the PMDP as e.g., in [6]. Then, our goal is to compute a policy $\boldsymbol{\mu}^*$ that maximizes the expected accumulated return, i.e., $\boldsymbol{\mu}^*(s) = \arg\max_{\boldsymbol{\mu} \in \mathcal{D}} U^{\boldsymbol{\mu}}(s)$, where $\mathcal{D}$ is the set of all stationary deterministic policies over $\mathcal{S}$, and

$$U^{\boldsymbol{\mu}}(s) = \mathbb{E}^{\boldsymbol{\mu}}[\sum_{t=0}^{\infty} \gamma^t \; R(s_t, \boldsymbol{\mu}(s_t), s_{t+1}) | s = s_0]. \tag{1}$$

In (1), $\mathbb{E}^{\boldsymbol{\mu}}[\cdot]$ denotes the expected value given that the PMDP follows the policy $\boldsymbol{\mu}$ [50], $0 \leq \gamma < 1$ is the discount factor, and $s_0, ..., s_t$ is the sequence of states generated by $\boldsymbol{\mu}$ up to time t, initialized at s_0. Since the PMDP has a finite state/action space and $\gamma < 1$, there exists a stationary deterministic optimal policy $\boldsymbol{\mu}^*$ [50]. The reward function R and the discount factor γ should be designed so that maximization of (1) is equivalent to maximization of the satisfaction probability. Efforts towards this direction are presented e.g., in [6], [8] while provably correct rewards and discount factors for PMDPs constructed using LDBA, instead of DRA, are proposed in [9], [14], [16]. However, as discussed in Section I, due to sparsity of these rewards, these methods are sample-inefficient. This is the main challenge that this paper aims to address.

Problem 1: Given (i) an MDP $\mathfrak{M}$ with unknown transition probabilities and underlying graph structure; (ii) a task specification captured by an LTL formula ϕ; (iii) a reward function R for the PMDP motivating satisfaction of its accepting condition, develop a sample-efficient RL algorithm that can learn a deterministic control policy $\boldsymbol{\mu}^*$ that maximizes (1).

III. Accelerated Reinforcement Learning for Temporal Logic Control

To solve Problem 1, we propose a new reinforcement learning (RL) algorithm that can quickly synthesize control policies that maximize (1). The proposed algorithm is summarized in Algorithm 1 and described in detail in the following

subsections. First, in Section III-A, we define a distance function over the DRA state-space. In Sections III-B–III-C, we describe the proposed logically-guided RL algorithm for LTL control objectives. To accelerate the learning phase, the distance function defined in Section III-A is utilized to guide exploration. A discussion on how the proposed algorithm can be applied to LDBA, that typically have a smaller state space than DRA, is provided in Appendix A.

A. Distance Function over the DRA State Space

First, the LTL task ϕ is converted into a DRA; see Definition 2.4 [line 2, Alg. 1]. Then, we define a distance-like function over the DRA state-space that measures how 'far' the robot is from accomplishing the assigned LTL tasks [line 3, Alg. 1]. In other words, this function returns how far any given DRA state is from the sets of accepting states $\mathcal{G}_i$. To define this function, first, we remove from the DRA all infeasible transitions that cannot be physically enabled. To define infeasible transitions, we first define feasible symbols as follows [33]; see Fig. 1.

Definition 3.1 (Feasible symbols $\sigma \in \Sigma$): A symbol $\sigma \in \Sigma$ is *feasible* if and only if $\sigma \not\models b^{\text{inf}}$, where b^{inf} is a Boolean formula defined as $b^{\text{inf}} = \vee_{\forall x_i \in \mathcal{X}}(\vee_{\forall x_e \in \mathcal{X}\setminus\{x_i\}}(\pi^{x_i} \wedge \pi^{x_e}))$, where b^{inf} requires the robot to be present in more than one MDP state simultaneously. All feasible symbols σ are collected in a set denoted by $\Sigma_{\text{feas}} \subseteq \Sigma$. □

Then, we prune the DRA by removing infeasible DRA transitions defined as follows:

Definition 3.2 (Feasibility of DRA transitions): A DRA transition from q_D to q'_D is feasible if there exists at least one feasible symbol $\sigma \in \Sigma_{\text{feas}}$ such that $\delta(q_D, \sigma) = q'_D$; otherwise, it is infeasible. □

Next, we define a function $d : \mathcal{Q}_D \times \mathcal{Q}_D \to \mathbb{N}$ that returns the minimum number of *feasible* DRA transitions required to reach a state $q'_D \in \mathcal{Q}_D$ starting from a state $q_D \in \mathcal{Q}_D$. Particularly, we define this function as follows [33], [35]:

$$d(q_D, q'_D) = \begin{cases} |SP_{q_D,q'_D}|, \text{if } SP_{q_D,q'_D} \text{ exists,} \\ \infty, \qquad\quad \text{otherwise,} \end{cases} \tag{2}$$

where SP_{q_D,q'_D} denotes the shortest path (in terms of hops) in the pruned DRA from q_D to q'_D and $|SP_{q_D,q'_D}|$ stands for its cost (number of hops). Note that if $d(q_D^0, q_D) = \infty$, for all $q_D \in \mathcal{G}_i$ and for all $i \in \{1, \ldots, n\}$, then the LTL formula can not be satisfied since the in the pruning process, only the DRA transitions that are impossible to enable are removed. Next, using (2), we define the following distance function:[1]

$$d_F(q_D, \mathcal{F}) = \min_{q_D^G \in \cup_{i\in\{1,\ldots,f\}}\mathcal{G}_i} d(q_D, q_D^G). \tag{3}$$

In words, (3) measures the distance from any DRA state q_D to the set of accepting pairs, i.e., the distance to the closest DRA state q_D^G that belongs to $\cup_{i\in\{1,\ldots,f\}}\mathcal{G}_i$; see also Fig. 1.

B. Learning Optimal Temporal Logic Control Policies

In this section, we present the proposed accelerated RL algorithm for LTL control synthesis [lines 4-20, Alg. 1]. The output of the proposed algorithm is a *stationary deterministic* policy $\boldsymbol{\mu}^*$ for $\mathfrak{P}$ maximizing (1). To construct $\boldsymbol{\mu}^*$, we employ episodic Q-learning (QL). Similar to standard QL, starting from an initial PMDP state, we define learning episodes over which the robot picks actions as per a stationary and stochastic control policy $\boldsymbol{\mu} : \mathcal{S} \times \mathcal{A}_{\mathfrak{P}} \to [0, 1]$ that eventually converges to $\boldsymbol{\mu}^*$ [lines 4-5, Alg. 1]. Each episode terminates after a user-specified number of time steps τ or if the robot reaches a deadlock PMDP state, i.e., a state with no outgoing transitions [lines 7-20, Alg. 1]. Notice that the hyper-parameter τ should be selected to be large enough to ensure that the agent learns how to repetitively visit the accepting states [8], [9], [13]. The RL algorithm terminates once an action value function $Q^{\boldsymbol{\mu}}(s, a)$ has converged. This action value function is defined as the expected return for taking action a when at state s and then following policy $\boldsymbol{\mu}$ [52], i.e.,

$$Q^{\boldsymbol{\mu}}(s, a) = \mathbb{E}^{\boldsymbol{\mu}}[\sum_{t=0}^{\infty} \gamma^t \; R(s_t, \boldsymbol{\mu}(s_t), s_{t+1}) | s_0 = s, a_0 = a]. \tag{4}$$

We have that $U^{\boldsymbol{\mu}}(s) = \max_{a\in\mathcal{A}_{\mathfrak{P}}(s)} Q^{\boldsymbol{\mu}}(s, a)$ [52]. The action-value function $Q^{\boldsymbol{\mu}}(s, a)$ can be initialized arbitrarily.

During any learning episode the following process is repeated until the episode terminates. First, given the PMDP state s_t at the current time step t, initialized as $s_t = s_0$ [line 9, Alg. 1], an action a_t is selected as per a policy $\boldsymbol{\mu}$ [line 11, Alg. 1]; the detailed definition of $\boldsymbol{\mu}$ will be given later. The selected action is executed yielding the next state $s_{t+1} = (x_{t+1}, q_{t+1})$, and a reward $R(s_t, a_t, s_{t+1})$. For instance, the reward function

Algorithm 1 Accelerated RL for LTL Control Objectives

1: Initialize: (i) $Q^{\boldsymbol{\mu}}(s, a)$ arbitrarily, (ii) $\hat{P}(x, a, x') = 0$, (iii) $c(x, a, x') = 0$, (iv) $n(x, a) = 0$, for all $x, x' \in \mathcal{X}$ and $a \in \mathcal{A}(x)$, and (v) $n_{\mathfrak{P}}(s, a, s') = 0$ for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}_{\mathfrak{P}}(s)$;
2: Convert ϕ to a DRA $\mathcal{D}$;
3: Construct distance function d_F over the DRA as per (3);
4: $\boldsymbol{\mu} = (\epsilon, \delta) - \text{greedy}(Q)$;
5: `episode-number` $= 0$;
6: **while** Q has not converged **do**
7: `episode-number` $=$ `episode-number` $+ 1$;
8: Initialize time step $t = 0$;
9: Initialize $s_t = (x_0, q_D^0)$ for a randomly selected x_0;
10: **while** $t < \tau$ **do**
11: Pick action a_t as per (8);
12: Execute a_t and observe $s_{t+1} = (x_{t+1}, q_{t+1})$, and $R(s_t, a_t, s_{t+1})$;
13: $n(x_t, a_t) = n(x_t, a_t) + 1$;
14: $c(x_t, a_t, x_{t+1}) = c(x_t, a_t, x_{t+1}) + 1$;
15: Update $\hat{P}(x_t, a_t, x_{t+1})$ as per (6);
16: $n_{\mathfrak{P}}(s_t, a_t) = n_{\mathfrak{P}}(s_t, a_t) + 1$;
17: Update $Q^{\boldsymbol{\mu}}(s_t, a_t)$ as per (7);
18: $s_t = s_{\text{next}}$;
19: $t = t + 1$;
20: Update $\epsilon, \delta_b, \delta_e$;
21: **end while**
22: **end while**

[1]Observe that, unlike [36], [51], $d_F(q_D, \mathcal{F})$ may not be equal to 0 even if $q_D \in \mathcal{G}_i$. The latter may happen if q_D does not have a feasible self-loop.

R can be constructed as in [6] defined as follows:

$$R(s, a_{\mathfrak{P}}, s') = \begin{cases} r_{\mathcal{G}}, & \text{if } s' \in \mathcal{G}_i^{\mathfrak{P}}, \\ r_{\mathcal{B}}, & \text{if } s' \in \mathcal{B}_i^{\mathfrak{P}}, \\ r_d, & \text{if } s' \text{ is a deadlock state}, \\ r_0, & \text{otherwise}, \end{cases} \tag{5}$$

In (5), we have that $r_{\mathcal{G}} > 0$, for all $i \in \{1, \ldots, f\}$, and $r_d < r_{\mathcal{B}} < r_0 \leq 0$. This reward function motivates the robot to satisfy the PMDP accepting condition, i.e., to visit the states $\mathcal{G}_j^{\mathfrak{P}}$ as often as possible and minimize the number of times it visits $\mathcal{B}_i^{\mathfrak{P}}$ and deadlock states while following the shortest possible path; deadlock states are visited when the LTL task is violated, e.g., when collision with an obstacle occurs.

Given the new state s_{t+1}, the MDP model of the robot is updated. In particular, every time an MDP transition is enabled, the corresponding transition probability is updated. Let $\hat{P}(x_t, a_t, x_{t+1})$ denote the estimated MDP transition probability from state $x_t \in \mathcal{X}$ to state $x_{t+1} \in \mathcal{X}$, when an action a is taken. These estimated MDP transition probabilities are initialized so that $\hat{P}(x, a, x') = 0$, for all combinations of states and actions, and they are continuously updated at every time step t of each episode as [lines 13-15]:

$$\hat{P}(x_t, a_t, x_{t+1}) = \frac{c(x_t, a_t, x_{t+1})}{n(x_t, a_t)}, \tag{6}$$

where (i) $n : \mathcal{X} \times \mathcal{A} \to \mathbb{N}$ is a function that returns the number of times action a has been taken at an MDP state x and (ii) $c : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{N}$ is a function that returns the number of times an MDP state x' has been visited after taking action a at a state x. Note that as $n(x, a) \to \infty$ the estimated transition probabilities $\hat{P}(x, a, x')$ converge asymptotically to the true transition probabilities $P(x, a, x')$, for all transitions.

Next, the action value function is updated as follows [52] [line 17, Alg. 1] :

$$\begin{aligned} &Q^{\boldsymbol{\mu}}(s_t, a_t) = Q^{\boldsymbol{\mu}}(s_t, a_t) + (1/n_{\mathfrak{P}}(s_t, a_t)) \\ &[R(s_t, a_t) - Q^{\boldsymbol{\mu}}(s_t, a_t) + \gamma \max_{a'} Q^{\boldsymbol{\mu}}(s_{t+1}, a'))], \end{aligned} \tag{7}$$

where $n_{\mathfrak{P}} : \mathcal{S} \times \mathcal{A}_{\mathfrak{P}} \to \mathbb{N}$ counts the number of times that action a has been taken at the PMDP state s. Once the action value function is updated, the current PMDP state is updated as $s_t = s_{t+1}$, the time step t is increased by one, and the policy $\boldsymbol{\mu}$ gets updated [lines 18-20, Alg. 1].

As a policy $\boldsymbol{\mu}$, we propose an extension of the ϵ-greedy policy, called (ϵ, δ)-greedy policy, that selects an action a at an PMDP state s by using the learned action-value function $Q^{\boldsymbol{\mu}}(s, a)$ and the continuously learned transition probabilities $\hat{P}(x, a, x')$. Formally, the (ϵ, δ)-greedy policy $\boldsymbol{\mu}$ is defined as

$$\boldsymbol{\mu}(s, a) = \begin{cases} 1 - \epsilon + \frac{\delta_e}{|\mathcal{A}_{\mathfrak{P}}(s)|} & \text{if } a = a^* \text{ and } a \neq a_b, \\ 1 - \epsilon + \frac{\delta_e}{|\mathcal{A}_{\mathfrak{P}}(s)|} + \delta_b & \text{if } a = a^* \text{ and } a = a_b, \\ \delta_e / |\mathcal{A}_{\mathfrak{P}}(s)| & \text{if } a \in \mathcal{A}_{\mathfrak{P}}(s) \setminus \{a^*, a_b\}, \\ \delta_b + \delta_e / |\mathcal{A}_{\mathfrak{P}}(s)| & \text{if } a = a_b \text{ and } a \neq a^*, \end{cases} \tag{8}$$

where $\delta_b, \delta_e \in [0, 1]$ and $\epsilon = \delta_b + \delta_e \in [0, 1]$. In words, according to this policy, (i) with probability $1 - \epsilon$, the *greedy* action $a^* = \text{argmax}_{a \in \mathcal{A}_{\mathfrak{P}}} Q(s, a)$ is taken (as in the standard ϵ-greedy policy); and (ii) an exploratory action is selected with probability $\epsilon = \delta_b + \delta_e$. The exploration strategy is defined as follows: (ii.1) with probability δ_e a random action a is selected (*random* exploration); and (ii.2) with probability δ_b the action, denoted by a_b, that is most likely to drive the robot towards an accepting product state in $\mathcal{G}_i^{\mathfrak{P}}$ is taken (*biased* exploration). The action a_b will be defined formally in Section III-C. As in standard QL, ϵ should asymptotically converge to 0 while ensuring that eventually all actions have been applied infinitely often at all states. This ensures that $\boldsymbol{\mu}$ asymptotically converges to the optimal greedy policy

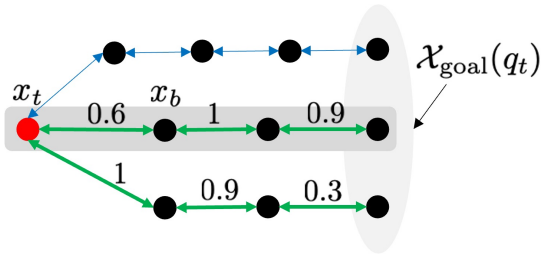

Fig. 2. Graphical depiction of the sets $\mathcal{X}_{\text{goal}}(q_t)$. The disks represent MDP states and the arrows between states mean that there exists at least one action such that the transition probability from one state to another one is non-zero. The length of the shortest path from x_t to $\mathcal{X}_{\text{goal}}$ is 3 hops, i.e., $J_{x_t, \mathcal{X}_{\text{goal}}} = 3$; see (12). Also, the paths p_j^t, $j \in \mathcal{J} = \{1, 2\}$ are highlighted with thick green lines. The numbers on top of the green edges represent $\max_a P(p_j^t(e), a, p_j^t(e+1))$; see (14). Observe that p^* is the green path highlighted with gray color.

$$\boldsymbol{\mu}^* = \underset{a \in \mathcal{A}_{\mathfrak{P}}}{\text{argmax}} Q^*(s, a) \tag{9}$$

where Q^* is the optimal action value function; see Sec. IV-A. We note that $Q^{\boldsymbol{\mu}^*}(s, \boldsymbol{\mu}^*(s)) = U^{\boldsymbol{\mu}^*}(s) = V^*(s)$, where $V^*(s)$ is the optimal value function that could have been computed if the MDP was fully known [52], [53]. Given ϵ, selection of the parameters δ_e and δ_b is discussed in Sec. IV.

C. Specification-guided Exploration for Accelerated Learning

Next, we describe the design of the biased action a_b in (8). First, we need to introduce the following definitions; see Fig. 2. Let $s_t = (x_t, q_t)$ denote the current PMDP state at the current learning episode and time step t of Algorithm 1. Let $\mathcal{Q}_{\text{goal}}(q_t) \subset \mathcal{Q}$ be a set that collects all DRA states that are one-hop reachable from q_t in the pruned DRA and they are closer to the accepting DRA states than q_t is, as per (3). Formally, $\mathcal{Q}_{\text{goal}}(q_t)$ is defined as follows:

$$\begin{aligned} \mathcal{Q}_{\text{goal}}(q_t) = \{q' \in \mathcal{Q} \mid (\exists \sigma \in \Sigma_{\text{feas}} \text{ such that} \\ \delta_D(q_t, \sigma) = q') \wedge (d_F(q', \mathcal{F}) = d_F(q_t, \mathcal{F}) - 1)\}. \end{aligned} \tag{10}$$

Also, let $\mathcal{X}_{\text{goal}}(q_t) \subseteq \mathcal{X}$ be a set of MDP states, denoted by x_{goal}, that if the robot eventually reaches, then transition from s_t to a product state $s_{\text{goal}} = [x_{\text{goal}}, q_{\text{goal}}]$ will occur, where $q_{\text{goal}} \in \mathcal{Q}_{\text{goal}}(q_t)$; see also Ex. 3.6. Formally, $\mathcal{X}_{\text{goal}}(q_t)$ is defined as follows:

$$\mathcal{X}_{\text{goal}}(q_t) = \{x \in \mathcal{X} \mid \delta_D(q_t, L(x)) \in \mathcal{Q}_{\text{goal}}(q_t)\}. \tag{11}$$

Next, we view the continuously learned MDP as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ where the set $\mathcal{V}$ is the set of MDP states, $\mathcal{E}$ is the set of edges, and $w : \mathcal{E} \to \mathbb{R}_+$ is function assigning weights to each edge. Specifically, an edge from the node (MDP state) x to x' exists if there exists at least one action $a \in \mathcal{A}(x)$ such that $\hat{P}(x, a, x') > 0$. Hereafter, we

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3484290

assign a weight equal to 1 to each edge; see also Remarks 3.4-3.5. We denote the cost of the shortest path from x to x' by $J_{x,x'}$. Next, we define the cost of the shortest path connecting a state x to the set $\mathcal{X}_{\text{goal}}$ as follows:

$$J_{x,\mathcal{X}_{\text{goal}}} = \min_{x' \in \mathcal{X}_{\text{goal}}} J_{x,x'}. \tag{12}$$

Let J be the total number of paths from x to $\mathcal{X}_{\text{goal}}$, where their length (i.e., number of hops) is $J_{x,\mathcal{X}_{\text{goal}}}$. We denote such a path by p_j^t, $j \in \mathcal{J} := \{1, \ldots, J\}$, and the e-th MDP state in this path by $p_j^t(e)$. Then, among all the paths p_j^t, we compute the one with the minimum uncertainty-based cost $C(p_j^t)$; see Fig. 2. We define this cost as

$$C(p_j^t) = \prod_{e=1}^{J_{x,\mathcal{X}_{\text{goal}}}} \left[\max_a \hat{P}(p_j^t(e), a, p_j^t(e+1))\right], \tag{13}$$

where the maximization is over all actions $a \in \mathcal{A}(p_j^t(e))$. We denote by p^* the path with the minimum cost $C(p_j^t)$, i.e., $p^* = p_{j^*}^t$, where $j^* = \text{argmax}_j\, C(p_j^t)$. Thus, we have that:

$$C(p^*) \geq C(p_j^t), \forall j \in \mathcal{J}. \tag{14}$$

Once p^* is constructed, the action a_b is defined as follows:

$$a_b = \underset{a \in \mathcal{A}(x_t)}{\text{argmax}}\ \hat{P}(x_t, a, x_b), \tag{15}$$

where $x_b = p^*(2)$; see Fig. 2. In words, a_b is the action with the highest probability of allowing the system to reach the state $p^*(2)$, i.e., to move along the best path p^*. Observe that computation of the biased action does not depend on the employed reward structure nor on perfectly learning all MDP transition probabilities.

Remark 3.3 (Initialization): Selection of the biased action a_b requires knowledge of (i) the MDP states x in (11) that need to be visited to enable transitions to DRA states in $\mathcal{Q}_{\text{goal}}$; and (ii) the underlying MDP graph structure, determined by the (unknown) transition probabilities, to compute (12). However, neither of them may be available in early episodes. In this case, we randomly initialize $\mathcal{X}_{\text{goal}}$ for (i). Similarly, for (ii), the estimated transition probabilities are randomly initialized (or, simply, set equal to 0 [line 1, Alg. 1]) initializing this way the MDP graph structure. If no paths can be computed to determine $J_{x_t,\mathcal{X}_{\text{goal}}}$ in (12), we select a random biased action.

Remark 3.4 (Computing Shortest Path): It is possible that the shortest path from x_t to $x_{\text{goal}} \in \mathcal{X}_{\text{goal}}(q_t)$ goes through states/nodes x that if visited, a transition to a new state $q \neq q_t$ that does not belong to $\mathcal{Q}_{\text{goal}}(q_t)$ may be enabled. Therefore, when we compute the shortest paths, we treat all such nodes x as 'obstacles' that should not be crossed. These states are collected in the set $\mathcal{X}_{\text{avoid}}$ defined as $\mathcal{X}_{\text{avoid}} = \{x \in \mathcal{X} \mid \delta(q_t, L(x)) = q_D \notin \mathcal{Q}_{\text{goal}}\}$.

Remark 3.5 (Weights & Shortest Paths): To design the biased action a_b, the MDP is viewed as a weighted graph where a weight $w = 1$ is assigned to all edges. In Section IV, this definition of weights allows us to show how the probability of making progress towards satisfying the assigned task (i.e., reaching the DRA states $\mathcal{Q}_{\text{goal}}$) within the minimum number of time steps (i.e., $J_{x_t,\mathcal{X}_{\text{goal}}}$ time steps) is positively affected by introducing bias in the exploration phase. Alternative weight assignments can be used that may further improve sample-efficiency in practice; see also Ex. 3.6. For instance, the assigned weights can be equal to the reciprocal of the estimated transition probabilities. In this case, the shortest path between two MDP states models the path with the least uncertainty that connects these two states. However, in this case the theoretical results shown in Section IV do not hold.

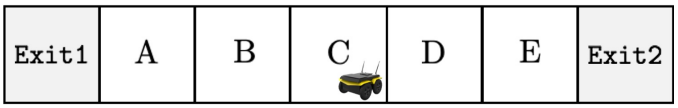

Fig. 3. MDP-based representation of the interaction of a ground robot with corridor-like environment. The square cells represent MDP states, i.e., $\mathcal{X} = \{\text{Exit1}, \text{Exit2}, \text{A}, \text{B}, \text{C}, \text{D}, \text{E}\}$. An action enabling transition between adjacent cells with non-zero probability exists for all MDP states.

Example 3.6 (Biased Exploration): Consider a robot operating in a corridor of a building as in Figure 3. The robot is tasked with exiting the building i.e., eventually reaching one of the two exits. This can be captured by the following LTL formula: $\phi = \Diamond(\pi^{\text{Exit1}} \vee \pi^{\text{Exit2}})$. The DRA of this specification is illustrated in Figure 1. Assume that $q_t = q_D^0$. Then, $\mathcal{X}_{\text{goal}} = \{\text{Exit1}, \text{Exit2}\}$. The robot can take two actions at each state (besides the 'exit' states): a_1 = 'left' and a_2 = 'right'. **(i)** Assume that $x_t = C$. Observe that $J_{x_t,\mathcal{X}_{\text{goal}}} = 3$ and that $J = 2$. Specifically, the following two paths p_j^t can be defined: p_1^t = C,D,E,Exit1 and p_2^t = C,B,A,Exit2. Consider also transition probabilities that satisfy $\max_a P(C, a, D) = 0.51$, $\max_a P(D, a, E) = 0.9$, $\max_a P(E, a, \text{Exit2}) = 1$, $\max_a P(C, a, B) = 0.9$, $\max_a P(B, a, A) = 0.6$, $\max_a P(A, a, \text{Exit1}) = 0.6$. In this case, we have that $C(p_1^t) = 0.459$ and $C(p_2^t) = 0.324$. According to (14), we have that $j^* = 1$ and, therefore, $x_b = p_1^t(2) = \text{D}$. The biased action a_b at x_t is $a_b = a_2$ as per (15). **(ii)** Assume that $x_t = D$. Then, we have that $J_{x_t,\mathcal{X}_{\text{goal}}} = 2$. Notice that there is only path to reach $\mathcal{X}_{\text{goal}}$ within $J_{x_t,\mathcal{X}_{\text{goal}}} = 2$ hops/time steps defined as p_1^t = D,E,Exit1. Consider also transition probabilities that satisfy $\max_a P(D, a, E) = 0.7$, $\max_a P(E, a, \text{Exit2}) = 0.7$, $\max_a P(D, a, C) = 1$, $\max_a P(C, a, B) = 1$, $\max_a P(B, a, A) = 1$, $\max_a P(A, a, \text{Exit1}) = 1$. In this case, we have that $C(p_1^t) = 0.49$. The biased action a_b at x_t is selected as follows. Assume $P(D, a_1, E) = 0.3$ and $P(D, a_2, E) = 0.7$. Given that $x_b = p_1^t(2) = \text{E}$, we have that $a_b = a_2$ as per (15). Observe that although there is a 'deterministic' path from x_t to Exit1 of length 4 that can be followed with probability 1, the biased action aims to drive the robot towards Exit2. This happens because the proposed algorithm is biased towards the shortest paths (of length 2 here), in terms of number of MDP transitions/hops, that will lead to DRA states that are closer to the accepting states by definition of the weights w. We note that the paths stemming from the biased action are not necessarily the paths with the least uncertainty; see also Rem. 3.5. Also, we highlight that we do not claim any optimality of a_b with respect to the task satisfaction probability; intuitively, in (ii), the biased action is 'sub-optimal' with respect to the task satisfaction probability.

IV. Algorithm Analysis

In this section, we show that any $(\epsilon, \delta)-$greedy policy achieves policy improvement; see Proposition 4.1. Also, we

provide conditions that δ_b and δ_e should satisfy under which the proposed biased exploration strategy results in learning control policies faster, in a probabilistic sense, than policies that rely on uniform-based exploration. We emphasize that these results should be interpreted primarily in an existential way as they rely on the unknown MDP transition probabilities. First, we provide 'myopic' sample-efficiency guarantees. Specifically, we show that starting from $s_t = (x_t, q_t)$, the probability of reaching PMDP states $s_{t+1} = (x_{t+1}, q_{t+1}))$, where x_{t+1} is closer to $\mathcal{X}_{\text{goal}}$ (see (11)) than x_t, is higher when bias is introduced in the exploration phase; see Section IV-B. Then, we provide non-myopic guarantees that ensure that starting from s_t the probability of reaching PMDP states $s_{t'} = (x_{t'}, q_{t'})$, where $t' > t$ and $q_{t'} \in \mathcal{Q}_{\text{goal}}$ (see (10)), in the minimum number of time steps (as determined by $J_{x_t,\mathcal{X}_{\text{goal}}}$) is higher when bias is introduced in the exploration phase; see Section IV-C.

A. Policy Improvement

Proposition 4.1 (Policy Improvement): For any (ϵ, δ)-greedy policy $\boldsymbol{\mu}$, the updated (ϵ, δ)-greedy policy $\boldsymbol{\mu'}$ obtained after updating the state-action value function $Q^{\boldsymbol{\mu}}(s, a)$ satisfies $U^{\boldsymbol{\mu'}}(s) \geq U^{\boldsymbol{\mu}}(s)$, for all $s \in \mathcal{S}$. □

Proof: To show this result, we follow the same steps as in the policy improvement result for the ϵ-greedy policy [52]. For simplicity of notation, hereafter we use $A = |\mathcal{A}_{\mathfrak{P}}(s)|$. Thus, we have that: $U^{\boldsymbol{\mu'}}(s) = \sum_{a\in\mathcal{A}_{\mathfrak{P}}(s)} \boldsymbol{\mu'}(s,a) Q^{\boldsymbol{\mu}}(s,a) = \frac{\delta_e}{A} \sum Q^{\boldsymbol{\mu}}(s,a) + (1-\epsilon) \max_{a\in\mathcal{A}_{\mathfrak{P}}(s)} Q^{\boldsymbol{\mu}}(s,a) + \delta_b Q^{\boldsymbol{\mu}}(s,a_b) \geq \frac{\delta_e}{A} \sum Q^{\boldsymbol{\mu}}(s,a) + (1-\epsilon) \sum_{a\in\mathcal{A}_{\mathfrak{P}}(s)} \left(\frac{\boldsymbol{\mu}(s,a) - \frac{\delta_e}{A} - \mathbb{I}_{a=a_b}\delta_b}{1-\epsilon} \right) Q^{\boldsymbol{\mu}}(s,a) + \delta_b Q^{\boldsymbol{\mu}}(s,a_b) = \sum_{a\in\mathcal{A}_{\mathfrak{P}}(s)} \boldsymbol{\mu}(s,a) Q^{\boldsymbol{\mu}}(s,a) = U^{\boldsymbol{\mu}}(s)$ where the inequality holds because the summation is a weighted average with non-negative weights summing to 1, and as such it must be less than the largest number averaged. ■

In Proposition 4.1, the equality $U^{\boldsymbol{\mu'}}(s) = U^{\boldsymbol{\mu}}(s)$, $\forall s \in \mathcal{S}$, holds if $\boldsymbol{\mu} = \boldsymbol{\mu'} = \boldsymbol{\mu}^*$, where $\boldsymbol{\mu}^*$ is the optimal policy [52].

B. Myopic Effect of Biased Exploration

In this section, we demonstrate the myopic benefit of the biased exploration; the proofs can be found in Appendix B. To formally describe it we introduce first the following definitions. Let $s_t = (x_t, q_t)$ be the PMDP state at the current time step t of an RL episode of Algorithm 1. Also, let $\mathcal{R}(x_t) \subseteq \mathcal{X}$ denote a set collecting all MDP states that can be reached within one hop from x_t, i.e.,

$$\mathcal{R}(x_t) = \{x \in \mathcal{X} \mid \exists a \in \mathcal{A}(x) \text{ such that } \hat{P}(x_t, a, x) > 0\}.^2 \tag{16}$$

Then, we can define the set $\mathcal{X}_{\text{closer}}$ that collects all MDP states that are one hop reachable from x_t and they are closer to $\mathcal{X}_{\text{goal}}(x_t)$ than x_t is, i.e.,

$$\mathcal{X}_{\text{closer}}(x_t) = \{x \in \mathcal{R}(x_t) \mid J_{x,\mathcal{X}_{\text{goal}}} = J_{x_t,\mathcal{X}_{\text{goal}}} - 1)\}. \tag{17}$$

The following result shows that the probability of $x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)$ is higher when biased exploration is employed.

[2]The reachable set in (16) is a subset of the actual set of one-hop neighbors of x_t since (16) uses the estimated transition probabilities (6).

Proposition 4.2: Let $s_t = (x_t, q_t)$ be the PMDP state at the current time step t of an RL episode of Algorithm 1. Let also $x_b \in \mathcal{X}_{\text{closer}}(x_t)$ denote the MDP state towards which the action a_b is biased. If $\delta_b > 0$ and (18) holds,

$$P(x_t, a_b, x) \geq \max_{\bar{x}\in\mathcal{X}_{\text{closer}}(x_t)} \sum_a \frac{P(x_t, a, \bar{x})}{|\mathcal{A}(x_t)|}, \forall x \in \mathcal{X}_{\text{closer}}(x_t), \tag{18}$$

where the summation is over $a \in \mathcal{A}(x_t)$, then we have that

$$\mathbb{P}_b(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)) \geq \mathbb{P}_g(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)). \tag{19}$$

In (19), $\mathbb{P}_g(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t))$ and $\mathbb{P}_b(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t))$ denote the probability of reaching any state $x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)$ starting from x_t without and with bias introduced in the exploration phase, respectively. □

Next, we provide a 'weaker' result which, however, does not require the strong requirement of (18). The following result shows that the probability that the next state x_{t+1} will be equal to $x_b \in \mathcal{X}_{\text{closer}}$ (as opposed to any state in $\mathcal{X}_{\text{closer}}$ in Prop. 4.2) is greater when bias is introduced in the exploration phase.

Proposition 4.3: Let $s_t = (x_t, q_t)$ be the PMDP state at the current time step t of an RL episode of Algorithm 1. Let also $x_b \in \mathcal{X}_{\text{closer}}(x_t)$ denote the MDP state towards which the action a_b is biased. If $\delta_b > 0$, then

$$\mathbb{P}_b(x_{t+1} = x_b) \geq \mathbb{P}_g(x_{t+1} = x_b), \tag{20}$$

where $\mathbb{P}_g(x_{t+1} = x_b)$ and $\mathbb{P}_b(x_{t+1} = x_b)$ denote the probability of reaching at $t+1$ the state x_b starting from x_t without and with bias introduced in the exploration phase, respectively.

C. Non-Myopic Effect of Biased Exploration

In this section, we demonstrate the non-myopic effect of the biased exploration; the proofs can be found in Appendix C. To present our main results, we need to introduce the following definitions. Let $s_t = (x_t, q_t)$ be the current PMDP state. Also, let $t^* = J_{x_t,\mathcal{X}_{\text{goal}}}$ denote the length (i.e., the number of hops/MDP transitions) of the paths p_j^t. Recall that all paths p_j^t, $j \in \mathcal{J}$, share the same length, in terms of number of hops, by construction. Second, we define a function $\beta : \mathcal{J} \to [0, 1]$ that maps every path p_j^t, $j \in \mathcal{J}$, into $[0, 1]$ as follows:

$$\beta(p_j^t) = \prod_{m=0}^{t^*-1} \{P(x_{t+m}, a_b, x_{t+m+1})\delta_b + P(x_{t+m}, a^*, x_{t+m+1})(1-\epsilon) + \frac{\delta_e}{|\mathcal{A}(x_{t+m})|}\}. \tag{21}$$

In (21), we have that $x_{t+m} = p_j^t(m+1)$, for all $m \in \{0, \ldots, t^*-1\}$ and a_b is the biased action computed at state $s_{t+m} = (x_{t+m}, q_t)$ as discussed in Section III-C, i.e., using the path $p_{j^*}^{t+m}$.

Proposition 4.4 (Most Likely Path): At time step t of the current RL episode, let (i) $s_t = (x_t, q_t)$ be the current PMDP state; and (ii) $p_{j^*}^t$ be the path used to design the biased action at the time step t. Let R_j be a (Bernouli) random variable that is true if after t^* time steps (i.e., at the time step $t + t^*$), a

path p_j^t has been generated, for some $j \in \mathcal{J}$. If there exists δ_b and δ_e satisfying the following condition

$$\beta(p_{j^*}^t) \geq \max_{j \in \mathcal{J}} \beta(p_j^t), \tag{22}$$

then, we have that $\mathbb{P}_b(R_{j^*} = 1) \geq \max_{j \in \mathcal{J}} \mathbb{P}_b(R_j = 1)$, where $\mathbb{P}_b(R_{j^*} = 1)$ and $\mathbb{P}_b(R_j = 1)$ stand for the probability that $R_{j^*} = 1$ and $R_j = 1$, respectively, if the MDP evolves as per the proposed (ϵ, δ)-greedy policy. □

Remark 4.5 (Prop. 4.4): Prop. 4.4 implies that there exists δ_b and δ_e such that among all paths p_j^t, $j \in \mathcal{J}$, designed at time t, the most likely path that the MDP will generate over the next t^* time steps is $p_{j^*}^t$. For instance, if $\delta_b = 1$, and, therefore, $\epsilon = \delta_e = 0$, then we get that (22) is equivalent to $\prod_{m=0}^{t^*-1} P(x_{t+m}, a_b, x_{t+m+1}) \geq \max_{j \in \mathcal{J}} \prod_{m=0}^{t^*-1} P(\bar{x}_{t+m}, \bar{a}_b, \bar{x}_{t+m+1})$, due to (14)-(15), where $x_{t+m} = p_{j^*}^t(m+1)$, $\bar{x}_{t+m} = p_j^t(m+1)$ for all $m \in \{0, \ldots, t^* - 1\}$, and a_b and $\bar{a}_b$ denote the biased action at states x_{t+m} and $\bar{x}_{t+m}$ using the path $p_{j^*}^{t+m}$.

In what follows, we show that there exists δ_b and δ_e that ensure that the probability of generating the path $p_{j^*}^t$ under the (ϵ, δ)-greedy policy (captured by $\mathbb{P}_b(R_{j^*} = 1)$) is larger than the probability of generating any path p_j^t, $j \in \mathcal{J}$, under the ϵ-greedy policy. To make this comparative analysis meaningful, hereafter, we assume that probability of exploration $\epsilon = \delta_b + \delta_e$ is the same for both policies; thus, the probability of selecting the greedy action is the same for both policies, as well. Recall again that the ϵ-greedy policy can be recovered by removing bias from the (ϵ, δ)-greedy policy, i.e., by setting $\delta_b = 0$. To present this result, we need to define a function $\eta : \mathcal{J} \to [0, 1]$ mapping every path p_j^t, $j \in \mathcal{J}$, into $[0, 1]$ as follows:

$$\eta(p_j^t) = \prod_{m=0}^{t^*-1} \{P(x_{t+m}, a^*, x_{t+m+1})(1 - \epsilon) + \frac{\epsilon}{|\mathcal{A}(x_{t+m})|}\}. \tag{23}$$

In (23), we have that $x_{t+m} = p_j^t(m+1)$, for all $m \in \{0, \ldots, t^* - 1\}$ and a^* is the greedy action computed at state $s_{t+m} = (x_{t+m}, q_t)$.

Proposition 4.6 (Random vs Biased Exploration): At time step t of the current RL episode, let (i) $s_t = (x_t, q_t)$ be the current product state; and (ii) $p_{j^*}^t$ be the path used to design the current biased action. Let R_j be a (Bernouli) random variable that is true if after t^* time steps (i.e., at the time step $t + t^*$), a path p_j^t has been generated for some $j \in \mathcal{J}$ under a policy $\boldsymbol{\mu}$. If there exists δ_b and δ_e satisfying the following condition

$$\beta(p_{j^*}^t) \geq \max_{j \in \mathcal{J}} \eta(p_j^t) \tag{24}$$

then, we have that $\mathbb{P}_b(R_{j^*} = 1) \geq \max_{j \in \mathcal{J}} \mathbb{P}_g(R_j = 1)$, where $\mathbb{P}_b(R_{j^*} = 1)$ and $\mathbb{P}_g(R_j = 1)$ stand for the probability that $R_{j^*} = 1$ and $R_j = 1$, if the MDP evolves as per the proposed (ϵ, δ)-greedy and ϵ-greedy policy, respectively. □

Remark 4.7 (Prop. 4.6): Prop. 4.6 states that among all paths p_j^t of length t^*, $j \in \mathcal{J}$, there exists values for δ_b and δ_e under which there exists an MDP path (the one with index j^*) that is more likely to be generated over the next t^* time steps under the (ϵ, δ)-greedy than any path p_j^t, $j \in \mathcal{J}$ that can be generated under the ϵ-greedy policy. For instance, if $\delta_b = 1$ and $\delta_e = 0$, (i.e., $\epsilon = 1$) then (24) is equivalent to $\prod_{m=0}^{t^*-1} P(x_{t+m}, a_b, x_{t+m+1}) \geq \max_{j \in \mathcal{J}} \prod_{m=0}^{t^*-1} \frac{1}{|\mathcal{A}(\bar{x}_{t+m})|}$, where $x_{t+m} = p_{j^*}^t(m+1)$, and $\bar{x}_{t+m} = p_j^t(m+1)$ for all $m \in \{0, \ldots, t^* - 1\}$. Let $A_{\min} = \min_{x \in \mathcal{X}} |\mathcal{A}(x)|$. Then, for $\delta_b = 1$, the result in Proposition 4.6 holds if $\prod_{m=0}^{t^*-1} P(x_{t+m}, a_b, x_{t+m+1}) \geq (\frac{1}{A_{\min}})^{t^*}$. The latter is true if e.g., $P(x_{t+m}, a_b, x_{t+m+1}) \geq \frac{1}{A_{\min}}$ for all $m \in \{0, \ldots, t^* - 1\}$. We note that a similar result is presented in [33] which employs a similar biased exploration to address deterministic temporal logic planning problems (see Remark 4.5 in [33]).

Proposition 4.6 compares the sample-efficiency of (ϵ, δ)-greedy and ϵ-greedy policies with respect to a specific path $p_{j^*}^t$. In the following result, building upon Proposition 4.6, we provide a more general result. Specifically, we show that the probability that after t^*+1 time steps a PMDP state $s = (x, q)$, where $q \in \mathcal{Q}_{\text{goal}}$ (see (10)), will be reached is higher when bias is introduced in the exploration phase. We emphasize again that given the current PMDP state $s_t = (x_t, q_t)$ in an RL episode, the earliest that a PMDP state $s = (x, q)$, where $q \in \mathcal{Q}_{\text{goal}}$ can be reached is after $t^* + 1$ where $t^* = J_{x_t, \mathcal{X}_{\text{goal}}}$ iterations. The reason is that the length of the shortest path from x_t to states $\mathcal{X}_{\text{goal}}$ that can enable the transition from q_t to $\mathcal{Q}_{\text{goal}}$ is $t^* = J_{x_t, \mathcal{X}_{\text{goal}}}$.

Proposition 4.8 (Sample Efficiency): Let $s_t = (x_t, q_t)$ be the product state reached at the t-th time step of the current RL episode. A state $s_{\text{goal}} = (x, q_{\text{goal}})$, where $q_{\text{goal}} \in \mathcal{Q}_{\text{goal}}$ can be reached after at least t^*+1 time steps, where $t^* = J_{x_t, \mathcal{X}_{\text{goal}}}$. If there exist δ_b and δ_e satisfying the following condition:

$$\sum_{j \in \mathcal{J}} \beta(p_j^t) \geq \sum_{j \in \mathcal{J}} \eta(p_j^t), \tag{25}$$

where j^* stands for the index to the path selected as per (14), then $\mathbb{P}_b(q_{t+t^*+1} \in \mathcal{Q}_{\text{goal}}) \geq \mathbb{P}_g(q_{t+t^*+1} \in \mathcal{Q}_{\text{goal}})$, where $\mathbb{P}_b(q_{t+t^*+1} \in \mathcal{Q}_{\text{goal}})$ and $\mathbb{P}_g(q_{t+t^*+1} \in \mathcal{Q}_{\text{goal}})$ stand for the probability that a PMDP state with a DRA state in $\mathcal{Q}_{\text{goal}}$ will be reached after exactly $t^* + 1$ time steps using the (ϵ, δ)-greedy and ϵ-greedy policy, respectively. □

Remark 4.9 (Selecting parameters δ_b and δ_e): **(i)** The result in Proposition 4.8 shows that there exist δ_b and δ_e to potentially improve sample efficiency compared to uniform/random exploration. However, selection of δ_b and δ_e as per Proposition 4.8 requires knowledge of the actual MDP transition probabilities along all paths p_j^t, $j \in \mathcal{J}$ which are not available. To address this, the estimated transition probabilities, computed in (6), can be used instead. To mitigate the fact that the initial estimated probabilities may be rather inaccurate, δ_e can be selected so that $\delta_e > \delta_b$ for the first few episodes. Intuitively, this allows to initially perform random exploration to learn an accurate enough MDP transition probabilities across all directions. Once this happens and given ϵ, values for δ_b and δ_e that satisfy the requirement (25) (using the estimated probabilities) can be computed by applying a simple line search algorithm over all possible values for $\delta_b \in \{0, \epsilon\}$, since $\delta_e + \delta_b = \epsilon$. **(ii)** A more efficient approach would be to pick δ_b based on Proposition 4.6 instead of 4.8. The reason is that searching for δ_b that satisfies (24) requires less computations than (25); see also Remark 4.7. **(iii)** An even more computationally efficient, but heuristic, approach to pick

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3484290

δ_b and δ_e is the following. We select δ_b and δ_e so that $\delta_e > \delta_b$ for the first few episodes to learn an accurate enough MDP model and then allow $\delta_e < \delta_b$ to prioritize exploration towards directions that may contribute to mission progress while letting both δ_b and δ_b to asymptotically converge to 0. Nevertheless, the values for δ_b and δ_e selected in this way may not satisfy the requirements mentioned in Propositions 4.4, 4.6, and 4.8.

Remark 4.10 (Limitations): Alternative definitions of d_F may affect the construction of the set $\mathcal{Q}_{\text{goal}}$ in (10). Currently, d_F captures the shortest path, in terms of number of hops, between a DRA state and the set of accepting states. However, this definition neglects the underlying MDP structure which may compromise sample-efficiency. Specifically, the shortest DRA-based path may be harder for the MDP to realize than a longer DRA-based path, depending on the MDP transition probabilities. The result presented in Proposition 4.8 shows that given a distance function d_F and, consequently, $\mathcal{Q}_{\text{goal}}$, there exist conditions that the parameters δ_b and δ_e should satisfy, so that the probability of reaching $\mathcal{Q}_{\text{goal}}$ within the minimum possible number of time steps (i.e., $J_{x,\mathcal{X}_{\text{goal}}}$ time steps) is larger when the (ϵ, δ)-greedy policy is used. This does not necessarily imply that the probability of eventually reaching accepting states is also larger as this depends on the definition of d_F and, consequently, $\mathcal{Q}_{\text{goal}}$. Designing d_F that optimizes sample-efficiency is a future research direction. However, our comparative experiments in Section V demonstrate sample-efficiency of the proposed method under various settings.

V. Numerical Experiments

To demonstrate the sample-efficiency of our method, we provide extensive comparisons against existing model-free and model-based RL algorithms. All methods have been implemented on Python 3.8 and evaluated on a computer with an Nvidia RTX 3080 GPU, 12th Gen Intel(R) Core(TM) i7-12700K CPU, and 8GB RAM.

A. Setting up Experiments & Baselines

MDP: We consider environments represented as 10×10, 20×20, and 50×50 discrete grid worlds, resulting in MDPs with $|\mathcal{X}| = 100, 400$, and $2,500$ states denoted by $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$, respectively. The robot has nine actions:'left', 'right', 'up', 'down', 'idle' as well as the corresponding four diagonal actions. At any MDP state x, excluding the boundary ones, the set of actions $\mathcal{A}(x)$ that the robot can apply includes eight of these nine actions that are randomly selected while ensuring that the idle action is available at any state. The set of actions at boundary MDP states exclude those ones that drive the robot outside the environment. The transition probabilities are designed so that given any action, besides 'idle', the probability of reaching the intended state is randomly selected from the interval $[0.7, 0.8]$ while the probability of reaching neighboring MDP states is randomly selected as long as the summation of transition probabilities over the next states x' is equal to 1, for a fixed action a and starting state x. The action 'idle' is applied deterministically.

Baselines: In the following case studies we demonstrate the performance of Algorithm 1 when it is equipped with the proposed (ϵ, δ)-greedy policy (8), the ϵ-greedy policy, the Boltzman policy, and the UCB1 policy. Notice that Alg. 1 is model-free when it is equipped with these baselines as it does not require learning the MDP. We also compare it against a standard model-based approach that explicitly computes and stores the product MDP (PMDP) [3]. Computing the PMDP requires learning the underlying MDP model which can be achieved e.g., by simply letting the agent randomly explore the environment and then estimating the transition probabilities as in (6).[3] In our implementation, we directly use the ground-truth MDP transition probabilities giving an 'unfair' advantage to the model-based approach over the proposed one. Given the resulting PMDP, we apply dynamic programming to compute the optimal policy and its satisfaction probability [3].

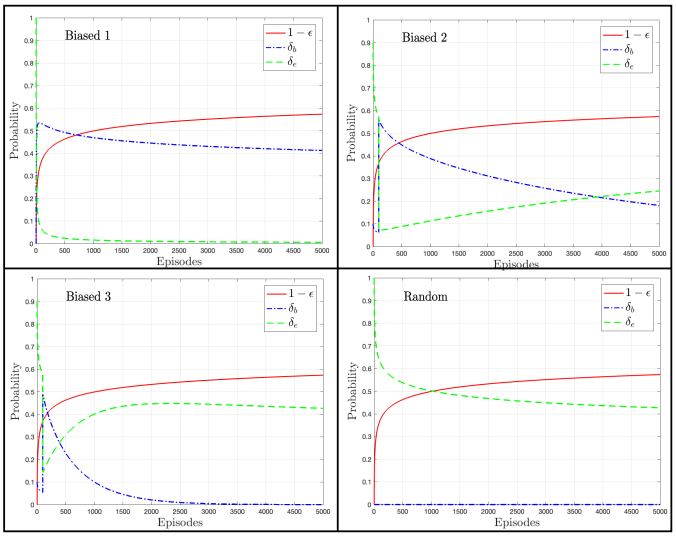

Fig. 4. Decay rates of the parameters δ_e, δ_b, and ϵ considered in Section V for $\mathfrak{M}_1$ and $\mathfrak{M}_2$. The rate at which $1-\epsilon$ (red) increases is the same in all figures. As the number of episodes goes to infinity, $1-\epsilon$ converges to 1 and both δ_b and δ_e converge to 0. Notice that, in the bottom right figure, δ_b is always equal to 0 resulting in random exploration (ϵ-greedy policy).

To examine sensitivity of the proposed algorithm with respect to the parameters δ_e and δ_b, we have considered three different decay rates for δ_e and δ_b, as per (iii) in Remark 4.9. Hereafter, we refer to the corresponding exploration strategies as 'Biased 1', 'Biased 2', and 'Biased 3', and 'Random', where the latter corresponds to the ϵ-greedy policy. The rate at which δ_b decreases over time gets smaller as we proceed from 'Biased 1', 'Biased 2', 'Biased 3', to 'Random'. In other words, 'Biased 1' incurs the most 'aggressive' bias in the exploration phase. The evolution of these parameters for the MDPs $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is illustrated in Fig. 4. Similar biased strategies were selected for $\mathfrak{M}_3$. The only difference is that δ_b is designed so that it converges to 0 slower due to the larger size of the state space. The corresponding mathematical formulas are provided in Appendix D. To make the comparison between the (ϵ, δ)- and the ϵ-greedy policy fair, we select the same ϵ for both. The Boltzmann control policy is defined as follows: $\boldsymbol{\mu}_B(s) = \frac{e^{Q^{\boldsymbol{\mu}_B}(s,a)/T}}{\sum_{a' \in \mathcal{A}_{\mathfrak{B}}} e^{Q^{\boldsymbol{\mu}_B}(s,a')/T}}$, where $T \geq 0$ is the temperature parameter. The UCB1 control policy is defined as: $\boldsymbol{\mu}_U(s) = \text{argmax}_{a \in \mathcal{A}_{\mathfrak{B}}} \left[Q^{\boldsymbol{\mu}_U}(s,a) + C \times \sqrt{\frac{2\log(N(s))}{n(s,a)}} \right]$, where (i) $N(s)$ and $n(s,a)$ denote the number of times state

[3]This would result in learning transition probabilities of $\mathfrak{M}_1$ and $\mathfrak{M}_2$ in 1.1 and 90 minutes, respectively, with maximum error equal to 0.05.

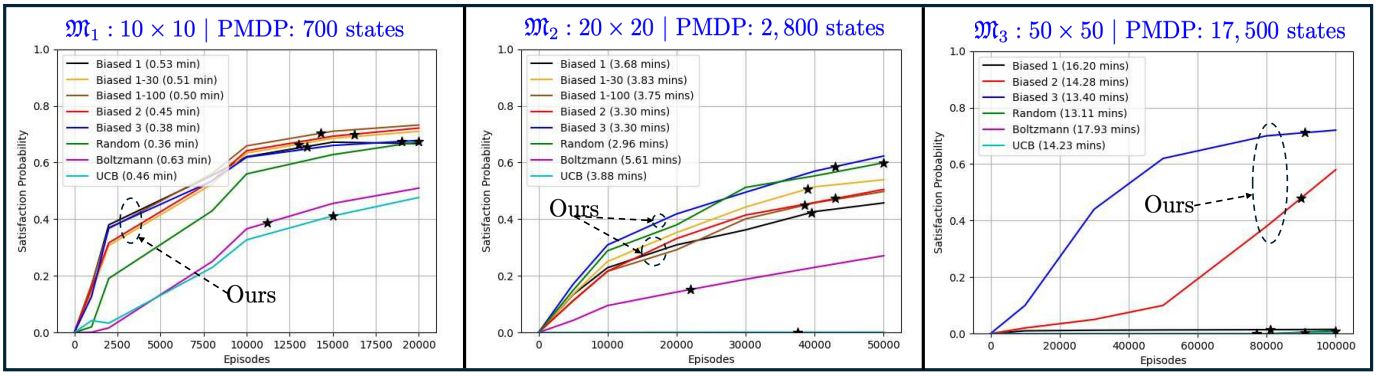

Fig. 5. A Simple Coverage Task (Section V-B): Comparison of average satisfaction probability $\bar{\mathbb{P}}$ when Algorithm 1 is applied with the proposed (ϵ, δ)-greedy policy, ϵ-greedy policy, Boltzmann policy, and UCB1 policy over the MDPs $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$. Biased $1-30$ and Biased $1-100$ refer to the cases where the Biased 1 exploration method is applied under the constraint that the MDP transition probabilities are updated only during the first 30 and 100 episodes, respectively. The legend also includes the total runtime per method. The black stars on top of each reward curve denote the training episode where the corresponding policy is when the fastest policy has finished training over the total number of episodes.

s has been visited and the number of times action a has been selected at state s and (ii) C is an exploration parameter. This control policy is biased towards the least explored directions. In each case study, we pick values for C and T from a fixed set that yield the best performance. In all case studies, we adopt the reward function in (5) with $\gamma = 0.99$ and $r_{\mathcal{G}} = 10$, $r_{\mathcal{B}} = -0.1$, $r_d = -100$, and $r_o = 0$. To convert the LTL formulas into DRA, we have used the ltl2dstar toolbox [54].

Performance Metrics: We utilize the satisfaction probabilities of the policies learned at various stages during training to assess performance of our algorithm and the baselines. Specifically, given a learned/fixed policy $\boldsymbol{\mu}$ and an initial PMDP state $s = (x, q_D^0)$, we compute the probability $\mathbb{P}(\boldsymbol{\mu} \models \phi | s = (x, q_D^0))$ using dynamic programming. We compute this probability for all $x \in \mathcal{X}$ and then we compute the average satisfaction probability $\bar{\mathbb{P}} = [\sum_{\forall x \in \mathcal{X}} \mathbb{P}(\boldsymbol{\mu} \models \phi | s = (x, q_D^0))]/|\mathcal{X}|$. We report the average $\bar{\mathbb{P}}$ over five runs; see Figs. 5-8. The satisfaction probabilities are computed using the unknown-to-the-agent MDP transition probabilities. Since runtimes for a training episode may differ across methods, we also report runtime metrics; see Figs. 5-8. Specifically, we document the runtimes required for all methods to complete a a predetermined maximum number of episodes, as well as the training episode each method reaches when the fastest one completes the training process. This allows us to compare satisfaction probabilities over the policies more fairly based on fixed runtimes rather than a fixed number of training episodes.

Summary of Comparisons: Our experiments show that the proposed (ϵ, δ)-greedy policy outperforms the model-free baselines, learning policies with higher satisfaction probabilities over the same timeframe. This performance gap widens significantly as the size of the PMDP increases. Specifically, our method begins learning policies with non-zero satisfaction probabilities within the first few hundred training episodes. The baselines can catch up relatively quickly, narrowing the performance gap, typically after a few thousand episodes, but only in small PMDPs (fewer than $10,000$ states). In larger PMDPs (more than $10,000$ states), our method significantly outperforms the model-free baselines. Additionally, the proposed (ϵ, δ)-greedy policy and the ϵ-greedy policy have similar runtimes, while they tend to be faster than UCB and, especially, Boltzmann. The model-based approach, on the other hand, demonstrates faster computation of the optimal policy compared to model-free baselines, including ours, when applied to small PMDPs (e.g., with fewer than $5,000$ states). However, this approach is memory inefficient, requiring storage of the PMDP and the action value function $Q^{\boldsymbol{\mu}}$. As a result, it failed to handle case studies with large PMDPs (more than $15,000$ states). In contrast, our method was able to handle PMDPs with hundreds of thousands of states; see e.g., Section V-E.

Remark 5.1 (Limitations & Implementation Improvements): A limitation of our method compared to model-free baselines is that it requires learning an MDP model, which can become memory-inefficient over large-scale MDPs. However, we believe that this limitation can be mitigated by more efficient implementations of our approach. For instance, in our current implementation [40], we store all learned MDP transition probabilities used to compute the biased action. However, the selection of the biased action does not require learning all transition probabilities; see (15). Instead, it only requires learning which action is most likely to drive the system from a state x to a neighboring state x'. Once this property is learned for a pair of states x and x', the estimated transition probabilities $\hat{P}(x, a, x')$ in (6) can be discarded.

B. Case Study I: A Simple Coverage Task

First, we consider a coverage/sequencing mission requiring the agent to eventually reach the states 99, and 46 or 90 while avoiding 99 until 33 is reached, and always avoiding the obstacle states 73, 24, 15, and 88. This task is captured by the following LTL formula $\phi = (\Diamond \pi^{99}) \wedge \Diamond(\pi^{46} \vee \pi^{90}) \wedge (\Diamond \pi^{33}) \wedge (\neg \pi^{99} \mathcal{U} \pi^{33}) \wedge \Box \neg \pi^{\text{obs}}$, where π^{obs} is satisfied when the robot visits one of the obstacle states. This formula corresponds to a DRA with 7 states and 1 accepting pair. Thus, the PMDP constructed using $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$ has 700, $2,800$, and $17,500$ states, respectively.

The comparative results are shown in Fig. 5. Our algorithm achieves the best performance when applied to $\mathfrak{M}_1$, regardless of the biased strategy. As for $\mathfrak{M}_2$, the best performance is achieved by our (ϵ, δ)-greedy policy coupled with 'Biased 3', followed closely by the ϵ-greedy policy, 'Biased 2', and 'Biased 1'. Notice that ϵ-greedy can catch up quickly when applied to $\mathfrak{M}_1$ and $\mathfrak{M}_2$ due to the relatively small size of the resulting PMDPs. This figure also shows the performance of

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3484290

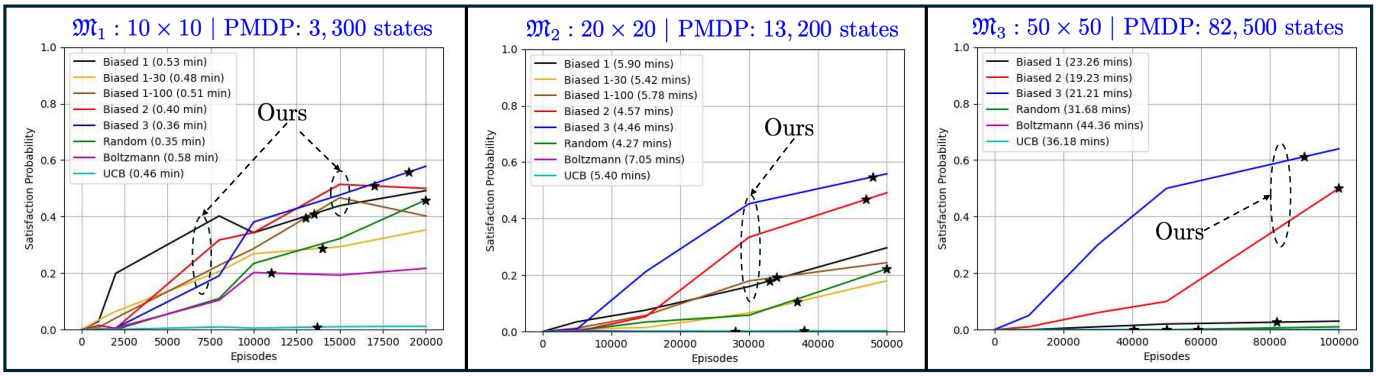

Fig. 6. A More Complex Coverage Task (Section V-C): Comparison of average accumulated reward (top row) and satisfaction probability $\bar{\mathbb{P}}$ (bottom row) when Algorithm 1 is applied with the proposed (ϵ, δ)-greedy policy, ϵ-greedy policy, Boltzmann policy, and UCB1 policy over the MDPs $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$. The legend also includes the total runtime per method. The black stars on top of each reward curve denote the training episode where the corresponding policy is when the fastest policy has finished training over the total number of episodes.

'Biased 1' when the MDP transition probabilities are updated only for the first 30 and 100 episodes for $\mathfrak{M}_1$ and $\mathfrak{M}_2$. The performance of our approach is not significantly affected by this choice, demonstrating robustness against model inaccuracies. This occurs because our algorithm does not require learning the ground truth values of the transition probabilities to compute the biased action; see (15).

The benefit of our method becomes more pronounced when $\mathfrak{M}_3$ is considered, resulting in a larger PMDP. In this case, the average satisfaction probability $\bar{\mathbb{P}}$ of the policies learned by all baselines is close to 0 after $100,000$ training episodes (approximately 15 minutes). In contrast, the proposed (ϵ, δ)-greedy policy, coupled with 'Biased 2' and 'Biased 3', learned policies with $\bar{\mathbb{P}} = 0.58$ and $\bar{\mathbb{P}} = 0.71$, respectively, within the same timeframe. Also, notice that 'Biased 1' failed to yield a satisfactory policy for $\mathfrak{M}_3$ within the same timeframe; recall that 'Biased 1' for $\mathfrak{M}_3$ is more aggressive than 'Biased 1 for $\mathfrak{M}_1$ and $\mathfrak{M}_2$. We attribute this to the aggressive nature of this exploration strategy towards a desired high-reward path, which possibly does not allow the agent to sufficiently explore a significant portion of the PMDP state space, resulting in a low average satisfaction probability. This shows that increasing the amount of bias does not necessarily yield policies with higher satisfaction probabilities.

The model-based approach was able to compute the optimal policy for the MDPs $\mathfrak{M}_1$ and $\mathfrak{M}_2$, but failed for $\mathfrak{M}_3$ due to excessive memory requirements. Specifically, the optimal policy for $\mathfrak{M}_1$ was computed in 0.5 minutes, and for $\mathfrak{M}_2$, it took 5.98 minutes (without including the time to learn the MDP model). The corresponding optimal average satisfaction probabilities $\bar{\mathbb{P}}$ were 0.916 for $\mathfrak{M}_1$ and 0.911 for $\mathfrak{M}_2$. We noticed that the model-based approach tends to be faster than the model-free baselines, particularly for smaller PMDPs.

C. Case Study II: A More Complex Coverage Task

Second, we consider a more complex sequencing task compared to the one in Section V-B, which involves visiting a larger number of MDP states. The goal is to eventually reach the MDP states $x = 81, 95, 80, 88$, and 92 in any order, while always avoiding the states $x = 5, 15, 54, 32, 24, 66, 42, 70$, and 71 representing obstacles in the environment. This task can be formulated using the following LTL formula: $\phi = \Diamond\pi^{81} \wedge \Diamond\pi^{95} \wedge \Diamond\pi^{80} \wedge \Diamond\pi^{88} \wedge \Diamond\pi^{80} \wedge \Diamond\pi^{92} \wedge \Box\neg\pi^{\text{obs}}$, where π^{obs} is true if the robot visits any of the obstacle states. This formula corresponds to a DRA with 33 states and 1 accepting pair. Therefore, the PMDPs constructed using $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$ have $3,300$, $13,200$, and $82,500$ states, respectively, which are significantly larger than those of Section V-B.

Overall, our method, especially when coupled with 'Biased 2' and 'Biased 3', learns policies with higher satisfaction probabilities faster than the baselines; see Fig. 6. The benefit of our method is more pronounced as the PMDP size increases, as shown in the cases of $\mathfrak{M}_2$ and $\mathfrak{M}_3$. For example, when considering the MDP $\mathfrak{M}_3$, our method equipped with 'Biased 2' and 'Biased 3' learns policies with $\bar{\mathbb{P}} = 0.55$ and $\bar{\mathbb{P}} = 0.64$, respectively, while $\bar{\mathbb{P}} < 0.05$ for all other baselines, given the same amount of training time. Also, as in Section V-B, observe that 'Biased 1' failed to learn a satisfactory policy for $\mathfrak{M}_3$.

The model-based baseline computed the optimal policy $\boldsymbol{\mu}^*$ for the MDPs $\mathfrak{M}_1$ and $\mathfrak{M}_2$ in 1.1 and 112.15 minutes, respectively, while it failed to compute the optimal policy for $\mathfrak{M}_3$ due to excessive memory requirements. The average optimal satisfaction probability of the learned policies for $\mathfrak{M}_1$ and $\mathfrak{M}_2$ is 0.9854 and 0.9466, respectively.

D. Case Study III: Surveillance Task

Third, we consider a surveillance/recurrence mission captured by the following LTL formula: $\phi = \Box\Diamond\pi^{90} \wedge \Box\Diamond\pi^{70} \wedge \Box\Diamond(\pi^{80} \vee \pi^{63}) \wedge \Box\Diamond\pi^{88} \wedge (\neg\pi^{88}\mathcal{U}\pi^{90}) \wedge \Box\neg\pi^{\text{obs}}$. This formula requires the robot to (i) visit infinitely often and in any order the states 90, 70, 80 or 63 and 88; (ii) avoid reaching 88 until 80 is visited; and (iii) always avoid the obstacle in state 33. The corresponding DRA has 16 states and 1 accepting pair. Thus, the PMDP constructed using $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$ has $1,600$, $6,400$, and $40,000$ states, respectively.

The comparative performance results are shown in Figure 7. Observe that the (ϵ, δ)-greedy policy, especially when paired with 'Biased 2' and 'Biased 3', performs better that the model-free baselines in terms of sample-efficiency across all considered MDPs. For instance, in the case of $\mathfrak{M}_1$, our proposed algorithm learns policies with average satisfaction probabilities ranging from 0.75 to 0.9, depending on the biased exploration strategy, within $2,500$ training episodes. In contrast, the average satisfaction probability for the baselines is around 0.4 after the same number of episodes. As the number of episodes increases, the baselines manage to catch up due to

This article has been accepted for publication in IEEE Transactions on Automatic Control. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAC.2024.3484290

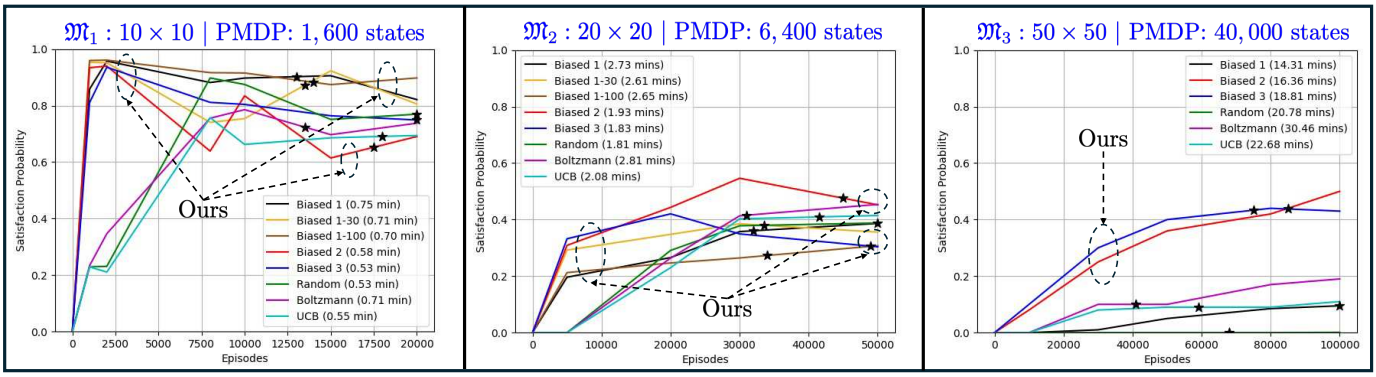

Fig. 7. Surveillance Task (Section V-D): Comparison of average satisfaction probability $\bar{\mathbb{P}}$ when Algorithm 1 is applied with the proposed (ϵ, δ)-greedy policy, ϵ-greedy policy, Boltzmann policy, and UCB1 policy over the MDPs $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$. The legend also includes the total runtime per method. The black stars on top of each reward curve denote the training episode where the corresponding policy is when the fastest policy has finished training over the total number of episodes.

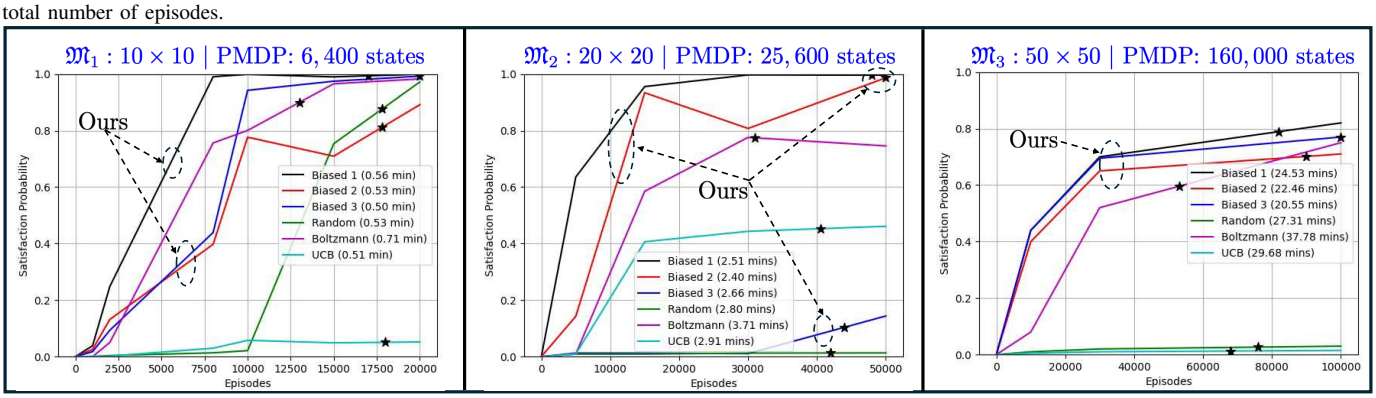

Fig. 8. Disjoint Surveillance Task (Section V-E): Comparison of average satisfaction probability $\bar{\mathbb{P}}$ when Algorithm 1 is applied with the proposed (ϵ, δ)-greedy policy, ϵ-greedy policy, Boltzmann policy, and UCB1 policy over the MDPs $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$. The legend includes the total runtime per method. The black stars on top of each reward curve denote the training episode where the corresponding policy is when the fastest policy has finished training over the total number of episodes.

the relatively small PMDP size. Similar trends are observed for $\mathfrak{M}_2$. As in the other case studies, the benefit of our method becomes more evident when considering $\mathfrak{M}_3$, which yields a significantly larger PMDP. In this scenario, our proposed algorithm, coupled with 'Biased 2' and 'Biased 3', learns a control policy with satisfaction probabilities of $\bar{\mathbb{P}} = 0.51$ and $\bar{\mathbb{P}} = 0.45$ within $100,000$ episodes (or approximately 20 minutes), respectively. In contrast, the baselines achieve satisfaction probabilities $\bar{\mathbb{P}} < 0.2$ within the same timeframe. As discussed in Section V-C, 'Biased 1' performs poorly in $\mathfrak{M}_3$, possibly due to its aggressive bias.

The model-based approach can compute the optimal policy only for the MDPs $\mathfrak{M}_1$ and $\mathfrak{M}_2$ while it failed in the case of $\mathfrak{M}_3$ due to excessive memory requirements. Regarding the MDP $\mathfrak{M}_1$ it computed an optimal policy corresponding to $\bar{\mathbb{P}} = 0.989$ within 2.31 minutes. As for the MDP $\mathfrak{M}_2$, it computed the optimal policy with $\bar{\mathbb{P}} = 0.981$ within 7.71 minutes.

E. Case Study IV: Disjoint Task

Finally, we consider a mission ϕ with two disjoint sub-tasks, i.e., $\phi = \phi_1 \vee \phi_2$ requiring the robot to accomplish either ϕ_1 or ϕ_2. The sub-tasks are defined as $\phi_1 = (\Diamond \pi^{99} \wedge \Diamond \pi^{45} \wedge \Diamond \pi^{32} \wedge \Box \neg \pi^{64})$ and $\phi_2 = (\Diamond \pi^{18} \wedge \Diamond \pi^{72} \wedge \Diamond \pi^{4})$. The LTL formula ϕ corresponds to a DRA with 64 states and 2 accepting pairs. As a result, the PMDP constructed using $\mathfrak{M}_1$, $\mathfrak{M}_2$, and $\mathfrak{M}_3$ has $6,400$, $25,600$, and $160,000$ states, respectively. This task requires the robot to eventually either visit the states 99, 45, and 32 while always avoiding 64 or visit the states 18, 72, and 4. Notice that the optimal satisfaction probability of ϕ_1 and ϕ_2 is 1 and less than 1, respectively.

The comparative results are reported in Figure 8. In $\mathfrak{M}_1$, our method coupled with 'Biased 1' achieves the best performance, closely followed by 'Biased 3'. Both biased exploration strategies result in a control policy satisfying ϕ with probability very close to 1 in approximately 0.5 minutes. Additionally, all other baselines, except UCB, perform satisfactorily, learning policies with $\mathbb{P} \in [0.8, 0.9]$ in the same time frame. The performance gap between our method and the baselines becomes more pronounced with the larger PMDPs constructed using $\mathfrak{M}_2$ and $\mathfrak{M}_3$. Specifically, in $\mathfrak{M}_2$, 'Biased 1' and 'Biased 2' achieve the best performance followed by 'Boltzmann', 'UCB', 'Biased 2', and 'ϵ-greedy'. In fact, 'Biased 1' and 'Biased 2' still manage to learn a policy with $\bar{\mathbb{P}}$ very close to 1 in 2.40 mins while for the other baselines it holds that $\bar{\mathbb{P}} < 0.8$. It is worth noting that the performance of 'Biased 3' has dropped significantly compared to $\mathfrak{M}_1$. This drop may be attributed to δ_b converging quite fast to 0 relative to the large size of the PMDP. In fact, once δ_b is almost equal to 0, then the (ϵ, δ)-greedy policy closely resembles the standard ϵ-greedy policy which in this case has also learned a policy with very low average satisfaction probability. Recall that $\mathfrak{M}_2$ shared exactly the same biased exploration strategies ('Biased 1', 'Biased 2', and 'Biased 3') across all case studies regardless of the PMDP size. However, the PMDP for $\mathfrak{M}_2$ is significantly larger than the ones considered in the other case studies which may explain the poor performance of 'Biased

3' compared the other $\mathfrak{M}_2$ case studies. Observe in $\mathfrak{M}_3$ that our method outperforms all baselines. Specifically, within 20.55 mins, the average satisfaction probability corresponding to 'Biased 1', 'Biased 2', 'Biased 3', and 'Boltzmann' is 0.8, 0.71, 0.78, and 0.6 respectively. The Boltzmann policy requires in total 37.78 minutes to eventually yield a policy with $\bar{\mathbb{P}} = 0.76$. Finally, the model-based approach was able to compute an optimal policy only for $\mathfrak{M}_1$ within 6.1 minutes with $\bar{\mathbb{P}} = 0.9772$; interestingly, model-free methods are faster in this case study.

VI. Conclusions

In this paper, we proposed a new accelerated reinforcement learning (RL) for temporal logic control objectives. The proposed RL method relies on new control policy, called (ϵ, δ)-greedy, that prioritizes exploration in the vicinity of task-related regions. This results in enhanced sample-efficiency as supported by theoretical results and comparative experiments. Our future work will focus on enhancing scalability by using function approximations (e.g., neural networks).

Appendix A
Extensions: Biased Exploration over LDBA

In this appendix, we show that the proposed exploration strategy can be extended to Limit Deterministic Büchi Automaton (LDBA) that typically have a smaller state space than DRA which can further accelerate learning [38]. First, any LTL formula can be converted in an LDBA defined as follows:

Definition A.1 (LDBA [38]): An LDBA is defined as $\mathfrak{A} = (\mathcal{Q}, q_0, \Sigma, \mathcal{F}, \delta)$ where $\mathcal{Q}$ is a finite set of states, $q_0 \in \mathcal{Q}$ is the initial state, $\Sigma = 2^{\mathcal{AP}}$ is a finite alphabet, $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_f\}$ is the set of accepting conditions where $\mathcal{F}_j \subset \mathcal{Q}$, $1 \leq j \leq f$, and $\delta : \mathcal{Q} \times \Sigma \to 2^{\mathcal{Q}}$ is a transition relation. The set of states $\mathcal{Q}$ can be partitioned into two disjoint sets $\mathcal{Q} = \mathcal{Q}_N \cup \mathcal{Q}_D$, so that (i) $\delta(q, \pi) \subset \mathcal{Q}_D$ and $|\delta(q, \pi)| = 1$, for every state $q \in \mathcal{Q}_D$ and $\pi \in \Sigma$; and (ii) for every $\mathcal{F}_j \in \mathcal{F}$, it holds that $\mathcal{F}_j \subset \mathcal{Q}_D$ and there are ε-transitions from $\mathcal{Q}_N$ to $\mathcal{Q}_D$. □

An infinite run ρ of $\mathfrak{A}$ over an infinite word $w = \sigma_0\sigma_1\sigma_2 \dots \in \Sigma^\omega$, $\sigma_t \in \Sigma = 2^{\mathcal{AP}}$ $\forall t \in \mathbb{N}$, is an infinite sequence of states $q_t \in \mathcal{Q}$, i.e., $\rho = q_0 q_1 \dots q_t \dots$, such that $q_{t+1} \in \delta(q_t, \sigma_t)$. The infinite run ρ is called *accepting* (and the respective word w is accepted by the LDBA) if $\texttt{Inf}(\rho) \cap \mathcal{F}_j \neq \emptyset, \forall j \in \{1, \dots, f\}$, where $\texttt{Inf}(\rho)$ is the set of states that are visited infinitely often by ρ. Also, an ε-transition allows the automaton to change its state without reading any specific input. In practice, the ε-transitions between $\mathcal{Q}_N$ and $\mathcal{Q}_D$ reflect the "guess" on reaching $\mathcal{Q}_D$: accordingly, if after an ε-transition the associated labels in the accepting LDBA set cannot be read, or if the accepting states cannot be visited, then the guess is deemed to be wrong, and the trace is disregarded and is not accepted by the automaton. However, if the trace is accepting, then the trace will stay in $\mathcal{Q}_D$ ever after, i.e. $\mathcal{Q}_D$ is invariant.

Given a (non-pruned) LDBA, we construct the product MDP (PMDP), similarly to Definition 2.6. The formal definition of this PMDP can be found in [8], [9]. To synthesize a policy that satisfies the LDBA accepting condition, we can adopt any reward function for the product MDP proposed in the literature [8], [9]. Once the LDBA is constructed, it is pruned exactly as discussed in Section III-A. The ϵ-transitions are not pruned. Given the resulting automaton, similar to (3), we define the distance to an accepting set of states $\mathcal{F}_j$ as $d_F(q, \mathcal{F}_j) = \min_{q_G \in \mathcal{F}_j} d(q, q_G)$ where $d(q, q_G)$ is defined as in (2). This function is used to bias exploration so that each set $\mathcal{F}_j$ is visited infinitely often. To design a biased exploration strategy that can account for the LDBA accepting condition, we first define the set $\mathcal{V}$ that collects indices j to the set of accepting states $\mathcal{F}_j$ that have not been visited during the current RL episode. Then, among all non-visited set of accepting states $\mathcal{F}_j$, we pick one randomly based on which we define the set $\mathcal{Q}_{\text{goal}}(q_t)$. Similar to (10), we define the set $\mathcal{Q}_{\text{goal}}(q_t)$ as: $\mathcal{Q}_{\text{goal}}(q_t) = \{q' \in \mathcal{Q} \mid (\exists \sigma \in \Sigma_{\text{feas}}$ such that $q' \in \delta(q_t, \sigma)) \wedge (d_F(q', \mathcal{F}_j) = d_F(q_t, \mathcal{F}_j) - 1)\}$, where $j \in \mathcal{V}$. Recall, that all ϵ- transitions in the LDBA are feasible. Thus, by definition, $\mathcal{Q}_{\text{goal}}(q_t)$ includes all states q where the transition from q_t to q is an ϵ-transition. Given $\mathcal{Q}_{\text{goal}}(q_t)$, the biased action is selected exactly as described in Section III-C. Once the set of states $\mathcal{F}_j$ is visited, the set $\mathcal{V}$ is updated as $\mathcal{V} = \mathcal{V} \setminus \{j\}$, and then the set $\mathcal{Q}_{\text{goal}}(q_t)$ is updated accordingly.

Appendix B
Proof for Results of Section IV-B

A. Proof Proposition 4.2

The probability of reaching any state $s_{t+1} = (x_{t+1}, q_{t+1})$ where $x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)$ under a stochastic policy $\boldsymbol{\mu}(s, a)$ is: $\sum_{x \in \mathcal{X}_{\text{closer}}} \sum_{a \in \mathcal{A}(x_t)} \boldsymbol{\mu}(s_t, a) P(x_t, a, x)$. Thus, we have that:[4]

$$\mathbb{P}_b(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)) - \mathbb{P}_g(x_{t+1} \in \mathcal{X}_{\text{closer}}(x_t)) = \sum_{x \in \mathcal{X}_{\text{closer}}(x_t)} \sum_{a \in \mathcal{A}(x_t)} P(x_t, a, x)(\boldsymbol{\mu}_b(s_t, a) - \boldsymbol{\mu}_g(s_t, a)), \quad (26)$$

where $\boldsymbol{\mu}_g$ and $\boldsymbol{\mu}_b$ refer to the ϵ-greedy (no biased exploration) and (ϵ, δ)-greedy policy (biased exploration), respectively. In what follows, we compute $\boldsymbol{\mu}_b(s_t, a) - \boldsymbol{\mu}_g(s_t, a)$, for all $a \in \mathcal{A}_{\mathfrak{P}}$. Recall, that $\boldsymbol{\mu}_b(s_t, a)$ is the probability of selecting the action a at state s_t. Also, hereafter, we assume that the greedy action a^* is different from the biased action a_b; however, the same logic applies if $a_b = a^*$, leading to the same result. For simplicity of notation, we use $A = |\mathcal{A}_{\mathfrak{P}}(s)|$.

First, for the action $a = a^*$, we have that (a) $\boldsymbol{\mu}_b(s_t, a^*) - \boldsymbol{\mu}_g(s_t, a^*) = (1 - \epsilon + \frac{\delta_\epsilon}{A}) - (1 - \epsilon + \frac{\epsilon}{A}) = (1 - \epsilon + \frac{\delta_\epsilon}{A}) - (1 - \epsilon + \frac{\delta_b + \delta_\epsilon}{A}) = \frac{\delta_b}{A}$. Similarly, for $a = a_b$, we have that (b) $\boldsymbol{\mu}_b(s_t, a_b) - \boldsymbol{\mu}_g(s_t, a_b) = (\delta_b + \frac{\delta_\epsilon}{A}) - \frac{\epsilon}{A} = -\frac{\delta_b(m-1)}{A}$. Also, for all other actions $a \neq a_b, a^*$, we have that (c) $\boldsymbol{\mu}_b(s_t, a) - \boldsymbol{\mu}_g(s_t, a) = \frac{\delta_\epsilon}{A} - \frac{\epsilon}{A} = -\frac{\delta_b}{A}$. Substituting the above equations (a)-(c) into (26) yields: $\mathbb{P}_b(x_{t+1} \in \mathcal{X}_{\text{closer}}) - \mathbb{P}_g(x_{t+1} \in \mathcal{X}_{\text{closer}}) = \delta_b \sum_{x \in \mathcal{X}_{\text{closer}}} (P(x_t, a_b, x) - \sum_{a \in \mathcal{A}(x_t)} \frac{P(x_t, a, x)}{A})$. Due to (18) and that $\delta_b > 0$, we conclude that $\mathbb{P}_b(s_{t+1} \in \mathcal{X}_{\text{closer}}) - \mathbb{P}_g(s_{t+1} \in \mathcal{X}_{\text{closer}}) \geq 0$ completing the proof.

B. Proof Of Proposition 4.3

The probability of reaching a state s_{t+1} where $x_{t+1} = x_b$ under a policy $\mu(s, a)$ is: $\sum_{a \in \mathcal{A}(x_t)} \mu(s_t, a) P(x_t, a, x_b)$. Thus, we have $\mathbb{P}_b(x_{t+1} = x_b) - \mathbb{P}_g(x_{t+1} = x_b) = \sum_{a \in \mathcal{A}(x_t)} P(x_t, a, x_b)(\mu_b(s_t, a) - \mu_g(s_t, a))$ Following the

[4]Note that q_{t+1} is selected deterministically, due to the DRA structure, i.e., $q_{t+1} = \delta_D(q_t, L(x_t))$.

same steps as in the proof of Proposition 4.2, we get that $\mathbb{P}_b(x_{t+1}=x_b)-\mathbb{P}_g(x_{t+1}=x_b)\geq 0$ if $\delta_b>0$, which holds by assumption, and $P(x_t,a_b,x_b)\geq \sum_{a\in\mathcal{A}(x_t)}\frac{P(x_t,a,x_b)}{|\mathcal{A}(x_t)|}$ which holds by definition of a_b in (15). Specifically, given x_t and x_b, we have that $P(x_t,a_b,x_b)\geq P(x_t,a,x_b)$ for all $a\in\mathcal{A}(x_t)$ due to (15). Thus, $P(x_t,a_b,x_b)$ must be greater than or equal to the average transition probability over the actions a i.e., $\sum_{a\in\mathcal{A}(x_t)}\frac{P(x_t,a,x_b)}{|\mathcal{A}(x_t)|}$ completing the proof.

APPENDIX C
PROOF OF RESULTS OF SECTION IV-C

A. Proof of Proposition 4.4

By definition of R_{j^*} and R_j, we can rewrite the inequality $\mathbb{P}_b(R_{j^*}=1)\geq \max_{j\in\mathcal{J}}\mathbb{P}_b(R_j=1)$ as

$$\prod_{m=0}^{t^*-1}\sum_{a\in\mathcal{A}(x_{t+m})}\mu_b(s_{t+m},a)P(x_{t+m},a,x_{t+m+1})\geq \max_{j\in\mathcal{J}}\prod_{m=0}^{t^*-1}\sum_{\bar{a}\in\mathcal{A}(\bar{x}_{t+m})}\mu_b(\bar{s}_{t+m},\bar{a})P(\bar{x}_{t+m},\bar{a},\bar{x}_{t+m+1}). \tag{27}$$

where $s_{t+m}=(x_{t+m},q_t)$, $x_{t+m}=p_{j^*}^t(m+1)$, $\bar{s}_{t+m}=(\bar{x}_{t+1},q_t)$, $\bar{x}_{t+m}=p_j^t(m+1)$, for all $m\in\{0,\dots,t^*-1\}$. Recall that by construction of the paths p_j^t, the DRA state will remain equal to q_t as the MDP agent moves along any of the paths p_j^t, for all $j\in\mathcal{J}$; see Remark 3.4. We will show that (27) holds by contradiction. Assume that there exists at least one path $p_{\bar{j}}^t$, $\bar{j}\in\mathcal{J}$, that does not satisfy (27), i.e.,

$$\prod_{m=0}^{t^*-1}\sum_{a\in\mathcal{A}(x_{t+m})}\mu_b(s_{t+m},a)P(x_{t+m},a,x_{t+m+1})< \prod_{m=0}^{t^*-1}\sum_{\bar{a}\in\mathcal{A}(\bar{x}_{t+m})}\mu_b(\bar{s}_{t+m},\bar{a})P(\bar{x}_{t+m},\bar{a},\bar{x}_{t+m+1}), \tag{28}$$

where $\bar{s}_{t+m}$ and $\bar{x}_{t+m}$ are defined as per $p_{\bar{j}}^t$.

Next, we assume that $a^*\neq a_b$ and $\bar{a}^*\neq\bar{a}_b$; the same logic applies even if this is not the case leading to the same result. Using (8), we plug the values of $\mu_b(s_{t+m},a)$ and $\mu_b(\bar{s}_{t+m},\bar{a})$ for all $a\in\mathcal{A}(x_{t+m})$ and $\bar{a}\in\mathcal{A}(\bar{x}_{t+m})$ in (28) which yields:

$$\begin{aligned}&\prod_{m=0}^{t^*-1}\{P(x_{t+m},a_b,x_{t+m+1})(\delta_b+\frac{\delta_e}{|\mathcal{A}(x_{t+m})|})+\\ &P(x_{t+m},a^*,x_{t+m+1})(1-\epsilon+\frac{\delta_e}{|\mathcal{A}(x_{t+m})|})+\\ &\sum_{a\neq a^*,a_b}P(x_{t+m},a,x_{t+m+1})(\frac{\delta_e}{|\mathcal{A}(x_{t+m})|})\}<\\ &\prod_{m=0}^{t^*-1}\{P(\bar{x}_{t+m},\bar{a}_b,\bar{x}_{t+m+1})(\delta_b+\frac{\delta_e}{|\mathcal{A}(\bar{x}_{t+m})|})+\\ &P(\bar{x}_{t+m},\bar{a}^*,\bar{x}_{t+m+1})(1-\epsilon+\frac{\delta_e}{|\mathcal{A}(\bar{x}_{t+m})|})+\\ &\sum_{\bar{a}\neq\bar{a}^*,\bar{a}_b}P(\bar{x}_{t+m},\bar{a},\bar{x}_{t+m+1})(\frac{\delta_e}{|\mathcal{A}(\bar{x}_{t+m})|})\}.\end{aligned} \tag{29}$$

In (29), a_b and $\bar{a}_b$ stand for the biased action computed when the PMDP state is s_{t+m} and $\bar{s}_{t+m}$ (using the optimal path $p_{j^*}^{t+m}$, as per (14), as discussed in Section III-C). The same notation extends to all other actions. The purpose of this notation is only to emphasize that the biased and greedy actions at s_{t+m} and $\bar{s}_{t+m}$ are not necessarily the same. By rearranging the terms in (29), we get the following result

$$\begin{aligned}&\prod_{m=0}^{t^*-1}\{P(x_{t+m},a_b,x_{t+m+1})\delta_b+\\ &P(x_{t+m},a^*,x_{t+m+1})(1-\epsilon)+\frac{\delta_e}{|\mathcal{A}(x_{t+m})|}\}<\\ &\prod_{m=0}^{t^*-1}\{P(\bar{x}_{t+m},\bar{a}_b,\bar{x}_{t+m+1})\delta_b+\\ &P(\bar{x}_{t+m},\bar{a}^*,\bar{x}_{t+m+1})(1-\epsilon)+\frac{\delta_e}{|\mathcal{A}(\bar{x}_{t+m})|}\}.\end{aligned} \tag{30}$$

Due to (21), (30) can expressed as $\beta(p_{j^*}^t)<\beta(p_{\bar{j}}^t)$ which contradicts (22) completing the proof.[5]

B. Proof of Proposition 4.6

This proof follows the same steps as the proof of Proposition 4.4. The inequality $\mathbb{P}_b(R_{j^*}=1)\geq\max_{j\in\mathcal{J}}\mathbb{P}_g(R_j=1)$ can re-written as

$$\prod_{m=0}^{t^*-1}\left(\sum_{a\in\mathcal{A}(x_{t+m})}\mu_b(s_{t+m},a)P(x_{t+m},a,x_{t+m+1})\right)\geq \max_{j\in\mathcal{J}}\prod_{m=0}^{t^*-1}\left(\sum_{\bar{a}\in\mathcal{A}(\bar{x}_{t+m})}\mu_g(\bar{s}_{t+m},\bar{a})P(\bar{x}_{t+m},\bar{a},\bar{x}_{t+m+1})\right) \tag{31}$$

where $s_{t+m}=(x_{t+m},q_t)$, $x_{t+m}=p_{j^*}^t(m+1)$, $\bar{s}_{t+m}=(\bar{x}_{t+m},q_t)$, and $\bar{x}_{t+m}=p_j^t(m+1)$, for all $m\in\{1,\dots,t^*-1\}$. We will show this result by contradiction. Assume that there exists at least one path $p_{\bar{j}}^t$, $\bar{j}\in\mathcal{J}$, that does not satisfy (31), i.e.,

$$\prod_{m=0}^{t^*-1}\left(\sum_{a\in\mathcal{A}(x_{t+m})}\mu_b(s_{t+m},a)P(x_{t+m},a,x_{t+m+1})\right)< \prod_{m=0}^{t^*-1}\left(\sum_{\bar{a}\in\mathcal{A}(\bar{x}_{t+m})}\mu_g(\bar{s}_{t+m},\bar{a})P(\bar{x}_{t+m},a,\bar{x}_{t+m+1})\right), \tag{32}$$

where $\bar{s}_{t+m}$ and $\bar{x}_{t+m}$ are defined as per $p_{\bar{j}}^t$.

In what follows, we denote by a^* and a_b the greedy and the biased action as per μ_b, and $\bar{a}^*$ the greedy action as per μ_g. We assume that $a^*\neq a_b$; the same logic applies even if this is not the case leading to the same final result. We plug the values of $\mu_b(s_{t+m},a)$ and $\mu_g(\bar{s}_{t+m},\bar{a})$ for all $a\in\mathcal{A}(x_{t+m})$

[5]Notice that $\beta(p_j^t)$ is equal to the probability that, starting from x_t, the MDP path p_j^t, $j\in\mathcal{J}$, will be generated by the end of the time step $t+t^*$, under the proposed (ϵ,δ)-greedy policy.

and $\bar{a} \in \mathcal{A}(\bar{x}_{t+m})$ in (32) yielding:

$$\prod_{m=0}^{t^*-1}\{P(x_{t+m}, a_b, x_{t+m+1})\delta_b + P(x_{t+m}, a^*, x_{t+m+1})(1-\epsilon) + \frac{\delta_e}{|\mathcal{A}(x_{t+m})|}\} < \prod_{m=0}^{t^*-1}\{P(\bar{x}_{t+m}, \bar{a}^*, \bar{x}_{t+m+1})(1-\epsilon) + \frac{\epsilon}{|\mathcal{A}(\bar{x}_{t+m})|}\} \quad (33)$$

Due to (21) and (23), the result in (33) is equivalent to $\beta(p^t_{j^*}) < \eta(p^t_{\bar{j}})$ which contradicts (24) completing the proof.[6]

C. Proof of Proposition 4.8

To show this result, it suffices to show that

$$\mathbb{P}_b(x_{t+t^*} \in \mathcal{X}_{\text{goal}}) \geq \mathbb{P}_g(x_{t+t^*} \in \mathcal{X}_{\text{goal}}). \quad (34)$$

The reason is that if at the time step $t + t^*$ an MDP state in $\mathcal{X}_{\text{goal}}$ is reached, then at the next time step $t + t^* + 1$, a DRA state in $\mathcal{Q}_{\text{goal}}$ will be reached. Notice that the MDP states in $\mathcal{X}_{\text{goal}}$ can be reached at the time step $t+t^*$ if any of the MDP paths p^t_j, $j \in \mathcal{J}$, originating at x_t, are followed. Let R_j be a (Bernoulli) random variable that is true if after t^* time steps (i.e., at the time step $t + t^*$), a path p^t_j, $j \in \mathcal{J}$, has been generated under a policy μ. Then, (34) can be equivalently expressed as:

$$\sum_{j\in\mathcal{J}} \mathbb{P}_b(R_j = 1) \geq \sum_{j\in\mathcal{J}} \mathbb{P}_g(R_j = 1). \quad (35)$$

The rest of the proof follows the same logic as the proof of Proposition 4.6. First, we can rewrite (35) as follows:

$$\sum_{j\in\mathcal{J}}\left(\prod_{m=0}^{t^*-1}\left(\sum_{a\in\mathcal{A}(x_{t+m})}\mu_b(s_{t+m}, a)P(x_{t+m}, a, x_{t+m+1})\right)\right) \geq \quad (36)$$

$$\sum_{j\in\mathcal{J}}\left(\prod_{m=0}^{t^*-1}\left(\sum_{\bar{a}\in\mathcal{A}(\bar{x}_{t+m})}\mu_g(\bar{s}_{t+m}, \bar{a})P(\bar{x}_{t+m}, \bar{a}, \bar{x}_{t+m+1})\right)\right).$$

Next, as in the proof of Proposition 4.6, we show that (36) holds by contradiction. Specifically, assume that (36) does not hold. Then, after plugging the values of $\mu_b(s_{t+m}, a)$ and $\mu_g(\bar{s}_{t+m}, \bar{a})$ for all $a \in \mathcal{A}(x_{t+m})$ and $\bar{a} \in \mathcal{A}(\bar{x}_{t+m})$ in (36) and after rearranging the terms, we get that $\sum_{j\in\mathcal{J}} \beta(p^t_j) < \sum_{j\in\mathcal{J}} \eta(p^t_j)$. This contradicts (25) completing the proof.

APPENDIX D
DECAY RATES IN NUMERICAL SIMULATIONS

In this section, we mathematically define the decay rates used for ϵ, δ_b, and δ_e in Section V. The parameter ϵ evolves over episodes $\texttt{epi}$, as $\epsilon(\texttt{epi}) = 1/(\texttt{epi}^\alpha)$ where α is selected to be equal to 0.1 for $\mathfrak{M}_1$ and $\mathfrak{M}_2$ and 0.05 for $\mathfrak{M}_3$. In 'Biased 1', δ_b and δ_b evolve over episodes, as $\delta_b(\texttt{epi}) = (1 - \frac{1}{\texttt{epi}^\beta})\epsilon(\texttt{epi})$ and $\delta_e(\texttt{epi}) = \frac{\epsilon(\texttt{epi})}{\texttt{epi}^\beta}$. We select $\beta = 0.4$ for $\mathfrak{M}_1$ and $\mathfrak{M}_2$ and $\beta = 0.15$ for $\mathfrak{M}_3$.

[6]Notice that $\eta(p^t_j)$ is equal to the probability that, starting from x_t, the MDP path p^t_j, will be generated by the end of the time step $t + t^*$, if the PMDP evolves as per the ϵ-greedy policy.

Observe that $\delta_b(\texttt{epi}) + \delta_e(\texttt{epi}) = \epsilon(\texttt{epi})$. To define 'Biased 2' and 'Biased 3', we need first to define the following function denoted by $g(\texttt{epi})$. If $\texttt{epi} < 100$, then $g(\texttt{epi}) = 1 - 0.9\exp(-A\texttt{epi})$. Otherwise, we have that $g(\texttt{epi}) = 1 - 0.1\exp(-A\texttt{epi})$ for some A. Then, we have that $\delta_b(\texttt{epi})$ and $\delta_e(\texttt{epi})$ evolve as $\delta_b(\texttt{epi}) = (1 - g(\texttt{epi}))\epsilon(\texttt{epi})$ and $\delta_e(\texttt{epi}) = g(\texttt{epi})\epsilon(\texttt{epi})$. This choice prioritizes random exploration during the first 100 episodes. The larger the A, the faster δ_b converges to 0. Regarding $\mathfrak{M}_1$ and $\mathfrak{M}_2$, we select $A = 0.00015$ for 'Biased 2', $A = 0.0015$ for 'Biased 3', and $A = \infty$ for 'Random'. As for $\mathfrak{M}_3$, we choose $A = 0.000015$ for 'Biased 2' and $A = 0.00015$ for 'Biased 3'.

REFERENCES

[1] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[2] D. Dewey, "Reinforcement learning and the reward engineering principle," in *AAAI Spring Symposium Series*, 2014.

[3] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[4] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, "Temporal logic motion control using actor–critic methods," *The International Journal of Robotics Research*, vol. 34, no. 10, pp. 1329–1344, 2015.

[5] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2018.

[6] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *ACM/IEEE International Conference on Cyber-Physical Systems*, Montreal, Canada, 2019.

[7] M. Bouton, J. Karlsson, A. Nakhaei, K. Fujimura, M. J. Kochenderfer, and J. Tumova, "Reinforcement learning with probabilistic guarantees for autonomous driving," *arXiv preprint arXiv:1904.07189*, 2019.

[8] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, Nice, France, 2019, pp. 5338–5343.

[9] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10 349–10 355.

[10] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic ltl motion planning with environment and motion uncertainties," *IEEE Transactions on Automatic Control*, vol. 66, no. 5, pp. 2386–2392, 2020.

[11] A. Lavaei, F. Somenzi, S. Soudjani, A. Trivedi, and M. Zamani, "Formal controller synthesis for continuous-space mdps via model-free reinforcement learning," in *ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2020, pp. 98–107.

[12] K. Jothimurugan, S. Bansal, O. Bastani, and R. Alur, "Compositional reinforcement learning from logical specifications," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[13] M. Hasanbeig, D. Kroening, and A. Abate, "Lcrl: Certified policy synthesis via logically-constrained reinforcement learning," in *Quantitative Evaluation of Systems: 19th International Conference, QEST 2022, Warsaw, Poland, September 12–16, 2022, Proceedings*. Springer, 2022, pp. 217–231.

[14] H. Hasanbeig, D. Kroening, and A. Abate, "Certified reinforcement learning with logic guidance," *Artificial Intelligence*, vol. 322, 2023.

[15] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Learning optimal strategies for temporal tasks in stochastic games," *IEEE Transactions on Automatic Control*, 2024.

[16] Z. Xuan, A. K. Bozkurt, M. Pajic, and Y. Wang, "On the uniqueness of solution for the bellman equation of ltl objectives," in *Learning for Dynamics and Control*, 2024.

[17] M. Hasanbeig, N. Y. Jeppu, A. Abate, T. Melham, and D. Kroening, "Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7647–7656.

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

[19] Z. Wen, D. Precup, M. Ibrahimi, A. Barreto, B. Van Roy, and S. Singh, "On efficiency in hierarchical reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6708–6718, 2020.

[20] R. Toro Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith, "Reward machines:: Exploiting reward function structure in reinforcement learning," *Journal of Artificial Intelligence Research*, vol. 73, pp. 173–208, 2022.

[21] M. Cai, M. Mann, Z. Serlin, K. Leahy, and C.-I. Vasile, "Learning minimally-violating continuous control for infeasible linear temporal logic specifications," in *American Control Conference (ACC)*, 2023, pp. 1446–1452.

[22] A. Balakrishnan, S. Jakšić, E. A. Aguilar, D. Ničković, and J. V. Deshmukh, "Model-free reinforcement learning for spatiotemporal tasks using symbolic automata," in *62nd IEEE Conference on Decision and Control (CDC)*, Singapore, 2023, pp. 6834–6840.

[23] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[24] Y. Zhai, C. Baek, Z. Zhou, J. Jiao, and Y. Ma, "Computational benefits of intermediate rewards for goal-reaching policy learning," *Journal of Artificial Intelligence Research*, vol. 73, pp. 847–896, 2022.

[25] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2158–2165, 2023.

[26] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[27] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," *Advances in neural information processing systems*, vol. 30, 2017.

[28] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "Ucb exploration via q-ensembles," *arXiv preprint arXiv:1706.01502*, 2017.

[29] S. Amin, M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup, "A survey of exploration methods in reinforcement learning," *arXiv preprint arXiv:2109.00157*, 2021.

[30] J. Fu and U. Topcu, "Probably approximately correct MDP learning and control with temporal logic constraints," *arXiv preprint arXiv:1404.7073*, 2014.

[31] T. Brázdil, K. Chatterjee, M. Chmelik, V. Forejt, J. Křetínskỳ, M. Kwiatkowska, D. Parker, and M. Ujma, "Verification of markov decision processes using learning algorithms," in *Automated Technology for Verification and Analysis: 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings 12*. Springer, 2014, pp. 98–114.

[32] F. Fernández, J. García, and M. Veloso, "Probabilistic policy reuse for inter-task transfer learning," *Robotics and Autonomous Systems*, vol. 58, no. 7, pp. 866–871, 2010.

[33] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 812–836, 2020.

[34] Y. Kantaros, S. Kalluraya, Q. Jin, and G. J. Pappas, "Perception-based temporal logic planning in uncertain semantic maps," *IEEE Transactions on Robotics*, 2022.

[35] X. Ding, M. Lazar, and C. Belta, "Ltl receding horizon control for finite deterministic systems," *Automatica*, vol. 50, no. 2, pp. 399–408, 2014.

[36] B. Lacerda, D. Parker, and N. Hawes, "Optimal policy generation for partially satisfiable co-safe ltl specifications." in *International Joint Conference on Artificial Intelligenc*, vol. 15. Citeseer, 2015, pp. 1587–1593.

[37] Y. Kantaros, "Accelerated reinforcement learning for temporal logic control objectives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyoto, Japan, October 2022.

[38] S. Sickert, J. Esparza, S. Jaax, and J. Křetínskỳ, "Limit-deterministic Büchi automata for linear temporal logic," in *CAV*, 2016, pp. 312–332.

[39] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible ltl constraints," *IEEE Transactions on Automatic Control*, 2021.

[40] Software:, `https://github.com/kantaroslab/AccRL`.

[41] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[42] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[43] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Persistent surveillance for unmanned aerial vehicles subject to charging and temporal logic constraints," *Autonomous Robots*, vol. 40, no. 8, pp. 1363–1378, 2016.

[44] M. Guo and M. M. Zavlanos, "Distributed data gathering with buffer constraints and intermittent communication," in *International Conference on Robotics and Automation*, May-June 2017, pp. 279–284.

[45] Y. Kantaros and M. M. Zavlanos, "Distributed intermittent connectivity control of mobile robot networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3109–3121, 2017.

[46] J. Fang, Z. Zhang, and R. V. Cowlagi, "Decentralized route-planning for multi-vehicle teams to satisfy a subclass of linear temporal logic specifications," *Automatica*, vol. 140, p. 110228, 2022.

[47] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 1002–1028, 2020.

[48] X. C. D. Ding, S. L. Smith, C. Belta, and D. Rus, "Ltl control in uncertain environments with probabilistic satisfaction guarantees," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 3515–3520, 2011.

[49] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Transanctions on Automatic Control*, 2018.

[50] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[51] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Trans. on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

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

[53] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, no. 11, pp. 2724–2734, 2008.

[54] ltl2dstar, `https://www.ltl2dstar.de/`.

Yiannis Kantaros (S'14-M'18) is an Assistant Professor in the Department of Electrical and Systems Engineering, Washington University in St. Louis (WashU), St. Louis, MO, USA. He received the Diploma in Electrical and Computer Engineering in 2012 from the University of Patras, Patras, Greece. He also received the M.Sc. and the Ph.D. degrees in mechanical engineering from Duke University, Durham, NC, in 2017 and 2018, respectively. Prior to joining WashU, he was a postdoctoral associate in the Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA. His current research interests include machine learning, distributed control and optimization, and formal methods with applications in robotics. He received the Best Student Paper Award at the IEEE Global Conference on Signal and Information Processing (GlobalSIP) in 2014, a Best Multi-Robot Systems Paper Award, Finalist, at the IEEE International Conference in Robotics and Automation (ICRA) in 2024, the 2017-18 Outstanding Dissertation Research Award from the Department of Mechanical Engineering and Materials Science, Duke University, and a 2024 NSF CAREER Award.

Jun Wang (S'22) is a PhD candidate in the Department of Electrical and Systems Engineering at Washington University in St. Louis. He received his B.Eng. degree in Software Engineering from Sun Yat-Sen University in 2019 and his MSE degree in Robotics from the University of Pennsylvania in 2021. His research interests include robotics, machine learning, and control theory.