Learning-Based Probabilistic LTL Motion Planning With Environment and Motion Uncertainties

Mingyu Cai, Hao Peng, Zhijun Li, and Zhen Kan

***Abstract*—This article considers control synthesis of an autonomous agent with linear temporal logic (LTL) specifications subject to environment and motion uncertainties. Specifically, the probabilistic motion of the agent is modeled by a Markov decision process (MDP) with unknown transition probabilities. The operating environment is assumed to be partially known, where the desired LTL specifications might be partially infeasible. A relaxed product MDP is constructed that allows the agent to revise its motion plan without strictly following the desired LTL constraints. A utility function composed of violation cost and state rewards is developed. Rigorous analysis shows that, if there almost surely (i.e., with probability 1) exists a policy that satisfies the relaxed product MDP, any algorithm that optimizes the expected utility is guaranteed to find such a policy. A reinforcement learning-based approach is then developed to generate policies that fulfill the desired LTL specifications as much as possible by optimizing the expected discount utility of the relaxed product MDP.**

***Index Terms*—Linear temporal logic (LTL), Markov decision process (MDP), motion planning, reinforcement learning.**

I. Introduction

Autonomous agents are often tasked with complex missions operating in an environment with uncertainties. Besides typical motion uncertainties, e.g., agent may not exactly follow the controls due to potential sensing noise or actuation failures, environment uncertainties are often encountered in practice. For example, when operating in a partially known environment, desired missions can be infeasible if the real environment is found during the runtime to be prohibitive to the agent. Motivated by these practical challenges, this article focuses on the probabilistic motion planning of an autonomous agent subject to environment and motion uncertainties.

Motion planning with linear temporal logic (LTL) constraints has generated substantial interest in robotics (cf., [1]–[3] to name a few). To address environment uncertainties, LTL constraints that cannot be fully satisfied in the given environment are often relaxed to allow the tasks to be fulfilled as much as possible. For instance, control synthesis under soft LTL constraints was developed in [4] to maximize the satisfaction of the LTL constraints if the desired task cannot be completed. A minimal revision problem was considered in [5] and [6] where the revised motion planning is closest to the original LTL. A least-violating control strategy was investigated in [7] for potentially infeasible tasks within a partially known workspace. While LTL control synthesis with environment uncertainties was investigated, motion uncertainties are largely ignored in [4]–[7].

The Markov decision process (MDP) is widely used to model probabilistic motion of robotic systems. Recently, there is growing interest in the control synthesis of MDPs with LTL task specifications by probabilistic model checking (cf., [8]–[12] to name a few). If the transition probabilities in MDP are known *a priori,* existing methods such as probabilistic model checking and dynamic programming can be used to identify policies satisfying LTL specifications [13]. If an MDP with unknown transition probabilities is considered, learning approaches are often employed to learn optimal policies in the presence of motion uncertainties [14]–[16]. For instance, model-based reinforcement learning was employed in the works of [17]–[19] to synthesize control policies satisfying given LTL specifications. In [15], the Q-learning is extended for agents with unknown stochastic dynamics to achieve robust satisfaction of signal temporal logic specifications. In [20], deep reinforcement learning was employed to synthesize control policies for mobile robots with LTL motion constraints and stochastic motion uncertainties. In [21] and [22], instead of the traditional Rabin automata, a limit-deterministic Bchi automata (LDBA) is used to synthesize policies maximizing the satisfaction of given LTL specifications. Although the cases in which accepting maximum end components (AMECs) do not exist were partially addressed in the works of [20], [21], and [23], it is still unclear how these approaches can be extended to handle cases in which the absence of AMECs are caused by infeasible environments. Other representative approaches that address motion uncertainties include the robust policy maximizing the worst-case satisfaction probability [24] and the finite-memory control policy optimizing both the prefix and suffix of the system trajectory [25]. Despite extensive studies of motion uncertainties on MDPs, it remains unclear how the aforementioned results can be extended to handle environment uncertainties.

This article considers control synthesis of an autonomous agent with LTL specifications subject to both environment and motion uncertainties. Specifically, the probabilistic motion of the agent is modeled by an MDP with unknown transition probabilities to capture motion uncertainties. The operating environment is assumed to be uncertain (e.g., partially known), resulting in the possibility that the desired mission might not be fully realized by the agent. A relaxed product MDP is constructed based on the MDP and the deterministic Rabin automata (DRA) generated from the desired LTL specifications [26], which allows the agent to revise its motion plan without strictly following the desired LTL constraints. That is, any policy satisfying the acceptance condition of the relaxed product MDP is feasible in the real environment. To evaluate the revised motion plans, a utility function composed of violation cost and state rewards is developed, where the violation cost function is designed to quantify the differences between the revised and the desired motion plan, and the state-rewards are

Manuscript received January 28, 2020; accepted June 29, 2020. Date of publication July 3, 2020; date of current version April 26, 2021. Recommended by Associate Editor L. Palopoli. *(Corresponding author: Zhen Kan.)*

Mingyu Cai is with the Department of Mechanical Engineering, The University of Iowa, Iowa City, IA 52246 USA (e-mail: mingyu-cai@uiowa.edu).

Hao Peng is with the Apex.AI Inc, Palo Alto, CA 94303 USA (e-mail: haopeng@ufl.edu).

Zhijun Li and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230052, China (e-mail: zjli@ieee.org; zkan@ustc.edu.cn).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2020.3006967

designed to enforce the satisfaction of the relaxed product MDP. Since the workspace is only partially known, real-time sensed information is used to update the relaxed product MDP to facilitate the revision of motion plans. A reinforcement learning-based approach is then developed to generate policies that fulfill the LTL specifications as much as possible by optimizing the expected discount utility of the relaxed product MDP.

This article is closely related to the works of [17] and [25]. Particularly, the relaxed product MDP developed to handle environment uncertainties is inspired by the soft and hard constraints in [25], while the learning-based approach developed to handle uncertain transitions of the MDP is inspired by [17]. However, differing from [17] and [25], this article jointly considers control synthesis of the MDP with unknown transition probabilities (i.e., motion uncertainties) in a partially known workspace (i.e., environment uncertainties). Specifically, the relaxed product MDP originally developed for nondeterministic automata in [25] is modified in this article to handle deterministic automata. The model-based learning in [17] is not suitable for large-sized problems since it has to store all transition probabilities and state values. Moreover, the optimality of the policy in [17] is sensitive to the accuracy of the learned transition probabilities. In contrast, a model-free learning approach is developed in this article, which relaxes such limitations. Compared to the work of [18], which returns no policies if there do not exist AMECs in the product MDP, this article does not require the computation of AMECs. In addition, based on the designed utility function, the control synthesis problem with relaxed LTL specifications is translated to an expected utility optimization problem in this article. Rigorous analysis shows that, if there almost surely (i.e., with probability 1) exists a policy that satisfies the relaxed product MDP, any algorithm that optimizes the expected utility is guaranteed to find such a policy. A crucial benefit of this result is that advances from optimization and learning approaches can be leveraged to address control synthesis problems for the dynamic system with motion and environment uncertainties.

II. PRELIMINARIES

An LTL formula is built on a set of atomic propositions Π, which are properties of system states that can be either true or false, and standard Boolean operators such as $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation), and temporal operators $\Diamond$ (eventually), $\bigcirc$ (next), and $\Box$ (always). The semantics of the LTL formula are defined over words, which are an infinite sequence $o = o_0 o_1 \dots$ with $o_i \in 2^{\Pi}$ for all i, where 2^{Π} represents the power set of Π. Denote by $o \models \phi$ if the word o satisfies the LTL formula ϕ, and a word satisfies ϕ if ϕ is true at the first position of the word. Detailed descriptions of the syntax and semantics of the LTL can be found in [8].

An LTL formula can be translated to a DRA.

Definition 1: A DRA is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states; Σ is the input alphabet; $\delta : Q \times \Sigma \to Q$ is the transition function; $q_0 \in Q$ is the initial state; and $F = \{(L_1, K_1), (L_2, K_2), \dots, (L_{n_F}, K_{n_F})\}$ is a set of pairs where $L_i, K_i \subseteq Q, \forall i \in \{1, \dots n_F\}$.

A run of a DRA $\mathcal{R}$ is an infinite sequence $r_R = q_0 q_1 \dots$ where, for each $i \geq 0$, $q_i \in Q$ and $q_{i+1} = \delta(q_i, \sigma)$ for some input $\sigma \in \Sigma$. A run $r_R = q_0 q_1 \dots$ is accepting if there exists a pair (L_i, K_i), $i \in \{1, \dots n_F\}$, such that $\inf(r_R) \cap L_i = \emptyset$ and $\inf(r_R) \cap K_i \neq \emptyset$, where $\inf(r_R)$ represents a set of states in r_R that is visited infinitely often. For any LTL formula ϕ over Π, one can construct a DRA with input alphabet $\Sigma = 2^{\Pi}$ accepting all and only words that satisfy ϕ [27]. Denote by $\mathcal{R}_\phi$ the DRA generated from ϕ. To convert an LTL formula to a DRA, readers are referred to [26] and the references therein for algorithms with freely available implementations.

Transition systems with motion uncertainties are often modeled as an MDP.

Definition 2: A labeled finite MDP is a tuple $\mathcal{M} = (S, \mathcal{A}, P, s_0, \Pi, L)$, where S is a finite set of states; A is a finite set of actions and $\mathcal{A} : S \to 2^A$ represents the set of actions enabled at state $s \in S$; $P : S \times A \times S \to [0, 1]$ is the transition probability function such that, for all states $s \in S$, $\sum_{s' \in S} P(s, a, s') = 1$ if $a \in \mathcal{A}(s)$ and $P(s, a, s') = 0$ if $a \notin \mathcal{A}(s)$; $s_0 \in S$ is the initial state; Π is a set of atomic propositions; and $L : S \to 2^{\Pi}$ is a labeling function.

Based on the DRA and MDP in Def. 1 and 2, a product MDP can be constructed.

Definition 3: A product MDP between a labeled MDP $\mathcal{M} = (S, \mathcal{A}, P, s_0, \Pi, L)$ and a DRA $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$ is defined as a tuple $P_M = (S_{P_M}, A_{P_M}, P_{P_M}, s_{P_M 0}, F_{P_M})$, where $S_{P_M} = S \times Q$ is the set of states; $A_{P_M}(s_{P_M}) = \mathcal{A}(s)$ where $s_{P_M} = (s, q) \in S_{P_M}$; $P_{P_M}(s_{P_M}, a, s'_{P_M}) = P(s, a, s')$ if $\delta(q, L(s')) = q'$ and $P_{P_M}(s_{P_M}, a, s'_{P_M}) = 0$ otherwise, where $s'_{P_M} = (s', q')$; $s_{P_M 0} = (s_0, q)$ with $q = \delta(q_0, L(s_0))$ is the initial state; and $F_{P_M} = \{(L_{P_M 1}, K_{P_M 1}), (L_{P_M 2}, K_{P_M 2}), \dots, (L_{P_M n_F}, K_{P_M n_F})\}$ is the acceptance condition, where $L_{P_M i} = S \times L_i$ and $K_{P_M i} = S \times K_i$ for all $i \in \{1, \dots n_F\}$

III. PROBLEM FORMULATION

Consider a labeled finite MDP $\mathcal{M} = (S, \mathcal{A}, P, s_0, \Pi, L)$ representing the probabilistic motion of an agent in an operating environment, where partitioned areas of the environment are abstracted to states in S, the transitions of states in S represent the transitions between adjacent areas, and $P(s_i, a, s_j)$ captures the transition probability under a control action $a \in \mathcal{A}(s_i) \forall s_i, s_j \in S$. The transition probabilities are assumed to be unknown *a priori* to model motion uncertainties. The atomic propositions Π represent properties of S in the environment, and the labeling function L indicates the associated properties (e.g., if a state in S is an obstacle or a destination). The high-level task specifications to be performed by the agent are then described by an LTL formula ϕ over Π.

Let $\mu : S \to \mathcal{A}$ be an action function such that $\mu(s_i) \in \mathcal{A}(s_i)\ \forall s_i \in S$. A policy π over the MDP $\mathcal{M}$ is an infinite sequence of action functions $\pi = \{\mu_0, \mu_1, \dots\}$, which resolves nondeterministic choices in $\mathcal{M}$ by applying μ_k at each time step k. Hence, the policy π induces a Markov chain over $\mathcal{M}$, denoted by $MC^{\pi}_{\mathcal{M}}$. If $\mu_k = \mu$ for all k, then π is called a stationary policy. A path of $\mathcal{M}$ under π (i.e., a run of $MC^{\pi}_{\mathcal{M}}$) is then defined as an infinite sequence of states $r^{\pi}_{\mathcal{M}} = s_0 s_1 \dots$, where the state transitions satisfy $P(s_i, \mu_i(s_i), s_{i+1}) > 0$ for all i. A path $r^{\pi}_{\mathcal{M}}$ generates a word $o = o_0 o_1 \dots$ where $o_i = L(s_i)$ for all i. Let $L(r^{\pi}_{\mathcal{M}})$ denote the word generated by the path $r^{\pi}_{\mathcal{M}}$.

In this article, the environment is assumed to only partially known. For instance, the agent may only know the destinations to be visited, without knowing the potential obstacles it may encounter. Due to the lack of environment knowledge, it is possible that the preassigned task ϕ cannot be fully accomplished (e.g., a destination is surrounded by water that the mobile robot cannot traverse). Therefore, considering a labeled MDP with unknown transition probabilities, this article aims to identify policies π over $\mathcal{M}$ that can handle infeasible LTL specifications (i.e., satisfy the desired task ϕ the most whenever the environment is infeasible).

IV. Learning-Based Control Synthesis

Since the desired ϕ can be infeasible in the given environment, Section IV-A discusses how the original ϕ can be relaxed and how the violation of ϕ can be quantified. The control synthesis problem with relaxed LTL specifications is then translated to an expected utility optimization problem in Section IV-B, where a learning-based approach is developed to solve the utility optimization problem in Section IV-C.

A. Relaxed Product MDP

When operating in an infeasible environment, the traditional product MDP in Definition 3 needs to be relaxed so that the revised motion does not have to strictly stick with the infeasible LTL specifications.

Definition 4: A relaxed product MDP between a labeled MDP $\mathcal{M} = (S, \mathcal{A}, P, s_0, \Pi, L)$ and a DRA $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$ is defined as a tuple $\mathcal{P} = (S_P, A_P, P_P, s_{p0}, F_P, W, V, L_P)$, where

1) $S_P = S \times Q$ is the set of states, e.g., $s_p = (s, q) \in S_P$ and $s'_p = (s,' q') \in S_P$;
2) A_P represents extended actions, e.g., $a_P = (a, l) \in A_P$;
3) $P_P(s_p, a_P, s'_p) = P(s, a, s')$, if $P(s, a, s') \neq 0$ with $a \in \mathcal{A}(s)$ and $\exists l \in 2^\Pi$ such that $\delta(q, l) = q'$, and $P_P(s_p, a_P, s'_p) = 0$ otherwise;
4) F_P is the acceptance condition given by $F_P = \{(L_{p1}, K_{p1}), (L_{p2}, K_{p2}), \ldots, (L_{pn_F}, K_{pn_F})\}$, where $L_{pi} = S \times L_i$ and $K_{pi} = S \times K_i \ \forall i \in \{1, \ldots n_F\}$;
5) $L_p : S_P \to 2^\Pi$ is a labeling function such that $L_P(s_p) = L(s)$;
6) $W = \{W^i\}_{i=1}^{n_F}$ is a set of state-reward functions where $W^i : S_P \to \mathbb{R}$ is defined as

$$W^i = \begin{cases} w_L^i, & \text{if } s_p \in L_{pi} \\ w_K^i, & \text{if } s_p \in K_{pi} \\ 0, & \text{if } s_p \in S_P \setminus (L_{pi} \cup K_{pi}) \end{cases} \tag{1}$$

where $w_L^i < 0$ and $w_K^i > 0$;
7) $V : S_P \times L_P \times S_P \to \mathbb{R}^-$ represents the violation cost over state transitions.

Let $\boldsymbol{W}^i \in \mathbb{R}^{|S_P|}$ denote the stacked rewards of states in S_P. In (1), the negative reward w_L^i on L_{Pi}, the positive reward w_K^i on K_{Pi}, and the reward of 0 on neutral states are designed to bias the policy toward the acceptance condition of $\mathcal{P}$, since L_{Pi} and K_{Pi} need to be visited finitely and infinitely often, respectively. Since the traditional relaxed product MDP is only valid for nondeterministic automata, the extended actions A_P are designed to enable deterministic automata. Specifically, given a state $s_P = (s, q) \in S_P$, we design $A_P(s_P) = \{a_P = (a, l)\}$, where $a \in \mathcal{A}(s)$ and $l \in 2^\Pi$ such that $\exists q' \in Q$ for $\delta(q, l) = q'$. That is, under an action $a_P \in A_P(s_P)$, one always has $\sum_{s'_p \in S_P} P_P(s_P, a_P, s'_p) = 1$, ensuring that $\mathcal{P}$ is a valid MDP. Denote by $\Gamma(a_P) = a$ the projection of a_P to the action of A on $\mathcal{M}$.

To define V, suppose that $\Pi = \{\alpha_1, \alpha_2 \ldots \alpha_M\}$ and consider an evaluation function $\text{Eval} : 2^\Pi \to \{0, 1\}^M$, where $\text{Eval}(l) = \{v_i\}^M$ with $v_i = 1$ if $\alpha_i \in l$ and $v_i = 0$ if $\alpha_i \notin l$, where $i = 1, 2, \ldots, M$ and $l \in 2^\Pi$. To quantify the difference between two elements in 2^Π, consider $\rho(l, l') = \|v - v'\|_1 = \sum_{i=1}^M |v_i - v'_i|$, where $v = \text{Eval}(l)$, $v' = \text{Eval}(l')$, $l, l' \in 2^\Pi$, and $\|\cdot\|_1$ is the l_1 norm. The distance from $l \in 2^\Pi$ to a set $\mathcal{X} \subseteq 2^\Pi$ is then defined as $\text{Dist}(l, \mathcal{X}) = 0$ if $l \in \mathcal{X}$ and $\text{Dist}(l, \mathcal{X}) = \min_{l' \in \mathcal{X}} \rho(l, l')$ if $l \notin \mathcal{X}$. Now the violation cost of the transition from $s_P = (s, q)$ to $s'_P = (s', q')$ is defined as

$$V\left(s_p, L_P\left(s'_p\right), s'_p\right) = -\text{Dist}\left(L_P\left(s'_p\right), \mathcal{X}(s_p, s'_p)\right) \tag{2}$$

where $\mathcal{X}(s_p, s'_p) = \mathcal{X}(q, q') = \{l \in 2^\Pi | q' = \delta(q, l)\}$ is the set of input alphabets that enables the transition from q to q'. Hence, the violation cost $V(s_p, L_P(s'_p), s'_p)$ quantifies how much the transition from s_p to s'_p in $\mathcal{P}$ violates the constraints imposed by ϕ. Note that, $\mathcal{X} \neq \emptyset$, since there always exists an $l \in 2^\Pi$ such that $q' = \delta(q, l)$ by Definition 4, which indicates that a policy fully satisfying ϕ has zero violation cost.

There are two differences between the P_M in Definition 3 and the relaxed $\mathcal{P}$ in Definition 4. First, the constraints $\delta(q, L(s')) = q'$ of the transition from $s_p = (s, q)$ to $s'_p = (s,' q')$ on P_M is relaxed in $\mathcal{P}$ only requiring there exists $l \in 2^\Pi$ such that $\delta(q, l) = q'$, which allows the relaxation of the constraints of ϕ. Consequently, $\mathcal{P}$ is more connected than P_M in terms of possible transitions. Second, the designed violation cost V over transitions can facilitate the selection of policies close to the original ϕ if the environment is infeasible.

Consider a relaxed product MDP $\mathcal{P}$ generated by $\mathcal{M}$ and $\mathcal{R}_\phi$ corresponding to an LTL formula ϕ. Let $\pi : S_P \to A_P$ be a stationary policy on $\mathcal{P}$ that maps each state $s_P \in S_P$ to an action $a_P \in A_P(s_P)$. Denote by $r_\mathcal{P}^\pi = s_{p0} s_{p1} \ldots$ a run of $\mathcal{P}$ initialized at s_{p0} under π.

Assumption 1: It is assumed that there almost surely (i.e., with probability 1) exists a run $r_\mathcal{P}^\pi$ such that, $\exists (L_{pi}, K_{pi}) \in F_P, \inf(r_\mathcal{P}^\pi) \cap L_{pi} = \emptyset \wedge \inf(r_\mathcal{P}^\pi) \cap K_{pi} \neq \emptyset$.

Assumption 1 indicates the acceptance condition of the relaxed MDP can be reached with probability 1. As discussed in [8], Assumption 1 is mild and only a qualitative property of Markov chains, which indicates the motion planning problem is feasible. In other words, we only assume the almost sure existence of a path $r_\mathcal{P}^\pi$ satisfying the acceptance condition of $\mathcal{P}$. Since there may exist multiple accepting runs, the following section will focus on the synthesis of policies that stay close to the original LTL specification ϕ while satisfying the acceptance condition of $\mathcal{P}$ when the environment is infeasible. It should be noted that an accepting run $r_\mathcal{P}^\pi$ in Assumption 1 does not necessarily indicate the desired task ϕ is fully satisfied. Instead, an accepting run $r_\mathcal{P}^\pi$ is a relaxed motion planning for the agent when the environment is infeasible.

B. Control Synthesis

This section focuses on identifying policies that induce accepting paths of $\mathcal{P}$ while staying close to the original ϕ. Since the closeness to ϕ can be measured via the violation cost V, a utility function is designed and the subsequent analysis shows that maximizing the expected utilities can find a policy that fulfills the LTL specification ϕ as much as possible.

For $i \in \{1, \ldots, n_F\}$, let $\mathcal{P}^i$ denote $\mathcal{P}$ with the state-reward function W^i in (1) and $\boldsymbol{U}_\pi^i = [U_\pi^i(s_{p0})\ U_\pi^i(s_{p1})\ \ldots]^T \in \mathbb{R}^N$ denotes the stacked expected discounted utility induced by the policy π on $\mathcal{P}^i$, where $N = |S_P|$. The utility is designed as

$$\boldsymbol{U}_\pi^i = \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_\pi^n \left(\boldsymbol{W}^i + \beta \boldsymbol{R}_\pi \mathbf{1}_N\right) \tag{3}$$

where $0 < \gamma < 1$ is a discount factor, $\boldsymbol{P}_\pi \in \mathbb{R}^{N \times N}$ contains the probabilities $P_P(s_p, \pi(s_p), s'_p)$ under π for all $s_p, s'_p \in S_P$, $\boldsymbol{W}^i = [W^i(s_{p0})\ W^i(s_{p1})\ \ldots]^T \in \mathbb{R}^N$ is the stacked state-rewards, $\beta \in \mathbb{R}^+$ is a weight indicating the relative importance, $\mathbf{1}_N$ is an N-dimensional vector of ones, and $\boldsymbol{R}_\pi = \boldsymbol{P}_\pi \circ \boldsymbol{V} \in \mathbb{R}^{N \times N}$ is the Hadamard product of $\boldsymbol{P}_\pi$ and $\boldsymbol{V}$, i.e., $\boldsymbol{R}_\pi = [P_P(s_p, \pi(s_p), s'_p) \cdot V(s_p, L_P(s'_p), s'_p)]_{N \times N}$, where $\boldsymbol{V} \in \mathbb{R}^{N \times N}$ contains the violation cost between pairs of states in S_P. For notational convenience, the superscript i (i.e., the index of the Rabin acceptance condition of the LTL specification) is omitted in the rest of this work, and we use $\boldsymbol{W}$ and $\boldsymbol{U}_\pi$ to represent the reward and utility vectors of $\mathcal{P}$ with the Rabin acceptance condition (L_{pi}, K_{pi}).

Based on the defined utility in (3), an optimal policy π^* is designed as

$$\pi^* = \arg\max_\pi \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_\pi^n \left(\boldsymbol{W} + \beta \boldsymbol{R}_\pi \mathbf{1}_N\right). \tag{4}$$

Since the policy π^* in (4) is optimal, $U_\pi(s_p) \leq U_{\pi^*}(s_p)$ for all $s_p \in S_P$. When considering all pairs (L_{pi}, K_{pi}), $i = 1, \ldots, n_F$, solving (4) yields a collection of policies $\{\pi_i^*\}_{i=1}^{n_F}$. Note that $\boldsymbol{V}$ is a nonpositive matrix, since the violation cost of transitions between any pair of states in (2) is nonpositive. Consequently, $\boldsymbol{R}_\pi$ in (4) is a nonpositive matrix, due to the Hadamard product of the probability matrix $\boldsymbol{P}_\pi$ and $\boldsymbol{V}$. If a Markov chain $MC_{\mathcal{M}}^{\pi}$ induced by π fully satisfies ϕ, the violation cost associated with the satisfactory path is zero. Therefore, maximizing the utilities in (4) tends to bias the selection of policies toward those that can generate Markov chains close to the original LTL specification ϕ.

Theorem 1: Consider a relaxed MDP product $\mathcal{P}$ generated by $\mathcal{M}$ and $\mathcal{R}_\phi$ corresponding to an LTL formula ϕ. If there exists a policy $\bar{\pi}$ such that an induced run $r_{\mathcal{P}}^{\bar{\pi}}$ satisfies Assumption 1, then there exists $i^* \in \{1, \ldots n_F\}$ such that any optimization method that obtains the solution to (4) of $\mathcal{P}^{i^*}$ with β, γ, w_L, and w_K satisfying the conditions in (12), (15), (16), and (19) can find the policy $\bar{\pi}$.

Proof: See the Appendix for the proof. ■

Theorem 1 indicates that the control synthesis with relaxed LTL specifications can be cast into an expected utility optimization problem. Solving (4) can generate a collection of strategies $\{\pi_i^*\}_{i=1}^{n_F}$ for each $\mathcal{P}^i$. As demonstrated in Theorem 1, with a proper design of β, γ, w_L, and w_K, there exists at least one policy in $\{\pi_i^*\}_{i=1}^{n_F}$ guaranteed to satisfy the acceptance condition of the relaxed MDP.

C. Reinforcement Learning-Based Strategy

This section develops a model-free learning strategy to address the utility optimization problem in (4), which is outlined in Algorithm 1. Due to the limited environmental knowledge, the system starts with an initial, possibly imprecise, belief about the environment. To correct the potentially imprecise state labels (e.g., a state s'_p that is believed to be an obstacle initially can then be found to not be), the state label $L_P(s'_p)$ needs to be updated based on newly sensed information at current state s_p before applying learning algorithms. It is assumed that the environment remains static between two consecutive samplings. Specifically, let $\text{Info}(s_p) = \{L_P(s'_p) | s'_p \in \text{Sense}(s_p)\}$ denote the observed labels of s'_p that are different from the current belief, where $\text{Sense}(s_p)$ represents a local set of states that can be sensed at s_p. If the sensed labels $L_P(s'_p)$ are consistent with the current belief of s_p, $\text{Info}(s_p) = \emptyset$. Otherwise, the properties of s'_P need to be updated. For a given LTL formula ϕ, the violation cost defined in (2) depends only on the state labels. Note that $L_P(s_p) = L(s)$, where $s_p = (s, q)$, which indicates that all states $s_p \in S_P$ with the same s need to be updated. Let $[\![s_P]\!] = \{s_p = (s, q) | q \in Q\}$ denote a class of s_p with the same s. Let $L_P([\![s'_p]\!])$ denote the label of the class $[\![s'_p]\!]$ and let $V([\![\hat{s}_p]\!], L_P([\![s'_p]\!]), [\![s'_p]\!])$ denote the violation cost corresponding to the transitions from $[\![\hat{s}_p]\!]$ to $[\![s'_p]\!]$, where $[\![\hat{s}_p]\!]$ represents a class of states that can transit to $[\![s'_p]\!]$ in one step. Lines 7–14 in Algorithm 1 show how $\text{Info}(\cdot)$ is used to update the state labels and the violation cost associated with state transitions.

As discussed in [28], reinforcement learning occurs over multiple trials where the system has to be reset periodically. In addition, there exist deadlock states in the DRA from which there does not exist a trace satisfying the accepting condition. Since the learning process may enter deadlock states or causes safety violation, inspired by [17], the function ResetRabinState is introduced to periodically reset the initial condition of the Rabin component in MDP $\mathcal{P}$, whenever either a set time interval is reached or a safety violation occurs (i.e., lines 17 and 18 in Algorithm 1). To address the exploitation versus exploration issue in reinforcement learning, the diminishing ε-greedy approach in [29] is applied. That is, the exploration and the exploitation in the learning process have the probability ε and $1 - \varepsilon$, respectively, where ε is set close to 1 initially to encourage exploration and gradually decreases to encourage exploitation as the learning evolves.

Algorithm 1: Learning-Based Control Synthesis.

1: **procedure** INPUT: LTL specification ϕ, initial MDP $\mathcal{M}$, α, β, γ, w_L, w_K, and the iteration threshold $\bar{k}$
   Output: the optimal policy π^*
   Initialization: construct the product MDP $\mathcal{P}$, initialize U and the policy π^*, set episode $k_e = 0$ and iteration $k = 0$;
2: **while** U is not converged **do**
3: $k_e + +$;
4: $s_p = s_{p0}$;
5: **while** $k \leq \bar{k}$ **do**
6: $k + +$;
7: **if** $\text{Info}(s_p) \neq \emptyset$ **then**
8: **for** all s'_p such that $L_P(s'_p) \in \text{Info}(s_p)$ **do**
9: Update the labels of $[\![s'_p]\!]$ to $L_P(s'_p)$;
10: **for** all $\hat{s}_p$ such that $\exists a_p$ for $P_P(\hat{s}_p, a_p, s'_p) \neq 0$ **do**
11: Update the violation cost $V([\![\hat{s}_p]\!], L_P([\![s'_p]\!]), [\![s'_p]\!])$;
12: **end for**
13: **end for**
14: **end for**
15: Select a current action a_p based on ϵ-greedy method;
16: $s_p \xrightarrow{a_P} s'_p$
17: **if** ResetRabin() is True **then**
18: $s'_p \leftarrow \text{ResetRabinState}(s'_p)$;
19: **else if** s_p is not `Null` **then**
20: Receive $R(s_p, a_p, s'_p)$;
21: $Q(s_p, a_p) \leftarrow (1-\alpha)Q(s_p, a_p) + \alpha\Big[R(s_p, a_p, s'_p) + \gamma \max_{a_p \in A_P(s_P)} Q(s'_p, a_p)\Big]$;
22: $U_\pi^*(s_p) = \max_{a_p \in A_P(s_P)} Q^*(s_p, a_p)$;
23: $\pi^*(s_p) = \arg\sup_{a_p \in A_P(s_P)} U_\pi^*(s_p)$;
24: **end if**
25: **end while**
26: **end while**
27: **end procedure**

Based on the Q-learning [30], the agent updates its Q-value after transiting from s_p to s'_p under an action a_p according to

$$\begin{aligned} Q\left(s_p, a_p\right) \leftarrow\ & (1-\alpha)\, Q\left(s_p, a_p\right) \\ & + \alpha\left[R\left(s_p, a_p, s'_p\right) + \gamma \max_{a_p \in A_P(s_P)} Q\left(s'_p, a_p\right)\right] \end{aligned} \tag{5}$$

where $Q(s_p, a_p)$ is the Q-value of the state-action pair (s_p, a_p), $0 < \alpha \leq 1$ is the learning rate, $0 \leq \gamma \leq 1$ is the discount factor, and

$$R\left(s_p, a_p, s'_p\right) = W^i\left(s_p\right) + \beta \cdot V\left(s_p, L_P\left(s'_p\right), s'_p\right) \tag{6}$$

denotes the immediate reward from s_p to s'_p under a_p. With standard learning rate α and discount factor γ as in [30], Q-value will converge to a unique limit Q^*, based on which the optimal expected utility and policy can be obtained as $U_\pi^*(s_p) = \max_{a_p \in A_P(s_P)} Q^*(s_p, a_P)$ and $\pi^*(s_p) = \arg\max_{a_P \in A_P(s_P)} Q^*(s_p, a_P)$. In (5) and (6), the discount γ is tuned for improved tradeoff in using immediate and future rewards, while β is designed to specify the relative importance between maximizing the satisfaction of ϕ and minimizing the violation cost (i.e., lines 20–23 in Algorithm 1).

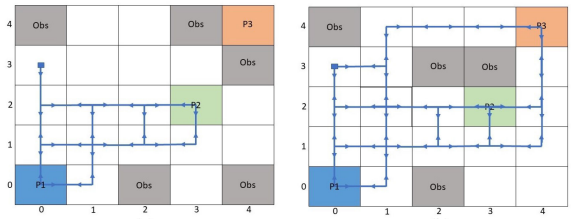

Fig. 1. Grid world example with a sample trajectory generated by policy π^*. The robot is tasked to avoid obstacles and visit $P1$, $P2$, and $P3$ infinitely often. The trajectory starts at $(0, 3)$ marked by the square, and the arrows indicate movements between adjacent cells. (a) and (b) infeasible and feasible cases, respectively.

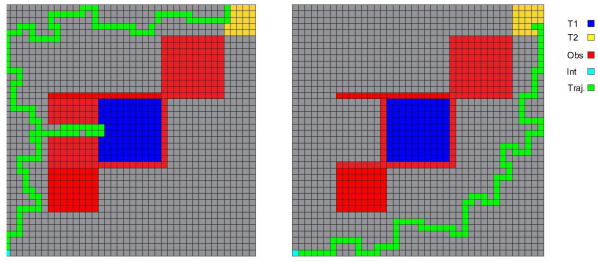

Fig. 2. (a) Planned trajectory of a feasible case. (b) Agent trajectory when the required task is not fully feasible (e.g., $T1$ is not reachable due to the obstacles) in the given environment. The trajectories (i.e., green blocks) in (a) and (b) are generated by Algorithm 1, starting from the left corner (i.e., the cyan block).

In Algorithm 1, the number of states is $|S| \times |Q|$, where $|Q|$ is determined by the DRA $\mathcal{R}_\phi$ and $|S|$ is the size of environment. Due to the consideration of the relaxed product MDP and extended actions in Definition 4, the maximum complexity of actions available at $s_P = (s, q)$ is $O(|\mathcal{A}(s)| \times |\Sigma|)$. Despite the increase of the action complexity, compared with [17], the developed learning approach in Algorithm 1 does not need to approximate and store all transition probabilities.

V. Case Study

Consider a robot tasked to perform $\phi = \square\neg\texttt{Obs} \wedge \square\lozenge P1 \wedge \square\lozenge P2 \wedge \square\lozenge P3$ in a grid environment as shown in Fig. 1. The robot knows the locations of $P1$, $P2$, and $P3$, without knowing the obstacle positions. The motion of the robot within the environment, i.e., "up," "down," "right," and "left," is modeled as an MDP $\mathcal{M}$. It is assumed that the action can be successfully implemented with probability 0.9 and fails with probability 0.1. The transition probability is assumed to be unknown in simulation and needs to be learned. The LTL formula ϕ is translated to a DRA $\mathcal{R}_\phi$ using ltl2dstate [31], where $\mathcal{R}_\phi$ has $|Q| = 9$ states. Based on $\mathcal{R}_\phi$ and $\mathcal{M}$, a relaxed product MDP $\mathcal{P}$ is created, which has $|S_P| = 225$ states.

The robot starts its motion planning according to an initial belief about the environment. Algorithm 1 is called to update the belief based on the sensed information and the observed transitions. Active temporal difference learning is employed to estimate the expected discount utilities and generate optimal policies. The design parameters are set as $\beta = 5$, $\gamma = 0.98$, $\alpha = 0.9$, $w_L = -8$, and $w_K = 10$. A set of 600 iterations, which were separated by the ResetRabinState function in Algorithm 1 every 200 iterations, were performed in simulation. The algorithm is implemented in MATLAB on a PC with Intel i7 4 cores processor, 3.60 GHz, and 32-GB memory.

The simulation results are shown in Fig. 1. Fig. 1(a) represents an infeasible environment where $P3$ is surrounded by obstacles, while Fig. 1(b) represents a feasible environment where ϕ can be fully performed. The lines in Fig. 1(a) and (b) indicate paths that maximize the utilities, and the arrows indicate the optimal actions at each cell. The revised path in Fig. 1(a) is generated based on the relaxed product MDP $\mathcal{P}$, which only visits $P1$ and $P2$ while avoiding obstacles.

To show the effectiveness of our approach in a larger scale problem, we consider a 40×40 grid environment as shown in Fig. 2. The task is expressed as $\varphi = \square\neg\texttt{Obs} \wedge \lozenge T1 \wedge \square(T1 \rightarrow \bigcirc(\neg T1 \mathcal{U} T2))$, where $T1$ and $T2$ represent the destinations to be visited sequentially and Obs represents the obstacles. With similar settings, the simulation results are shown in Fig. 2. Fig. 2(a) represents a feasible environment where φ can be fully performed, while Fig. 2(b) represents an infeasible environment where $T1$ is surrounded by obstacles. The learned trajectories in Fig. 2(a) and (b) are represented in green, indicating the paths that maximize the utilities. Since the environment in Fig. 2(b) is infeasible, a revised path is generated based on the relaxed product MDP $\mathcal{P}$, which only visits $T2$ while avoiding obstacles. In contrast, the path generated in Fig. 2(a) fully satisfies the mission requirement of φ.

VI. Conclusion

This article considers a learning-based method to synthesize control policies for MDP with LTL specifications subject to environment and motion uncertainties. A model-free learning framework is developed to generate policies that fulfill the LTL specifications as much as possible by optimizing the expected discount utility of the relaxed product MDP. The considered optimization has multiobjectives, i.e., maximizing reward collection and minimizing violation cost, which can even be conflicting in many cases. Future research will consider better tradeoff among multiple objectives. Additional research will also consider leveraging tools, such as barrier certificate [32] or cost optimization both in the prefix and suffix of the trajectory [25], to enable motion planning with both hard-constrained and soft-constrained tasks. Since the LDBA has an exponential-sized automaton [21], additional research will consider using the LDBA to reduce the complexity of the automaton and improve the computational efficiency.

Appendix

Proof of Theorem 1: Consider two policies $\bar{\pi}$ and π^*, where $\bar{\pi}$ is a policy satisfying the acceptance condition of $\mathcal{P}$, and π^* is an optimal policy satisfying (4). Let $MC_{\mathcal{P}}^{\bar{\pi}}$ denote the Markov chain induced by the policy $\bar{\pi}$ on $\mathcal{P}$, whose states can be represented by a disjoint union of a transient set $\mathcal{T}_{\bar{\pi}}$ and n closed irreducible recurrent sets $\mathcal{R}_{\bar{\pi}}^j$, $j \in \{1, \ldots, n\}$ [29]. Note that, for policy $\bar{\pi}$, there exists a pair $(L_{pi}, K_{pi}) \in F_P$ such that $L_{pi} \in \mathcal{T}_{\bar{\pi}}$ and $K_{pi} \cap \mathcal{R}_{\bar{\pi}}^j \neq \emptyset$ for all $j \in \{1, \ldots, n\}$. Let Π^* be the set of optimal policies that optimize the expected utility in (4). Similar to $MC_{\mathcal{P}}^{\bar{\pi}}$, the states of $MC_{\mathcal{P}}^{\pi^*}$ under a policy $\pi^* \in \Pi^*$ can also be divided into a transient set $\mathcal{T}_{\pi^*}$ and m recurrent sets $\mathcal{R}_{\pi^*}^j$, $j \in \{1, \ldots, m\}$.

The strategy of the following proof is based on contradiction. We will show that, if an optimal policy $\pi^* \in \Pi^*$ does not satisfy the acceptance condition of $\mathcal{P}$, there always exists a policy $\bar{\pi}$ with greater utility than π^*, which contradicts to the optimality of π^*. Hence, the policy maximizing the utility in (4) is guaranteed to satisfy the acceptance condition of $\mathcal{P}$.

To this end, suppose $\pi^* \in \Pi^*$ does not satisfy $\mathcal{P}$. Then, one of the following two cases must be true.

Case 1: There exists a recurrent set $\mathcal{R}_{\pi^*}^j \cap K_{pi} = \emptyset$, which means K_{pi} is only visited finitely often.

Case 2: There exists a recurrent set $\mathcal{R}_{\pi^*}^j \cap L_{pi} \neq \emptyset$, which means L_{pi} can be visited infinitely often.

Let $\Pi^* = \Pi_1 \cup \Pi_2$, where Π_1 and Π_2 represent the set of policies satisfying Cases 1 and 2, respectively. Based on (3), the expected utilities under π^* is

$$\boldsymbol{U}_{\pi^*} = \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\pi^*}^n \left(\boldsymbol{W} + \beta \boldsymbol{O}_{\pi^*}\right) \tag{7}$$

where $\boldsymbol{U}_{\pi^*} = [U_{\pi^*}(s_{p0})\ U_{\pi^*}(s_{p1})\ \ldots]^T \in \mathbb{R}^N$ is the stacked state utilities and $\boldsymbol{O}_{\pi^*} = [\mathcal{O}_{\pi^*}(s_{p0})\ \mathcal{O}_{\pi^*}(s_{p1})\ \ldots]^T = \boldsymbol{R}_{\pi^*}\mathbf{1}_N$, where N is the number of the states of $MC_{\mathcal{P}}^{\pi^*}$ and $\mathbf{1}_N \in \mathbb{R}^N$ is a vector of all ones. The rewards $\boldsymbol{W} \in \mathbb{R}^N$, transition probability matrix $\boldsymbol{P}_{\pi^*}^n \in \mathbb{R}^{N\times N}$, and $\boldsymbol{R}_{\pi^*} \in \mathbb{R}^{N\times N}$ are all defined similarly as in (4) over the states of $MC_{\mathcal{P}}^{\pi^*}$. Denote $[\boldsymbol{R}_{\pi^*}]_{ij}$ by $R_{ij}^{\pi^*}$ and $R_{ij}^{\pi^*} = P_{P,ij} \cdot V_{ij} \le 0$, due to the nonnegative probability $P_{P,ij}$ and the nonpositive violation cost V_{ij}. Hence, $\boldsymbol{O}_{\pi^*}$ in (7) is element-wise nonpositive, i.e., $\mathcal{O}_{\pi^*}(s_{pi}) \le 0$ for all $i \in \{1, \ldots N\}$.

Suppose there are r transient states (i.e., $|\mathcal{T}_{\pi^*}| = r$) and m recurrent sets $\mathcal{R}_{\pi^*}^i, i \in \{1, \ldots, m\}$, where each $\mathcal{R}_{\pi^*}^i$ has N_i states. The expected utility in (7) can be reorganized as

$$\begin{bmatrix} \boldsymbol{U}_{\pi^*}^{\text{tr}} \\ \boldsymbol{U}_{\pi^*}^{\text{rec}} \end{bmatrix} = \sum_{n=0}^{\infty} \gamma^n \begin{bmatrix} \boldsymbol{P}_{\pi^*}(\mathcal{T},\mathcal{T}) & \boldsymbol{P}_{\pi^*}^{\text{tr}} \\ \mathbf{0}_{\sum_{i=1}^m N_i \times r} & \boldsymbol{P}_{\pi^*}(\mathcal{R},\mathcal{R}) \end{bmatrix}^n \cdot \left(\begin{bmatrix} \boldsymbol{W}^{\text{tr}} \\ \boldsymbol{W}^{\text{rec}} \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{O}_{\pi^*}^{\text{tr}} \\ \boldsymbol{O}_{\pi^*}^{\text{rec}} \end{bmatrix} \right) \tag{8}$$

where $\boldsymbol{U}_{\pi^*}^{\text{tr}}$ and $\boldsymbol{U}_{\pi^*}^{\text{rec}}$ are the utilities of states in transient and recurrent classes, respectively. In (8), $\boldsymbol{P}_{\pi^*}(\mathcal{T},\mathcal{T}) \in \mathbb{R}^{r\times r}$ denotes the probability transition matrix between states in $\mathcal{T}_{\pi^*}$. The zero matrix $\mathbf{0}_{\sum_{i=1}^m N_i \times r}$ indicates the states in which recurrent classes have zero probability transiting to $\mathcal{T}_{\pi^*}$. The $\boldsymbol{P}_{\pi^*}^{\text{tr}} = [P_{\pi^*}^{\text{tr}_1} \ldots P_{\pi^*}^{\text{tr}_m}] \in \mathbb{R}^{r\times \sum_{i=1}^m N_i}$ is a probability transition matrix where $P_{\pi^*}^{\text{tr}_i} \in \mathbb{R}^{r\times N_i}$ represents the probability of transiting from a transient state in $\mathcal{T}_{\pi^*}$ to the states of $\mathcal{R}_{\pi^*}^i$. The $\boldsymbol{P}_{\pi^*}(\mathcal{R},\mathcal{R})$ is a diagonal block matrix, where the ith block is an $N_i \times N_i$ matrix containing transition probabilities between states within $\mathcal{R}_{\pi^*}^i$. Note that $\boldsymbol{P}_{\pi^*}(\mathcal{R},\mathcal{R})$ is a stochastic matrix since each block matrix is a stochastic matrix [33]. The rewards $\boldsymbol{W}$ and $\boldsymbol{O}_{\pi^*}$ can also be partitioned to $\boldsymbol{W}^{\text{tr}}$, $\boldsymbol{W}^{\text{rec}}$, $\boldsymbol{O}_{\pi^*}^{\text{tr}}$, and $\boldsymbol{O}_{\pi^*}^{\text{rec}}$ based on transient and recurrent classes, respectively.

Infeasibility of Case 1: First, consider a policy $\pi^* \in \Pi_1$, which indicates that there exists some j such that $K_{pi} \cap \mathcal{R}_{\pi^*}^j = \emptyset$. The utilities of recurrent states are obtained from (8) as

$$\boldsymbol{U}_{\pi^*}^{\text{rec}} = \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\pi^*}^n (\mathcal{R},\mathcal{R}) \left(\boldsymbol{W}^{\text{rec}} + \beta \boldsymbol{O}_{\pi^*}^{\text{rec}}\right). \tag{9}$$

Consider a state $s_p \in \mathcal{R}_{\pi^*}^j$ and let $\boldsymbol{P}_{\pi^*}^{s_p R_j}$ denote a row vector of $\boldsymbol{P}_{\pi^*}^n(\mathcal{R},\mathcal{R})$ that contains the transition probabilities from s_p to the states in the same recurrent class $\mathcal{R}_{\pi^*}^j$ in n steps. The utility of s_p is then obtained from (9) as

$$U_{\pi^*}^{\text{rec}}(s_p) = \sum_{n=0}^{\infty} \gamma^n \left[\mathbf{0}_{k_1}^T \ \boldsymbol{P}_{\pi^*}^{s_p R_j} \ \mathbf{0}_{k_2}^T\right] \left(\boldsymbol{W}^{\text{rec}} + \beta \boldsymbol{O}_{\pi^*}^{\text{rec}}\right) \tag{10}$$

where $k_1 = \sum_{i=1}^{j-1} N_i$, $k_2 = \sum_{i=j+1}^{m} N_i$, and $\mathbf{0}_{k_1}^T$ and $\mathbf{0}_{k_2}^T$ are vectors of zeros with dimension k_1 and k_2, respectively. Since $K_{pi} \cap \mathcal{R}_{\pi^*}^j = \emptyset$, based on Definition 4, the rewards of states in $\mathcal{R}_{\pi^*}^j$ are nonpositive. That is, the entries in $\boldsymbol{W}^{\text{rec}}$ corresponding to the states in $\mathcal{R}_{\pi^*}^j$ are nonpositive. Given that the entries in $\boldsymbol{O}_{\pi^*}^{\text{rec}}$ are all nonpositive and $\beta > 0$, (10) indicates that $U_{\pi^*}^{\text{rec}}(s_p) \le 0$ under the optimal policy π^*.

To show that Case 1 cannot be true, the following analysis will show that $U_{\bar{\pi}}^{\text{rec}}(s_p) > U_{\pi^*}^{\text{rec}}(s_p)$, which contradicts the optimal policy π^*. Based on the type of state s_p, two subcases are discussed for policy $\bar{\pi}$.

Subcase 1: If $s_p \in \mathcal{R}_{\bar{\pi}}^j$, one has $\mathcal{R}_{\bar{\pi}}^j \cap K_{pi} \neq \emptyset$ since $\bar{\pi}$ is a policy that satisfies the acceptance condition of $\mathcal{P}$. Thus, there exists at least one $s_K \in K_{pi}$ such that $s_K \in \mathcal{R}_{\bar{\pi}}^j$. In addition, since $L_{pi} \cap \mathcal{R}_{\bar{\pi}}^j = \emptyset$, the entries in $\boldsymbol{W}^{\text{rec}}$ corresponding to the recurrent states in $\mathcal{R}_{\bar{\pi}}^j$ have nonnegative rewards by Definition 4. Note that at minimum there is only one state $s_K \in K_{pi}$ in $\mathcal{R}_{\bar{\pi}}^j$ with reward $w_K > 0$, while the other rewards for recurrent states in $\mathcal{R}_{\bar{\pi}}^j$ are 0. Thus, considering the case of $\bar{\pi}$ in (10), $U_{\bar{\pi}}^{\text{rec}}(s_p)$ can be lower bounded as

$$U_{\bar{\pi}}^{\text{rec}}(s_p) \ge \sum_{n=0}^{\infty} \gamma^n \left(P_{\bar{\pi}}^{s_p s_K} w_K + \beta \boldsymbol{P}_{\bar{\pi}}^{s_p R_j} \boldsymbol{V}_{\bar{\pi}}^{\text{rec}} \mathbf{1}_{N_j}\right) \tag{11}$$

where $P_{\bar{\pi}}^{s_p s_K}$ is the transition probability from s_p to s_K in n steps, and $\boldsymbol{V}_{\bar{\pi}}^{\text{rec}} \in \mathbb{R}^{N_j \times N_j}$ represents the violation cost of states in $\mathcal{R}_{\bar{\pi}}^j$. Since s_p and s_K are recurrent states, there always exists a lower bound $\underline{P}_{\bar{\pi}}^{s_p s_K} \in \mathbb{R}^+$ of the transition probability $P_{\bar{\pi}}^{s_p s_K}$ in (11). Given that $\underline{P}_{\bar{\pi}}^{s_p s_K}, \beta, N_j$ are all positive, the positive reward w_K can be selected to large enough that

$$\underline{P}_{\bar{\pi}}^{s_p s_K} w_K + \beta N_j^2 \underline{V}_{\bar{\pi}}^{\text{rec}} > 0 \tag{12}$$

where $\underline{V}_{\bar{\pi}}^{\text{rec}} \in \mathbb{R}^-$ represents the minimal en*t*ry in $\boldsymbol{V}_{\bar{\pi}}^{\text{rec}}$. If (12) holds, $U_{\bar{\pi}}^{\text{rec}}(s_p) > 0$, leading to the contradiction $U_{\bar{\pi}}^{\text{rec}}(s_p) > U_{\pi^*}^{\text{rec}}(s_p)$. Hence, s_p cannot be in a recurrent set.

Subcase 2: If $s_p \in \mathcal{T}_{\bar{\pi}}$, at minimum all transient states of $\mathcal{T}_{\bar{\pi}}$ will have negative rewards, i.e., $\boldsymbol{W}^{\text{tr}} = [w_L \ldots w_L]^T \in \mathbb{R}^r$ with $w_L < 0$, and only one state $s_K \in K_{pi}$ is in the recurrent class with positive reward w_K.

As demonstrated in [33], for a transient state s, there always exists an upper bound $\Delta < \infty$ such that $\sum_{n=0}^{\infty} p^n(s,s) < \Delta$, where $p^n(s,s)$ denotes the probability of returning from a transient state s to itself in n time steps. For a recurrent state s, it is always true that

$$\sum_{n=0}^{\infty} \gamma^n p^n(s,s) > \frac{1}{1-\gamma^{\overline{n}}} \bar{p} \tag{13}$$

where there exists $\overline{n}$ such that $p^{\overline{n}}(s,s)$ is nonzero and can be lower bounded by $\bar{p}$ [33].

From (8), one has

$$\begin{aligned} \boldsymbol{U}_{\bar{\pi}}^{\text{tr}} > & \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\pi}}^n (\mathcal{T},\mathcal{T}) \left(\boldsymbol{W}^{\text{tr}} + \beta \boldsymbol{O}_{\bar{\pi}}^{\text{tr}}\right) \\ & + \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\pi}}^{\text{tr}} \boldsymbol{P}_{\bar{\pi}}^n (\mathcal{R},\mathcal{R}) \left(\boldsymbol{W}^{\text{rec}} + \beta \boldsymbol{V}_{\bar{\pi}}^{\text{rec}} \mathbf{1}_{\tilde{N}}\right). \end{aligned} \tag{14}$$

Let $\max(\cdot)$ and $\min(\cdot)$ represent the maximum and minimum entry of an input vector, respectively. Consider the lower bound $\underline{V}_{\bar{\pi}}^{\text{tr}} = \min(\boldsymbol{V}_{\bar{\pi}}^{\text{tr}} \mathbf{1}_r)$ and the upper bound $\bar{m} = \{\max(\overline{M}) | \overline{M} < \boldsymbol{P}_{\bar{\pi}}^{\text{tr}} \bar{\boldsymbol{P}} (\boldsymbol{W}^{\text{rec}} + \beta \boldsymbol{V}_{\bar{\pi}}^{\text{rec}} \mathbf{1}_{\tilde{N}})\}$, where $\tilde{N} = \sum_{j=1}^m N_j$ and $\bar{\boldsymbol{P}}$ is a block matrix whose nonzero entries are derived similar to the $\bar{p}$ in (13). Using the fact that $w_L < 0$ and $\sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\pi}}^n (\mathcal{T},\mathcal{T}) \le \Delta \mathbf{1}_{r\times r}$ [33], where $\mathbf{1}_{r\times r}$ is a $r \times r$ matrix of all ones, the utility $U_{\bar{\pi}}^{\text{tr}}(s_p)$ can be lower bounded from (13) and (14) as $U_{\bar{\pi}}^{\text{tr}}(s_p) > \Delta r (w_L + \beta \underline{V}_{\bar{\pi}}^{\text{tr}}) + \frac{1}{1-\gamma^{\overline{n}}} \bar{m}$. Since $U_{\pi^*}^{\text{rec}}(s_p) < 0$, the contradiction $U_{\bar{\pi}}^{\text{tr}}(s_p) > U_{\pi^*}^{\text{rec}}(s_p)$ will be achieved if

$$\Delta r \left(w_L + \beta \underline{V}_{\bar{\pi}}^{\text{tr}}\right) + \frac{1}{1-\gamma^{\overline{n}}} \bar{m} > 0 \tag{15}$$

which indicates that Case 1 cannot be true. Since $\Delta r (w_L + \beta \underline{V}_{\bar{\pi}}^{\text{tr}}) < 0$, it needs $\bar{m} > 0$ so that the inequality in (15) holds when γ is sufficiently close to 1. Note that when there is only one $s_K \in \mathcal{R}_{\bar{\pi}}^j$ and $s_K \in K_{Pi}$, $\bar{m} > 0$ if

$$w_K + \beta \tilde{N} \underline{V}_{\bar{\pi}}^{\text{rec}} > 0 \tag{16}$$

where $\underline{V}^{\text{rec}}_{\bar{\pi}} = \min(\boldsymbol{V}^{\text{rec}}_{\bar{\pi}} \mathbf{1}_{\tilde{N}})$. In summary, Case 1 cannot be true, if w_L, w_K, β, and γ are properly designed such that (12), (15), and (16) are satisfied.

Infeasibility of Case 2: For $\pi^* \in \Pi_2$, consider a state $s_p \in \mathcal{R}^i_{\pi^*} \cap L_{pi}$ for some i. The following analysis starts by deriving an upper bound of $U^{\text{rec}}_{\pi^*}(s_p)$, which will then be proven to be less than $U^{\text{rec}}_{\bar{\pi}}(s_p)$, leading to the contradiction that π^* is an optimal policy.

Consider that s_p has the negative reward w_L and all other states in $\mathcal{R}^i_{\pi^*}$ have positive reward w_K, which yields the upper bound from (9) as

$$\begin{aligned} U^{\text{rec}}_{\pi^*}(s_p) &\leq \sum_{n=0}^{\infty} \gamma^n (N_i - 1) w_K + \sum_{n=0}^{\infty} \gamma^n \beta P^{s_p s_p}_{\bar{\pi},n} w_L \\ &\leq \frac{(N_i - 1) w_K}{1-\gamma} + \beta w_L \frac{\bar{p}^{s_p s_p}_{\bar{\pi},n}}{1-\gamma} \triangleq \bar{U}^{\text{rec}}_{\pi^*}(s_p) \end{aligned} \tag{17}$$

where N_i is the number of states in $\mathcal{R}^i_{\pi^*}$, $P^{s_p s_p}_{\bar{\pi},n}$ represents the probability of returning from s_p to itself in n steps, $\bar{p}^{s_p s_p}_{\bar{\pi},n}$ is an upper bounder of $P^{s_p s_p}_{\bar{\pi},n}$, and the fact that $\boldsymbol{O}^{\text{rec}}_{\pi^*} \leq 0$ is used.

In addition, since $s_p \in L_{pi}$, the policy $\bar{\pi}$ indicates that s_p is in the transient class $\mathcal{T}_{\bar{\pi}}$. If (16) is satisfied, the lower bound of $U^{\text{tr}}_{\bar{\pi}}(s_p)$ is derived from (14) as

$$U^{\text{tr}}_{\bar{\pi}}(s_p) > \triangleq \Delta r \left(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}\right) \triangleq \underline{U}^{\text{tr}}_{\bar{\pi}}(s_p). \tag{18}$$

Based on (17) and (18), if w_L, w_K, β, and γ are properly designed such that

$$\Delta r \left(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}\right) > \frac{(N_i - 1) w_K}{1-\gamma} + \beta w_L \frac{\bar{p}^{s_p s_p}_{\bar{\pi},n}}{1-\gamma} \tag{19}$$

the contradiction $U^{\text{tr}}_{\bar{\pi}}(s_p) > U^{\text{rec}}_{\pi^*}(s_p)$ achieves, which indicates that Case 2 cannot be true.

Base on the analysis of Cases 1 and 2, the optimal policy π^* that optimizes (4) also satisfies the acceptance condition of $\mathcal{P}$, provided that (12), (15), (16), and (19) are satisfied.

The following shows how w_L, w_K, β, and γ can be determined to satisfy (12), (15), (16), and (19). First, given a fixed w_K, β can be determined such that (12) and (16) are satisfied. Based on the determined w_K and β and the conditions in (15) and (19), w_L and γ need to be selected to satisfy $\Delta r(1-\gamma^{\overline{n}})(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}) + \bar{m} > 0$ and $(N_i - 1)w_K + \beta \bar{p}^{s_p s_p}_{\bar{\pi},n} w_L - (1-\gamma)\Delta r(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}) < 0$. The reward w_L can then be determined first by solving $(N_i - 1)w_K + \beta \bar{p} w_L < -\epsilon$, where $0 < \epsilon < \bar{m}$, and γ can be determined by solving

$$\begin{aligned} \max \Big\{ &-\Delta r \left(1 - \gamma^{\overline{n}}\right) \left(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}\right) \\ &- (1-\gamma) \Delta r \left(w_L + \beta \underline{V}^{\text{tr}}_{\bar{\pi}}\right) \Big\} < \epsilon. \end{aligned}$$

References

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

[2] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 48–61, Feb. 2010.

[3] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *Proc. IEEE Conf. Decis. Control*, 2011, pp. 532–538.

[4] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[5] K. Kim and G. E. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2012, pp. 265–271.

[6] K. Kim, G. E. Fainekos, and S. Sankaranarayanan, "On the revision problem of specification automata," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 5171–5176.

[7] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. Int. Conf. Hybrid Syst., Comput. Control*, 2013, pp. 1–10.

[8] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[9] I. Cizelj and C. Belta, "Control of noisy differential-drive vehicles from time-bounded temporal logic specifications," *Int. J. Robot. Res.*, vol. 33, no. 8, pp. 1112–1129, 2014.

[10] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental controller synthesis in probabilistic environments with temporal logic constraints," *Int. J. Robot. Res*, vol. 33, no. 8, pp. 1130–1144, 2014.

[11] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 396–409, Apr. 2012.

[12] P. Nuzzo, J. Li, A. L. Sangiovanni-Vincentelli, Y. Xi, and D. Li, "Stochastic assume-guarantee contracts for cyber-physical system design," *ACM Trans. Embed Comput Syst.*, vol. 18, no. 1, pp. 1–26, 2019.

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

[14] T. Brázdil *et al.*, "Verification of Markov decision processes using learning algorithms," in *Proc. Int. Symp. Autom. Tech. Verif. Anal. Springer*, 2014, pp. 98–114.

[15] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Proc. IEEE Conf. Decis. Control*, 2016, pp. 6565–6570.

[16] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *Proc. IEEE Int. Conf. Intell. Robot. Syst.*, 2017, pp. 3834–3839.

[17] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *Proc. IEEE Conf. Decis. Control.*, 2014, pp. 1091–1096.

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

[19] J. Wang, X. Ding, M. Lahijanian, I. C. Paschalidis, and C. A. Belta, "Temporal logic motion control using actor–critic methods," *Int. J. Robot. Res*, vol. 34, no. 10, pp. 1329–1344, 2015.

[20] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, and M. M. Zavlanos, "Reduced variance deep reinforcement learning with temporal logic specifications," in *Proc. ACM/IEEE Int. Conf. Cyber-Physical Syst.*, 2019, pp. 237–248.

[21] M. Hasanbeig, A. Abate, and D. Kroening, "Certified reinforcement learning with logic guidance," 2019, arXiv: 1902.00778.

[22] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2019, pp. 395–412.

[23] 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," 2019, arXiv: 1909.05304, 2019.

[24] E. M. Wolff, U. Topcu, and R. M. Murray, "Robust control of uncertain Markov decision processes with temporal logic specifications," in *Proc. IEEE Conf. Decis. Control*, 2012, pp. 3372–3379.

[25] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4051–4066, Dec. 2018.

[26] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theor. Comput. Sci.*, vol. 363, no. 2, pp. 182–195, 2006.

[27] E. Gradel and W. Thomas, *Automata, Logics, and Infinite Games: A Guide to Current Research*, vol. 2500. Berlin, Germany: Springer, 2002.

[28] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Kuala Lumpur, Malaysia: Pearson, 2016.

[29] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, no. 2–3, pp. 209–232, 2002.

[30] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, 1992.

[31] J. Klein, "LTL2dstar-LTL to deterministic Streett and Rabin automata," 2007. [Online]. Available: https://www.ltl2dstar.de

[32] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Autom. Control*, vol. 64, no. 7, pp. 2737–2752, Jul. 2019.

[33] R. Durrett and R. Durrett, *Essentials of Stochastic Processes*, vol. 1. Berlin, Germany: Springer, 1999.