Reinforcement Learning Based Temporal Logic Control with Maximum Probabilistic Satisfaction

Mingyu Cai[1], Shaoping Xiao[1], Baoluo Li[2], Zhiliang Li[2], and Zhen Kan[2]

***Abstract*—This paper presents a model-free reinforcement learning (RL) algorithm to synthesize a control policy that maximizes the satisfaction probability of complex tasks, which are expressed by linear temporal logic (LTL) specifications. Due to the consideration of environment and motion uncertainties, we model the robot motion as a probabilistic labeled Markov decision process (PL-MDP) with unknown transition probabilities and probabilistic labeling functions. The LTL task specification is converted to a limit deterministic generalized Büchi automaton (LDGBA) with several accepting sets to maintain dense rewards during learning. The novelty of applying LDGBA is to construct an embedded LDGBA (E-LDGBA) by designing a synchronous tracking-frontier function, which enables the record of non-visited accepting sets of LDGBA at each round of the repeated visiting pattern, to overcome the difficulties of directly applying conventional LDGBA. With appropriate dependent reward and discount functions, rigorous analysis shows that any method, which optimizes the expected discount return of the RL-based approach, is guaranteed to find the optimal policy to maximize the satisfaction probability of the LTL specifications. A model-free RL-based motion planning strategy is developed to generate the optimal policy in this paper. The effectiveness of the RL-based control synthesis is demonstrated via simulation and experimental results.**

I. INTRODUCTION

Temporal logic has rich expressivity in describing complex high-level tasks beyond traditional go-to-goal navigation for robotic systems [1]–[3]. Due to a variety of uncertainties (e.g., transition probabilities and environment uncertainties), the robot's probabilistic motion is often modeled by a Markov decision process (MDP). Growing research has been devoted to investigating the motion planning of an MDP satisfying linear temporal logic (LTL) constraints. With the assumption of full knowledge of MDP, one common objective is to maximize the probability of accomplishing tasks [4]–[7]. Yet, it raises some challenges when the MDP is not fully known *a priori*. Hence, this work focuses on motion planning that maximizes the satisfaction probability of given tasks over an uncertain MDP.

Reinforcement learning (RL) is a widely-used approach for sequential decision-making problems [8]. When integrating with LTL specifications, model-based RL has been employed in [9]–[11] to generate policies to satisfy LTL tasks by learning unknown parameters of the MDP. However, there is a scalability issue due to the high need of memory to store the learned models. On the other hand, model-free RL generates policies to satisfy LTL formulas by designing appropriate accepting rewards to optimize Q values [12]–[19]. In [12], the robustness degree of truncated linear temporal logic (TLTL) was used as reward to facilitate learning. The deterministic finite automaton (DFA) was applied as reward machines in [20]–[22]. However, only finite horizon motion planning was considered in [12], [20]–[22].

Related works: This work extends previous research to tasks over infinite horizon, where finite horizon motion planning can be regarded as a special case of the infinite horizon setting. Along this line of research, in [13] and [14], LTL constraints were translated to Deterministic Rabin Automata (DRA), which may fail to find desired policies as discussed in [15]. Instead of using DRA, limit-deterministic Buchi automaton (LDBA) was employed in [15] and [16] without considering the workspace uncertainties. Moreover, since LDBA in works [15] and [16] has only one accepting set, it might lead to sparse reward issues during learning. In [17], limit-deterministic generalized Buchi automaton (LDGBA) was used, and a frontier function of rewards was designed to facilitate learning by assigning positive rewards to the accepting sets. However, directly applying the LDGBA as in [17] may fail to satisfy the LTL specification when applying the deterministic policy and such a drawback was also presented in [18]. Such an issue may be solved via selecting these actions based on the uniform distribution when applying the tubular RL method. However, the application of deterministic policies is crucial in practice especially for the continuous space, since many widely-applied deep RL methods adopt the actor-critic architecture e.g., as deep deterministic policy gradients (DDPG) and trust region policy optimization (TRPO) for high dimensional analysis. The works of [18] and [19] overcome this issue by designing binary-valued vectors and Boolean vectors, respectively. However, both [18] and [19] cannot guarantee the maximum probability of task satisfaction.

Contributions: Our framework studies motion planning that maximizes the probability of satisfying pre-specified LTL tasks in stochastic systems. Considering both motion and environment uncertainties, the robotic system is modeled as a probabilistic labeled Markov decision process (PL-MDP) with unknown transition probabilities and probabilistic labels. In this work, a synchronous tracking-frontier function is designed to construct an embedded LDGBA (E-LDGBA) from convention LDGBA, which is capable of recording non-visited accepting sets and incorporating deterministic policies. We construct the embedded product MDP (EP-MDP) between E-LDGBA and PL-MDP, and propose a new expected return by applying the reward and discount functions of [16]. Rigorous analysis shows that our framework is guaranteed to find the

[1]Department of Mechanical Engineering, The University of Iowa, Iowa City, IA, USA. [2]Department of Automation, University of Science and Technology of China, Hefei, Anhui, China.

optimal policy that maximizes the probability of satisfying LTL specifications.

II. PRELIMINARIES

A. Probabilistic Labeled MDP

A PL-MDP is a tuple $\mathcal{M}=(S,A,p_S,(s_0,l_0),\Pi,L,p_L)$, where S is a finite state space, A is a finite action space, $p_S: S\times A\times S\to[0,1]$ is the transition probability function, Π is a set of atomic propositions, and $L:S\to 2^{\Pi}$ is a labeling function. The pair (s_0,l_0) denotes an initial state $s_0\in S$ and an initial label $l_0\in L(s_0)$. The function $p_L(s,l)$ denotes the probability of $l\subseteq L(s)$ associated with $s\in S$ satisfying $\sum_{l\in L(s)}p_L(s,l)=1,\forall s\in S$. For simplicity, let $A(s)$ denote the set of actions that can be taken in state s. The transition probability p_S captures the motion uncertainties of the agent while the labeling probability p_L captures the environment uncertainties. It is assumed that p_S and p_L are not known *a priori*, and the agent can only observe its current state and the associated labels. Note that the standard MDP model can be regarded as a special case of PL-MDP with the deterministic label function.

Let ξ be a deterministic action function such that $\xi: S\to A$ maps a state $s\in S$ to an action in $A(s)$. The PL-MDP $\mathcal{M}$ evolves by taking an action ξ_i at each stage i, and thus the control policy $\boldsymbol{\xi}=\xi_0\xi_1\ldots$ is a sequence of actions, which yields a path $\boldsymbol{s}=s_0s_1s_2\ldots$ over $\mathcal{M}$ with $p_S(s_i,a_i,s_{i+1})>0$ for all i. If $\xi_i=\xi$ for all i, then $\boldsymbol{\xi}$ is called a stationary policy. The control policy $\boldsymbol{\xi}$ is memoryless if each ξ_i only depends on its current state, and $\boldsymbol{\xi}$ is called a finite memory policy if ξ_i depends on its past states.

Let $\Lambda: S\times A\times S\to\mathbb{R}$ denote a reward function. Given a discount function $\gamma: S\times A\times S\to\mathbb{R}$, the expected discounted return under policy $\boldsymbol{\xi}$ starting from $s\in S$ is defined as

$$U^{\boldsymbol{\xi}}(s)=\mathbb{E}^{\boldsymbol{\xi}}\left[\sum_{i=0}^{\infty}\gamma^i(s_i,a_i,s_{i+1})\cdot\Lambda(s_i,a_i,s_{i+1})\,|s_0=s\right].$$

An optimal policy $\boldsymbol{\xi}^*$ that maximizes the expected return for each state $s\in S$ is defined as

$$\boldsymbol{\xi}^*=\arg\max_{\boldsymbol{\xi}}U^{\boldsymbol{\xi}}(s).$$

The function $U^{\boldsymbol{\xi}}(s)$ is often referred to as the value function under policy $\boldsymbol{\xi}$. If the MDP is not fully known, but the state and action spaces are countably finite, tabular approaches are usually employed [8].

B. LTL and Limit-Deterministic Generalized Büchi Automaton

An LTL is built on atomic propositions, Boolean operators, and temporal operators [1]. Given an LTL that specifies the missions, the satisfaction of the LTL can be evaluated by an LDGBA [23]. Before defining LDGBA, we first introduce the generalized Büchi automaton (GBA).

Definition 1. A GBA is a tuple $\mathcal{A}=(Q,\Sigma,\delta,q_0,F)$, where Q is a finite set of states; $\Sigma=2^{\Pi}$ is a finite alphabet, $\delta: Q\times\Sigma\to 2^Q$ is the transition function, $q_0\in Q$ is an initial state, and $F=\{F_1,F_2,\ldots,F_f\}$ is a set of accepting sets with $F_i\subseteq Q$, $\forall i\in\{1,\ldots f\}$.

Denote by $\boldsymbol{q}=q_0q_1\ldots$ a run of a GBA, where $q_i\in Q$, $i=0,1,\ldots$. The run $\boldsymbol{q}$ is accepted by the GBA, if it satisfies the generalized Büchi acceptance condition, i.e., $\inf(\boldsymbol{q})\cap F_i\neq\emptyset$, $\forall i\in\{1,\ldots f\}$, where $\inf(\boldsymbol{q})$ denotes the infinitely part of $\boldsymbol{q}$.

Definition 2. A GBA is an LDGBA if the transition function δ is extended to $Q\times(\Sigma\cup\{\epsilon\})\to 2^Q$, and the state set Q is partitioned into a deterministic set Q_D and a non-deterministic set Q_N, i.e., $Q_D\cup Q_N=Q$ and $Q_D\cap Q_N=\emptyset$, where

- the state transitions in Q_D are total and restricted within it, i.e., $\left|\delta(q,\alpha)\right|=1$ and $\delta(q,\alpha)\subseteq Q_D$ for every state $q\in Q_D$ and $\alpha\in\Sigma$,
- the ϵ-transition is not allowed in the deterministic set, i.e., for any $q\in Q_D$, $\delta(q,\epsilon)=\emptyset$, and
- the accepting sets are only in the deterministic set, i.e., $F_i\subseteq Q_D$ for every $F_i\in F$.

In Definition 2, the ϵ-transitions are only defined for state transitions from Q_N to Q_D, which do not consume the input alphabet. To convert an LTL formula to an LDGBA, readers are referred to Owl [24].

III. PROBLEM STATEMENTS

The task specification to be performed by the robot is described by an LTL formula ϕ over Π. Given Task ϕ, the PL-MDP $\mathcal{M}$, and a policy $\boldsymbol{\xi}=\xi_0\xi_1\ldots$, the induced path $\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}=s_0\ldots s_is_{i+1}\ldots$ over $\mathcal{M}$ satisfies $s_{i+1}\in\left\{s\in S\middle|p_S(s_i,a_i,s)>0\right\}$. Let $L\left(\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}\right)=l_0l_1\ldots$ be the sequence of labels associated with $\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}$ such that $l_i\in L(s_i)$ and 1 $p_L(s_i,l_i)>0$. Denote by $L\left(\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}\right)\models\phi$ if the induced trace $\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}$ satisfies ϕ. The probabilistic satisfaction under the policy ξ from an initial state s_0 can be defined as

$$\Pr{}_M^{\boldsymbol{\xi}}(\phi)=\Pr{}_M^{\boldsymbol{\xi}}\left(L\left(\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}\right)\models\phi\middle|\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}\in\boldsymbol{S}_{\infty}^{\boldsymbol{\xi}}\middle|\right),\qquad(1)$$

where $\boldsymbol{S}_{\infty}^{\boldsymbol{\xi}}$ is a set of admissible paths from the initial state s_0 under the policy $\boldsymbol{\xi}$.

Assumption 1. *It is assumed that there exists at least one policy such that the induced traces satisfy task ϕ with non-zero probability.*

Assumption 1 is a mild assumption and widely employed in the literature (cf. [9], [15], [16]), which indicates that the LTL task can be satisfied with nonzero probability. Consequently, the following problem is considered.

Problem 1. Given an LTL-specified task ϕ and a PL-MDP $\mathcal{M}$ with unknown transition probabilities (i.e., motion uncertainties) and an unknown probabilistic label function (i.e., workspace uncertainties), the objective is to find the desired policy $\boldsymbol{\xi}^*$ that maximizes the satisfaction probability, i.e., $\boldsymbol{\xi}^*=\arg\max_{\boldsymbol{\xi}}\Pr{}_M^{\boldsymbol{\xi}}(\phi)$, by interacting with the environment.

In order to find the desired policy in PL-MDP $\mathcal{M}$ to satisfy the user-specified LTL formula ϕ, we can construct the standard product MDP between $\mathcal{M}$ and the LDGBA of ϕ as described in [1], [25]. Then, the problem becomes finding the

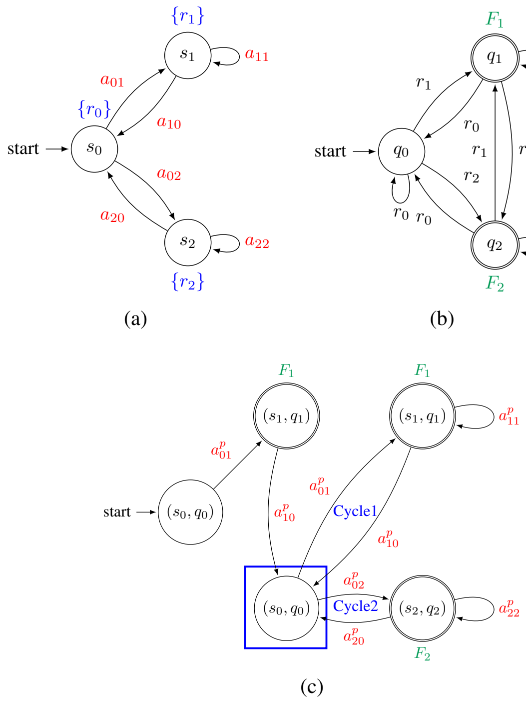

Figure 1: (a) PL-MDP model with the deterministic label function. (b) LDGBA of LTL formula φ_e. (c) The standard product MDP.

policy that satisfies the accepting condition of the standard product MDP with maximum probability. However, when considering deterministic policies, directly applying LDGBA [25] may fail to satisfy the LTL specifications, because there do not exist deterministic policies to select several actions with the same optimal expected return at one state. To illustrate this issue, Example 1 is provided.

Example 1. Here is an example to demonstrate why the LDGBA does not work in some cases for deterministic policies. Fig. 1 (a) shows a special case of the PL-MDP model that deterministically labels each state, in which there are three states s_0, s_1 and s_2 associated with three labels r_0, r_1, and r_2, respectively. The initial state of PL-MDP is s_0. The LTL specification of the PL-MDP is $\varphi_e = (\square\Diamond r_1) \wedge (\square\Diamond r_2)$, which requires the agent starting from s_0 labeled with r_0 to repetitively visit the states with labels r_1 and r_2. Fig. 1 (b) shows the corresponding LDGBA of φ_e with two accepting sets $F = \{\{q_1\}, \{q_2\}\}$. Fig. 1 (c) illustrates the resulted standard product MDP. By Def. 2, the policy that satisfies φ_e should enforce the repetitive trajectories, i.e., Cycles 1 and 2 in Fig. 1 (c). However, there exists no deterministic policy that can periodically select two actions a^P_{01} and a^P_{02} at state (s_0, q_0) (marked with a blue rectangle) in Fig. 1 (c). As a result, applying the standard product MDP cannot generate a pure deterministic optimal policy to complete Task φ_e.

Motivated by the challenge illustrated above, this paper proposes a novel automaton structure based on LDGBA to generate a deterministic policy to solve Problem 1. We also verify that the designed automaton structure accepts the same languages as LDGBA, which will not influence the optimal convergence of the RL agent.

Algorithm 1 Procedure of E-LDGBA

```
procedure INPUT: (LDGBA 𝒜, f_V, T and length L)
    Output:  A valid run q̄ with length L in 𝒜̄_φ
  set T = F and count = 1
  set q̄_cur = (q_0, T) and q̄ = (q_cur)
  set q_cur = q
  while count ≤ L do
      q_next = δ(q_cur, α)
      q̄_cur ← (q_next, T)
      check if q̄_cur is an accepting state
      T = f_V (q_next, T)
      add state q̄_cur to q̄
      count + + and q_cur ← q_next
  end while
end procedure
```

IV. Automaton Analysis

To solve Problem 1, Section IV-A first presents how the LDGBA in Definition 2 can be extended to an E-LDGBA, which keeps track of non-visited accepting sets and accepts the same language as the LDGBA. Section IV-B presents the construction of a EP-MDP between a PL-MDP and an E-LDGBA. The benefits of incorporating E-LDGBA are discussed in Section IV-C.

A. E-LDGBA

Given an LDGBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, inspired by [17], a tracking-frontier set T is designed to keep track of non-visited accepting sets. Particularly, T is initialized as F, which is then updated based on

$$f_V(q, T) = \begin{cases} T \setminus F_j, & \text{if } q \in F_j \text{ and } F_j \in T, \\ F \setminus F_j, & \text{if } q \in F_j \text{ and } T = \emptyset, \\ T, & \text{otherwise.} \end{cases} \tag{2}$$

Once an accepting set F_j is visited, it will be removed from T. If T becomes empty, it will be reset as $F \setminus F_j$. Since the acceptance condition of LDGBA requires to infinitely visit all accepting sets, we call it one round if all accepting sets have been visited (i.e., a round ends if T becomes empty). If a state q belongs to multiple sets of T, all of these sets should be removed from T. Based on (2), the E-LDGBA is constructed as follows.

Definition 3 (Embedded LDGBA)**.** Given an LDGBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, its corresponding E-LDGBA is denoted by $\overline{\mathcal{A}} = (\overline{Q}, \Sigma, \overline{\delta}, \overline{q_0}, \overline{F}, f_V, T)$ where T is initially set as $T = F$; $\overline{Q} = Q \times 2^Q$ is the set of augmented states e.g., $\overline{q} = (q, T)$; The finite alphabet Σ is the same as the LDGBA; The transition $\overline{\delta}\colon \overline{Q} \times (\Sigma \cup \{\epsilon\}) \to 2^{\overline{Q}}$ is defined as $\overline{q'} = \overline{\delta}(\overline{q}, \overline{\sigma})$ with $\overline{\sigma} \in (\Sigma \cup \{\epsilon\})$, e.g., $\overline{q} = (q, T)$ and $\overline{q'} = (q', T)$, and it satisfies two conditions: 1) $q' = \delta(q, \overline{\sigma})$, and 2) T is synchronously updated as $T = f_V(q', T)$ after transition $\overline{q'} = \overline{\delta}(\overline{q}, \alpha)$; $\overline{F} = \{\overline{F_1}, \overline{F_2} \ldots \overline{F_f}\}$ where $\overline{F_j} = \{(q, T) \in \overline{Q} \big| q \in F_j \wedge F_j \subseteq T\}$, $j = 1, \ldots f$, is a set of accepting states.

In Definition 3, we abuse the tuple structure since the frontier set T is synchronously updated after each transition, and each state of E-LDGBA is augmented with the tracking-frontier set T at every time-step via one-hot encoding. The accepting state is determined based on the current automaton state, and frontier set T that is synchronously updated after each transition. Such property is the innovation of E-LDGBA, which encourages all accepting sets to be visited in each round. In the following analysis, we will use $\overline{\mathcal{A}}_{\phi}$ and $\mathcal{A}_{\phi}$ to denote the E-LDGBA and LDGBA, respectively, corresponding to an LTL formula ϕ. Algorithm 1 shows the procedure of obtaining a valid run $\overline{\boldsymbol{q}}$ over an E-LDBGA $\overline{\mathcal{A}}_{\phi}$.

Given $\overline{\mathcal{A}}_{\phi}$ and $\mathcal{A}_{\phi}$ for the same LTL formula, the E-LDGBA $\overline{\mathcal{A}}_{\phi}$ keeps track of unvisited accepting sets of $\mathcal{A}_{\phi}$ by incorporating f_V and T. The T will be reset when all the accepting sets of $\mathcal{A}_{\phi}$ have been visited. Let $\mathcal{L}(\mathcal{A}_{\phi}) \subseteq \Sigma^{\omega}$ and $\mathcal{L}(\overline{\mathcal{A}}_{\phi}) \subseteq \Sigma^{\omega}$ be the accepted language of the $\mathcal{A}_{\phi}$ and $\overline{\mathcal{A}}_{\phi}$ automaton, respectively, with the same alphabet Σ. Based on [1], $\mathcal{L}(\mathcal{A}_{\phi}) \subseteq \Sigma^{\omega}$ is the set of all infinite words accepted by $\mathcal{A}_{\phi}$ that satisfy LTL formula ϕ.

Lemma 1. *For any LTL formula ϕ, we can construct LDGBA $\mathcal{A}_{\phi} = (Q, \Sigma, \delta, q_0, F)$ and E-LDGBA $\overline{\mathcal{A}}_{\phi} = (\overline{Q}, \Sigma, \overline{\delta}, \overline{q_0}, \overline{F}, f_V, T)$. Then it holds that*

$$\mathcal{L}(\overline{\mathcal{A}}_{\phi}) = \mathcal{L}(\mathcal{A}_{\phi}). \tag{3}$$

Proof: We prove (1) by showing that $\mathcal{L}(\overline{\mathcal{A}}_{\phi}) \supseteq \mathcal{L}(\mathcal{A}_{\phi})$ and $\mathcal{L}(\overline{\mathcal{A}}_{\phi}) \subseteq \mathcal{L}(\mathcal{A}_{\phi})$.

Case 1: $\mathcal{L}(\overline{\mathcal{A}}_{\phi}) \supseteq \mathcal{L}(\mathcal{A}_{\phi})$: For any accepting language $\boldsymbol{\omega} = \alpha_0\alpha_1 \ldots \in \mathcal{L}(\mathcal{A}_{\phi})$, there exists a corresponding run $\boldsymbol{r} = q_0\alpha_0 q_1\alpha_1 \ldots$ of $\mathcal{A}_{\phi}$ s.t.

$$\inf(\boldsymbol{r}) \cap F_i \neq \emptyset, \forall i \in \{1, \ldots f\}. \tag{4}$$

For the run $\boldsymbol{r}$, we can construct a sequence $\overline{\boldsymbol{r}} = \overline{q}_0\alpha_0\overline{q}_1\alpha_1 \ldots$ by add each state q with the set T, which is synchronously updated via (2) after each transition. It can be verified that such a run $\overline{\boldsymbol{r}}$ is a valid run of $\overline{\mathcal{A}}_{\phi}$ based on Def. 3. According to (4), since the tracking-frontier set T will be reset once all accepting sets have been visited, it holds $\inf(\overline{\boldsymbol{r}}) \cap \overline{F_i} \neq \emptyset, \forall i \in \{1, \ldots f\}$ s.t. $\boldsymbol{\omega} \in \mathcal{L}(\overline{\mathcal{A}}_{\phi})$.

Case 2: $\mathcal{L}(\overline{\mathcal{A}}_{\phi}) \subseteq \mathcal{L}(\mathcal{A}_{\phi})$: Similarly, for any accepting language $\overline{\boldsymbol{\omega}} = \overline{\alpha}_0\overline{\alpha}_1 \ldots \in \mathcal{L}(\overline{\mathcal{A}}_{\phi})$, there exists a corresponding run $\overline{\boldsymbol{r}} = \overline{q}_0\overline{\alpha}_0\overline{q}_1\overline{\alpha}_1 \ldots$ of $\overline{\mathcal{A}}_{\phi}$ s.t.

$$\inf(\overline{\boldsymbol{r}}) \cap \overline{F_i} \neq \emptyset, \forall i \in \{1, \ldots f\}. \tag{5}$$

For the run $\overline{\boldsymbol{r}}$, we can construct a sequence $\boldsymbol{r} = q_0\overline{\alpha}_0 q_1\overline{\alpha}_1 \ldots$ by projecting each state $\overline{q} = (q, T)$ into q. It can be simply verified that such a run $\boldsymbol{r}$ is a valid run of $\mathcal{A}_{\phi}$ based on Def. 3. According to (5), it holds $\inf(\boldsymbol{r}) \cap F_i \neq \emptyset, \forall i \in \{1, \ldots f\}$ s.t. $\overline{\boldsymbol{\omega}} \in \mathcal{L}(\mathcal{A}_{\phi})$. ∎

Lemma 1 indicates that both E-LDGBA and LDGBA accept the same language. Consequently, E-LDGBA can also be applied to verify the satisfaction of LTL specifications, and incorporating E-LDGBA into RL based model checking will not affect the convergence of optimality.

B. Embedded Product MDP

Definition 4. Given a PL-MDP $\mathcal{M}$ and an E-LDGBA $\overline{\mathcal{A}}_{\phi}$, the embedded product MDP (EP-MDP) is defined as $\mathcal{P} = \mathcal{M} \times \overline{\mathcal{A}}_{\phi} = \left(X, U^{\mathcal{P}}, p^{\mathcal{P}}, x_0, F^{\mathcal{P}}, T, f_V\right)$, where $X = S \times 2^{\Pi} \times \overline{Q}$ is the set of labeled states, i.e., $x = (s, l, q, T) \in X$ with $l \in L(s)$ satisfying $p_L(s, l) > 0$; $U^{\mathcal{P}} = A \cup \{\epsilon\}$ is the set of actions, where the ϵ-transitions are only allowed for transitions from Q_N to Q_D; $x_0 = (s_0, l_0, \overline{q_0})$ is the initial state; $F^{\mathcal{P}} = \left\{F_1^{\mathcal{P}}, F_2^{\mathcal{P}} \ldots F_f^{\mathcal{P}}\right\}$ where $F_j^{\mathcal{P}} = \left\{(s, l, \overline{q}) \in X \middle| \overline{q} \in \overline{F_j}\right\}$, $j = 1, \ldots f$, is the set of accepting states; $p^{\mathcal{P}} : X \times U^{\mathcal{P}} \times X \to [0, 1]$ is transition probability defined as: 1) $p^{\mathcal{P}}\left(x, u^{\mathcal{P}}, x'\right) = p_L(s', l') \cdot p_S(s, a, s')$ if $\overline{\delta}(q, l) = q'$ and $u^{\mathcal{P}} = a \in A(s)$; 2) $p^{\mathcal{P}}\left(x, u^{\mathcal{P}}, x'\right) = 1$ if $u^{\mathcal{P}} \in \{\epsilon\}$, $q' \in \overline{\delta}(q, \epsilon)$, and $(s', l') = (s, l)$; and 3) $p^{\mathcal{P}}\left(x, u^{\mathcal{P}}, x'\right) = 0$ otherwise. After completing each transition $q' = \delta(q, \alpha)$ based on $\overline{\delta}$, T is synchronously updated as $(T) = f_V(q', T)$ by (2).

The EP-MDP $\mathcal{P}$ captures the intersections between all feasible paths over $\mathcal{M}$ and all words accepted to $\overline{\mathcal{A}}_{\phi}$, facilitating the identification of admissible agent motions that satisfy task ϕ. Let $\boldsymbol{\pi}$ denote a policy over $\mathcal{P}$ and denote by $\boldsymbol{x}_{\infty}^{\boldsymbol{\pi}} = x_0 \ldots x_i x_{i+1} \ldots$ the infinite path generated by $\boldsymbol{\pi}$. A path $\boldsymbol{x}_{\infty}^{\boldsymbol{\pi}}$ is accepted if $\inf(\boldsymbol{x}_{\infty}^{\boldsymbol{\pi}}) \cap F_i^{\mathcal{P}} \neq \emptyset$, $\forall i \in \{1, \ldots f\}$. The accepting run $\boldsymbol{x}_{\infty}^{\pi}$ can yield a policy $\boldsymbol{\xi}$ in $\mathcal{M}$ that satisfies ϕ. We denote $\Pr^{\boldsymbol{\pi}}[x \models \mathrm{Acc}_p]$ as the probability of satisfying the acceptance of $\mathcal{P}$ under policy $\boldsymbol{\pi}$, and denote $\Pr_{max}[x \models \mathrm{Acc}_p] = \max_{\boldsymbol{\pi}} \Pr_M^{\boldsymbol{\pi}}(\mathrm{Acc}_p)$.

Consider a sub-EP-MDP $\mathcal{P}'_{(X', U')}$, where $X' \subseteq X$ and $U' \subseteq U^{\mathcal{P}}$. If $\mathcal{P}'_{(X', U')}$ is a maximum end component (MEC) of $\mathcal{P}$ and $X' \cap F_i^{\mathcal{P}} \neq \emptyset, \forall i \in \{1, \ldots f\}$, then $\mathcal{P}'_{(X', U')}$ is called an accepting maximum end component (AMEC) of $\mathcal{P}$. Once a path enters an AMEC, the subsequent path will stay within it by taking restricted actions from U'. There exist policies such that any state $x \in X'$ can be visited infinitely often. As a result, satisfying the task ϕ is equivalent to reaching an AMEC. Moreover, an MEC that does not contain any accepting set is called a rejecting maximum end component (RMEC) and an MEC with only partial accepting sets is called a neutral maximum end component (NMEC) [1]. Consequently, problem 1 can be reformulated as follows.

Problem 2. Given a user-specified LTL task ϕ and the PL-MDP with unknown transition probabilities (i.e., motion uncertainties) and unknown labeling probabilities (i.e., environment uncertainties), the goal is to find a policy $\boldsymbol{\pi}^*$ satisfying the acceptance condition of $\mathcal{P}$ with a maximum probability, i.e., $\Pr^{\boldsymbol{\pi}^*}[x \models \mathrm{Acc}_p] = \Pr_{max}[x \models \mathrm{Acc}_p]$.

C. Properties of EP-MDP

Definition 5. Let $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ denote the Markov chain induced by a policy $\boldsymbol{\pi}$ on $\mathcal{P}$, whose states can be represented by a disjoint union of a transient class $\mathcal{T}_{\boldsymbol{\pi}}$ and n_R closed irreducible recurrent classes $\mathcal{R}_{\boldsymbol{\pi}}^j$, $j \in \{1, \ldots, n_R\}$ [26].

Lemma 2. *Given an EP-MDP $\mathcal{P} = \mathcal{M} \times \overline{\mathcal{A}}_{\phi}$, the recurrent class R_{π}^j of $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$, $\forall j \in \{1, \ldots, n\}$, induced by π satisfies one of the following conditions:*

1) $R_{\boldsymbol{\pi}}^j \cap F_i^{\mathcal{P}} \neq \emptyset, \forall i \in \{1, \ldots f\}$, *or*
2) $R_{\boldsymbol{\pi}}^j \cap F_i^{\mathcal{P}} = \emptyset, \forall i \in \{1, \ldots f\}$.

Proof: The strategy of the following proof is based on contradiction. Assume there exists a policy such that

$R_{\boldsymbol{\pi}}^{j}\cap F_{k}^{\mathcal{P}}\neq\emptyset$, $\forall k\in K$, where K is a subset of $2^{\{1,\ldots f\}}\setminus\{\{1,\ldots f\},\emptyset\}$. As discussed in [27], for each state in recurrent class, it holds that $\sum\limits_{n=0}^{\infty}p^{n}\left(x,x\right)=\infty$, where $x\in R_{\pi}^{j}\cap F_{k}^{\mathcal{P}}$ and $p^{n}\left(x,x\right)$ denotes the probability of returning from a transient state x to itself in n steps. This means that each state in the recurrent class occurs infinitely often. However, based on the embedded tracking-frontier function of E-LDGBA in Def. 3, the tracking set T will not be reset until all accepting sets have been visited. As a result, $x_{k}=\left(s,q_{k}\right)\in R_{\boldsymbol{\pi}}^{j}\cap F_{k}^{\mathcal{P}}$ with $s\in S$ will not occur infinitely, which contradicts the property $\sum\limits_{n=0}^{\infty}p^{n}\left(x_{k},x_{k}\right)=\infty$. ∎

Lemma 2 indicates that, for any policy, all accepting sets will be placed either in the transient class or in one of the recurrent classes.

V. Learning-based Control Synthesis

In this section, we discuss a design of reward and discount functions, and present rigorous analysis to show how such a design can guide the agent over the EP-MDP to find an optimal policy whose traces satisfy the LTL task with a maximum probability. Reinforcement learning is leveraged to identify policies for Problem 2.

A. Reward Design

Let $F_{U}^{\mathcal{P}}$ denote the union of accepting states, i.e., $F_{U}^{\mathcal{P}}=\left\{x\in X\middle|x\in F_{i}^{\mathcal{P}},\forall i\in\{1,\ldots f\}\right\}$. Inspired by [16], we propose a reward function as:

$$R\left(x\right)=\begin{cases}1-r_{F}, & \text{if } x\in F_{U}^{\mathcal{P}},\\ 0, & \text{otherwise,}\end{cases}\tag{6}$$

and a discount function as

$$\gamma\left(x\right)=\begin{cases}r_{F}, & \text{if } x\in F_{U}^{\mathcal{P}},\\ \gamma_{F}, & \text{otherwise,}\end{cases}\tag{7}$$

where $r_{F}\left(\gamma_{F}\right)$ is a function of γ_{F} satisfying $\lim\limits_{\gamma_{F}\rightarrow1^{-}}r_{F}\left(\gamma_{F}\right)=1$ and $\lim\limits_{\gamma_{F}\rightarrow1^{-}}\frac{1-\gamma_{F}}{1-r_{F}\left(\gamma_{F}\right)}=0$.

Given a path $\boldsymbol{x}_{t}=x_{t}x_{t+1}\ldots$ starting from x_{t}, the return is denoted by

$$\mathcal{D}\left(\boldsymbol{x}_{t}\right):=\sum_{i=0}^{\infty}\left(\prod_{j=0}^{i-1}\gamma\left(\boldsymbol{x}_{t}\left[t+i\right]\right)\cdot R\left(\boldsymbol{x}_{t}\left[t+j\right]\right)\right)\tag{8}$$

where it holds $\prod\limits_{j=0}^{-1}:=1$, and $\boldsymbol{x}_{t}\left[t+i\right]$ denotes the $\left(i+1\right)$th state in $\boldsymbol{x}_{t}$. Based on (8), the expected return of any state $x\in X$ under policy π can be defined as

$$U^{\boldsymbol{\pi}}\left(x\right)=\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_{t}\right)\left|\boldsymbol{x}_{t}\left[t\right]=x\right.\right].\tag{9}$$

A bottom strongly connected component (BSCC) of the Markov chain $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ (Definition 5) is a strongly connected component with no outgoing transitions.

Lemma 3. *For any path $\boldsymbol{x}_{t}$ and $\mathcal{D}\left(\boldsymbol{x}_{t}\right)$ in (8), it holds that $0\leq\gamma_{F}\cdot\mathcal{D}\left(\boldsymbol{x}_{t}\left[t+1:\right]\right)\leq\mathcal{D}\left(\boldsymbol{x}_{t}\right)\leq1-r_{F}+r_{F}\cdot\mathcal{D}\left(\boldsymbol{x}_{t}\left[t+1:\right]\right)\leq1$, where $\boldsymbol{x}_{t}\left[t+1:\right]$ denotes the suffix of $\boldsymbol{x}_{t}$ starting from x_{t+1}. Let $BSCC\left(MC_{\mathcal{P}}^{\boldsymbol{\pi}}\right)$ denote the set of all BSCCs of an induced Markov chain $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ and let $X_{\mathcal{P}}^{\boldsymbol{\pi}}$ denotes the set of accepting states that belongs to a BSCC of $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ s.t. $X_{\mathcal{P}}^{\boldsymbol{\pi}}:=\left\{x\in X\middle|x\in F_{U}^{\mathcal{P}}\cap BSCC\left(MC_{\mathcal{P}}^{\boldsymbol{\pi}}\right)\right\}$. Then, for any states $x\in X_{\mathcal{P}}^{\boldsymbol{\pi}}$, it holds that $\lim\limits_{\gamma_{F}\rightarrow1^{-}}U^{\boldsymbol{\pi}}\left(x\right)=1$.*

The proof of Lemma 3 is omitted since it is a straightforward extension of Lemma 2 and Lemma 3 in [16], by replacing LDBA with LDGBA. Since we apply the LDGBA with several accepting sets which might result in more complicated situations, e.g., AMEC, NMEC and RMEC, we can not obtain the same results as in [16]. We then establish the following theorem which is one of the main contributions.

Theorem 1. *Given the EP-MDP $\mathcal{P}=\mathcal{M}\times\overline{\mathcal{A}}_{\phi}$, for any state $x\in X$, the expected return under any policy π satisfies*

$$\exists i\in\{1,\ldots f\},\lim_{\gamma_{F}\rightarrow1^{-}}U^{\boldsymbol{\pi}}\left(x\right)=\Pr{}^{\boldsymbol{\pi}}\left[\Diamond F_{i}^{\mathcal{P}}\right],\tag{10}$$

where $\Pr^{\boldsymbol{\pi}}\left[\Diamond F_{i}^{\mathcal{P}}\right]$ is the probability that the paths starting from state x will eventually intersect any one $F_{i}^{\mathcal{P}}$ of $F^{\mathcal{P}}$.

Proof: Based on whether or not the path $\boldsymbol{x}_{t}$ intersects with accepting states of $F_{i}^{\mathcal{P}}$, the expected return in (9) can be rewritten as

$$\begin{aligned}U^{\boldsymbol{\pi}}\left(x\right)=&\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_{t}\right)\middle|\boldsymbol{x}_{t}\models\Diamond F_{i}^{\mathcal{P}}\right]\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_{i}^{\mathcal{P}}\right]\\&+\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_{t}\right)\middle|\boldsymbol{x}_{t}\neq\Diamond F_{i}^{\mathcal{P}}\right]\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\not\models\Diamond F_{i}^{\mathcal{P}}\right]\end{aligned}\tag{11}$$

where $\Pr^{\boldsymbol{\pi}}\left[x\models\Diamond F_{i}^{\mathcal{P}}\right]$ and $\Pr^{\boldsymbol{\pi}}\left[x\not\models\Diamond F_{i}^{\mathcal{P}}\right]$ represent the probability of eventually reaching and not reaching $F_{i}^{\mathcal{P}}$ under policy π starting from state x, respectively.

To find the lower bound of $U^{\boldsymbol{\pi}}\left(x\right)$, for any $\boldsymbol{x}_{t}$ with $\boldsymbol{x}_{t}\left[t\right]=x$, let $t+N_{t}$ be the index that $\boldsymbol{x}_{t}$ first intersects a state in $X_{\mathcal{P}}^{\boldsymbol{\pi}}$, i.e., $N_{t}=\min\left[i\middle|\boldsymbol{x}_{t}\left[t+i\right]\in X_{\mathcal{P}}^{\boldsymbol{\pi}}\right]$. The following holds

$$\begin{aligned}&\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_{t}\right)\middle|\boldsymbol{x}_{t}\models\Diamond F_{i}^{\mathcal{P}}\right]\\&\overset{(1)}{\geq}\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_{t}\right)\left|\boldsymbol{x}_{t}\cap X_{\mathcal{P}}^{\boldsymbol{\pi}}\neq\emptyset\right.\right]\\&\overset{(2)}{\geq}\mathbb{E}^{\boldsymbol{\pi}}\left[\gamma_{F}^{N_{t}}\cdot\mathcal{D}\left(\boldsymbol{x}_{t}\left[t+N_{t}:\right]\right)\left|\boldsymbol{x}_{t}\left[t+N_{t}\right]=x\right|\boldsymbol{x}_{t}\cap X_{\mathcal{P}}^{\boldsymbol{\pi}}\neq\emptyset\right]\\&\overset{(3)}{\geq}\mathbb{E}^{\boldsymbol{\pi}}\left[\gamma_{F}^{N_{t}}\middle|\boldsymbol{x}_{t}\cap X_{\mathcal{P}}^{\boldsymbol{\pi}}\neq\emptyset\right]\cdot U_{\min}^{\boldsymbol{\pi}}\left(\boldsymbol{x}_{t}\left[t+N_{t}\right]\right)\\&\overset{(4)}{\geq}\gamma_{F}^{\mathbb{E}^{\boldsymbol{\pi}}\left[N_{t}\left|\boldsymbol{x}_{t}\left[t\right]=x\right|\boldsymbol{x}_{t}\cap X_{\mathcal{P}}^{\boldsymbol{\pi}}\neq\emptyset\right]}\cdot U_{\min}^{\boldsymbol{\pi}}\left(x_{Acc}\right)\\&=\gamma_{F}^{n_{t}}\cdot U_{\min}^{\boldsymbol{\pi}}\left(x_{Acc}\right),\end{aligned}\tag{12}$$

where $x_{Acc}\in X_{\mathcal{P}}^{\boldsymbol{\pi}}$, $U_{\min}^{\boldsymbol{\pi}}\left(x_{Acc}\right)=\min_{x\in X_{\mathcal{P}}^{\boldsymbol{\pi}}}U^{\boldsymbol{\pi}}\left(x\right)$, and n_{t} is a constant. By Lemma 3, one has $\lim\limits_{\gamma_{F}\rightarrow1^{-}}U_{\min}^{\boldsymbol{\pi}}\left(x_{Acc}\right)=1$. In (12), the first inequality **(1)** holds because visiting $X_{\mathcal{P}}^{\boldsymbol{\pi}}$ is one of the cases for $\Diamond F_{i}^{\mathcal{P}}$ so that $\boldsymbol{x}_{t}\models\Diamond F_{i}^{\mathcal{P}}$, e.g., $F_{i}^{\mathcal{P}}$ can be placed outside of all BSCCs; the second inequality **(2)** holds due to Lemma 3; the third inequality **(3)** holds due to the Markov properties of (8) and (9); the fourth inequality **(4)** holds due to Jensen's inequality. Based on (12), the lower bound of (11) is $U^{\boldsymbol{\pi}}\left(x\right)\geq\gamma_{F}^{n_{t}}\cdot U_{\min}^{\boldsymbol{\pi}}\left(x_{Acc}\right)\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_{i}^{\mathcal{P}}\right]$ from which one has

$$\lim_{\gamma_{F}\rightarrow1^{-}}U^{\boldsymbol{\pi}}\left(x\right)\geq\gamma_{F}^{n_{t}}\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_{i}^{\mathcal{P}}\right].\tag{13}$$

Similarly, let $t+M_{t}$ denote the index that $\boldsymbol{x}_{t}$ first enters a

BSCC that contains no accepting states. We have

$$\mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}\left(\boldsymbol{x}_t\right)\middle|\boldsymbol{x}_t \not\models \Diamond F_i^{\mathcal{P}}\right] \overset{(1)}{\leq} \mathbb{E}^{\pi}\left[1-r_F^{M_t}\middle|\boldsymbol{x}_t \not\models \Diamond F_i^{\mathcal{P}}\right] \overset{(2)}{\leq} 1-r_F^{\mathbb{E}^{\boldsymbol{\pi}}\left[M_t|\boldsymbol{x}_t[t]=x,\boldsymbol{x}_t\not\models\Diamond F^{\mathcal{P}}\right]} = 1-r_F^{m_t} \tag{14}$$

where m_t is a constant and (14) holds due to Lemma 3 and Markov properties.

Hence, the upper bound of (11) is obtained as

$$\lim_{\gamma_F\to 1^-} U^{\boldsymbol{\pi}}\left(x\right) \leq \Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_i^{\mathcal{P}}\right] + (1-r_F^{m_t})\Pr{}^{\boldsymbol{\pi}}\left[x\not\models\Diamond F_i^{\mathcal{P}}\right]. \tag{15}$$

By (13) and (15), we can conclude

$$\gamma_F^{n_t}\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_i^{\mathcal{P}}\right] \leq \lim_{\gamma_F\to 1^-} U^{\boldsymbol{\pi}}\left(x\right) \leq \Pr{}^{\boldsymbol{\pi}}\left[x\models\Diamond F_i^{\mathcal{P}}\right] + (1-r_F^{m_t})\cdot\Pr{}^{\boldsymbol{\pi}}\left[x\not\models\Diamond F_i^{\mathcal{P}}\right]$$

According to $\lim_{\gamma_F\to 1^-} r_F\left(\gamma_F\right)=1$ in the reward function, (10) can be concluded. ■

When the condition $\gamma_F \to 1^-$ holds, [16] proves the expected return as the probability of satisfying the accepting condition of LDBA. Different from [16], Theorem 1 only states that the expected return indicates the probability of visiting an accepting set, rather than showing the probability of satisfying the acceptance condition of E-LDGBA. Nevertheless, we will show in the following section how Theorem 1 can be leveraged to solve Problem 2.

Theorem 2. *Consider a PL-MDP $\mathcal{M}$ and an E-LDGBA $\overline{\mathcal{A}}_\phi$ corresponding to an LTL formula ϕ. Based on assumption 1, there exists a discount factor $\underline{\gamma}$ and any optimization method for (9) with $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$ to to obtain a policy $\bar{\boldsymbol{\pi}}$, then the induced run $r_{\mathcal{P}}^{\bar{\boldsymbol{\pi}}}$ satisfies the accpeting condition of the corresponding $\mathcal{P}$ (Def. 4).*

Proof: For any policy $\boldsymbol{\pi}$, $MC_{\mathcal{P}}^{\boldsymbol{\pi}} = \mathcal{T}_{\boldsymbol{\pi}}\sqcup\mathcal{R}_{\boldsymbol{\pi}}^1\sqcup\mathcal{R}_{\boldsymbol{\pi}}^2\ldots\mathcal{R}_{\boldsymbol{\pi}}^{n_R}$. Let $\boldsymbol{U}_{\boldsymbol{\pi}} = \begin{bmatrix} U^{\boldsymbol{\pi}}\left(x_0\right) & U^{\boldsymbol{\pi}}\left(x_1\right) & \ldots \end{bmatrix}^T \in \mathbb{R}^{|X|}$ denote the stacked expected return under policy $\boldsymbol{\pi}$, which can be reorganized as

$$\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{U}_{\boldsymbol{\pi}}^{rec} \end{bmatrix} = \sum_{n=0}^{\infty}\left(\prod_{j=0}^{n-1}\begin{bmatrix} \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{\mathcal{T}} & \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{0}_{\sum_{i=1}^{m}N_i\times r} & \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{rec} \end{bmatrix}\right) \cdot \begin{bmatrix} \boldsymbol{P}_{\boldsymbol{\pi}}\left(\mathcal{T},\mathcal{T}\right) & \boldsymbol{P}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{0}_{\sum_{i=1}^{m}N_i\times r} & \boldsymbol{P}_{\boldsymbol{\pi}}\left(\mathcal{R},\mathcal{R}\right) \end{bmatrix}^n \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{R}_{\boldsymbol{\pi}}^{rec} \end{bmatrix}, \tag{16}$$

where $\boldsymbol{U}_{\boldsymbol{\pi}}^{tr}$ and $\boldsymbol{U}_{\boldsymbol{\pi}}^{rec}$ are the expected return of states in transient and recurrent classes under policy $\boldsymbol{\pi}$, respectively. In (16), $\boldsymbol{P}_{\pi}\left(\mathcal{T},\mathcal{T}\right)\in\mathbb{R}^{r\times r}$ is the probability transition matrix between states in $\mathcal{T}_{\boldsymbol{\pi}}$, and $\boldsymbol{P}_{\boldsymbol{\pi}}^{tr} = [P_{\boldsymbol{\pi}}^{tr_1}\ldots P_{\boldsymbol{\pi}}^{tr_m}]\in\mathbb{R}^{r\times\sum_{i=1}^{m}N_i}$ is the probability transition matrix where $P_{\boldsymbol{\pi}}^{tr_i}\in\mathbb{R}^{r\times N_i}$ represents the transition probability from a transient state in $\mathcal{T}_{\boldsymbol{\pi}}$ to a state of $\mathcal{R}_{\boldsymbol{\pi}}^i$. The $\boldsymbol{P}_{\boldsymbol{\pi}}\left(\mathcal{R},\mathcal{R}\right)$ is a diagonal block matrix, where the ith block is a $N_i\times N_i$ matrix containing the transition probabilities between states within $\mathcal{R}_{\boldsymbol{\pi}}^i$. Note that $\boldsymbol{P}_{\boldsymbol{\pi}}\left(\mathcal{R},\mathcal{R}\right)$ is a stochastic matrix since each block matrix is a stochastic matrix [27]. Similarly, the rewards $\boldsymbol{R}_{\pi}$ can also be partitioned into $\boldsymbol{R}_{\boldsymbol{\pi}}^{tr}$ and $\boldsymbol{R}_{\boldsymbol{\pi}}^{rec}$.

The following proof is based on contradiction. Suppose there exists a policy $\boldsymbol{\pi}^*$ that optimizes the expected return, but not satisfy the accepting condition of $\mathcal{P}$. Based on Lemma 2, the following is true: $F_k^{\mathcal{P}}\subseteq\mathcal{T}_{\boldsymbol{\pi}^*},\forall k\in\{1,\ldots f\}$, where $\mathcal{T}_{\boldsymbol{\pi}^*}$ denotes the transient class of Markov chain induced by $\boldsymbol{\pi}^*$ on $\mathcal{P}$. First, consider a state $x_R\in\mathcal{R}_{\boldsymbol{\pi}^*}^j$ and let $\boldsymbol{P}_{\boldsymbol{\pi}^*}^{x_R R_j}$ denote a row vector of $\boldsymbol{P}_{\boldsymbol{\pi}^*}^n\left(\mathcal{R},\mathcal{R}\right)$ that contains the transition probabilities from x_R to the states in the same recurrent class $\mathcal{R}_{\boldsymbol{\pi}^*}^j$ after n steps. The expected return of x_R under $\boldsymbol{\pi}^*$ is then obtained from (16) as

$$U_{\boldsymbol{\pi}^*}^{rec}\left(x_R\right) = \sum_{n=0}^{\infty}\gamma^n\left[\boldsymbol{0}_{k_1}^T\ \boldsymbol{P}_{\pi^*}^{x_R R_j}\ \boldsymbol{0}_{k_2}^T\right]\boldsymbol{R}_{\boldsymbol{\pi}^*}^{rec},$$

where $k_1=\sum_{i=1}^{j-1}N_i$, $k_2=\sum_{i=j+1}^{n}N_i$. Since $\mathcal{R}_{\boldsymbol{\pi}^*}^j\cap F_i^{\mathcal{P}}=\emptyset,\forall i\in\{1,\ldots f\}$, by the designed reward function, all entries of $\boldsymbol{R}_{\boldsymbol{\pi}^*}^{rec}$ are zero. We can conclude $U_{\boldsymbol{\pi}^*}^{rec}\left(x_R\right)=0$. To show contradiction, the following analysis will show that $U_{\bar{\boldsymbol{\pi}}}^{rec}\left(x_R\right) > U_{\boldsymbol{\pi}^*}^{rec}\left(x_R\right)$ for any policy $\bar{\boldsymbol{\pi}}$ that satisfies the accepting condition of $\mathcal{P}$. Thus, it's true that there exists $\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j$ such that $\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j\cap F_k^{\mathcal{P}}\neq\emptyset,\forall k\{1,\ldots f\}$. We use $\underline{\gamma}$ and $\overline{\gamma}$ to denote the lower and upper bound of γ.

Case 1: If $x_R\in\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j$, there exist states such that $x_\Lambda\in\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j\cap F_i^{\mathcal{P}}$. From Lemma 2, the entries in $\boldsymbol{R}_{\bar{\boldsymbol{\pi}}}^{rec}$ corresponding to the recurrent states in $\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j$ have non-negative rewards and at least there exist f states in $\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j$ from different accepting sets $F_i^{\mathcal{R}}$ with positive reward r_F. From (16), $U_{\bar{\boldsymbol{\pi}}}^{rec}\left(x_R\right)$ can be lower bounded as

$$U_{\bar{\boldsymbol{\pi}}}^{rec}\left(x_R\right) \geq \sum_{n=0}^{\infty}\underline{\gamma}^n\left(P_{\bar{\boldsymbol{\pi}}}^{x_R x_\Lambda}r_F\right) > 0,$$

where $P_{\bar{\boldsymbol{\pi}}}^{x_R x_\Lambda}$ is the transition probability from x_R to x_Λ in n steps. We can conclude in this case $U_{\bar{\boldsymbol{\pi}}}^{rec}\left(x_R\right) > U_{\boldsymbol{\pi}^*}^{rec}\left(x_R\right)$.

Case 2: If $x_R\in\mathcal{T}_{\bar{\boldsymbol{\pi}}}$, there are no states of any accepting set $F_i^{\mathcal{P}}$ in $\mathcal{T}_{\bar{\boldsymbol{\pi}}}$. As demonstrated in [27], for a transient state $x_{tr}\in\mathcal{T}_{\bar{\boldsymbol{\pi}}}$, there always exists an upper bound $\Delta<\infty$ such that $\sum_{n=0}^{\infty}p^n\left(x_{tr},x_{tr}\right)<\Delta$, where $p^n\left(x_{tr},x_{tr}\right)$ denotes the probability of returning from a transient state x_T to itself in n time steps. In addition, for a recurrent state x_{rec} of $\mathcal{R}_{\bar{\boldsymbol{\pi}}}^j$, it is always true that

$$\sum_{n=0}^{\infty}\gamma^n p^n\left(x_{rec},x_{rec}\right) > \frac{1}{1-\gamma^{\overline{n}}}\bar{p}, \tag{17}$$

where there exists $\overline{n}$ such that $p^{\overline{n}}\left(x_{rec},x_{rec}\right)$ is nonzero and can be lower bounded by $\bar{p}$ [27]. From (16), one has

$$\boldsymbol{U}_{\bar{\boldsymbol{\pi}}}^{tr} > \sum_{n=0}^{\infty}\left(\prod_{j=0}^{n-1}\boldsymbol{\gamma}_{\bar{\boldsymbol{\pi}}}^{tr}\right)\cdot\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr}\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^n\left(\mathcal{R},\mathcal{R}\right)\boldsymbol{R}_{\bar{\boldsymbol{\pi}}}^{rec} > \underline{\gamma}^n\cdot\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr}\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^n\left(\mathcal{R},\mathcal{R}\right)\boldsymbol{R}_{\bar{\boldsymbol{\pi}}}^{rec}. \tag{18}$$

Let $\max\left(\cdot\right)$ and $\min\left(\cdot\right)$ represent the maximum and minimum entry of an input vector, respectively. The upper bound $\bar{m}=\left\{\max\left(\overline{M}\right)\middle|\overline{M}<\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr}\bar{\boldsymbol{P}}\boldsymbol{R}_{\bar{\boldsymbol{\pi}}}^{rec}\right\}$ and $\bar{m}\geq 0$, where $\bar{\boldsymbol{P}}$ is a block matrix whose nonzero entries are derived similarly to $\bar{p}$ in (17). The utility $U_{\bar{\boldsymbol{\pi}}}^{tr}\left(x_R\right)$ can be lower bounded from (17) and (18) as

$$U_{\bar{\boldsymbol{\pi}}}^{tr}\left(x_R\right) > \frac{1}{1-\underline{\gamma}^n}\bar{m}. \tag{19}$$

Since $U_{\boldsymbol{\pi}^*}^{rec}(x_R) = 0$, the contradiction $U_{\boldsymbol{\pi}}^{tr}(x_R) > 0$ is achieved if $\frac{1}{1-\underline{\gamma}^n}\bar{m}$. Thus, there exist $0 < \underline{\gamma} < 1$ such that $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$, which implies $U_{\boldsymbol{\pi}}^{tr}(x_R) > \frac{1}{1-\underline{\gamma}^n}\bar{m} \geq 0$. The procedure shows the contradiction of the assumption that $\boldsymbol{\pi}^*$ that does not satisfy the acceptance condition of $\mathcal{P}$ is optimal, and Theorem 2 is proved. ■

Theorem 2 proves that by selecting $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$, optimizing the expected return in (9) can find a policy satisfying the given task ϕ.

Theorem 3. *Given a PL-MDP $\mathcal{M}$ and an E-LDGBA $\overline{\mathcal{A}}_\phi$, by selecting $\gamma_F \to 1^-$, the optimal policy $\boldsymbol{\pi}^*$ that maximizes the expected return (9) of the corresponding EP-MDP also maximizes the probability of satisfying ϕ, i.e.,* $\Pr^{\boldsymbol{\pi}^*}[x \models Acc_{\mathcal{P}}] = \Pr_{max}[x \models Acc_{\mathcal{P}}]$.

Proof: Since $\gamma_F \to 1^-$, we have $\gamma_F > \underline{\gamma}$ and $r_F > \underline{\gamma}$ from Theorem 2. There exists an induced run $r_{\mathcal{P}}^{\boldsymbol{\pi}^*}$ satisfying the accepting condition of $\mathcal{P}$. According to Lemma 1, $\lim\limits_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}^*}(x)$ is exactly equal to the probability of visiting the accepting sets of an AMEC. Optimizing $\lim\limits_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}^*}(x)$ is equal to optimizing the probability of entering AMECs. ■

B. Model-Free Reinforcement Learning

Based on the Q-learning [8], the agent updates its Q-value from x to x' according to

$$Q\left(x, u^{\mathcal{P}}\right) \leftarrow (1-\alpha) Q\left(x, u^{\mathcal{P}}\right) + \alpha\left[R(x) + \gamma(x) \cdot \max_{\overline{u}^{\mathcal{P}} \in U^{\mathcal{P}}} Q\left(x', \overline{u}^{\mathcal{P}}\right)\right], \tag{20}$$

where $Q\left(x, u^{\mathcal{P}}\right)$ is the Q-value of the state-action pair $\left(x, u^{\mathcal{P}}\right)$, $0 < \alpha \leq 1$ is the learning rate, $0 \leq \gamma(x) \leq 1$ is the discount function as defined in section V-A. With the standard learning rate and discount factor, Q-value will converge to a unique limit Q^* as in [8]. Therefore, the optimal expected utility and policy can be obtained as $U_{\boldsymbol{\pi}}^*(x) = \max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}(x)} Q^*\left(x, u^{\mathcal{P}}\right)$ and $\boldsymbol{\pi}^*(x) = \arg\max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}(x)} Q^*\left(x, u^{\mathcal{P}}\right)$.

[1]The learning strategy is outlined in Alg. 2. In line 2, the dependent and discount values are selected based on section V-A.

Algorithm 2 RL based motion planning under LTL

1: **procedure** INPUT: ($\mathcal{M}$, ϕ, Λ)
Output: optimal policy $\boldsymbol{\pi}^*$
Initialization: Set $episode = 0$, $iteration = 0$ and τ (maximum allowed learning steps)
2: set $r_F = 0.99$ and $\gamma_F = 0.9999$ to determine $R(x)$ and $\gamma(x)$
3: **for** all $x \in X$ **do**
4: $U(x) = 0$ and $Q\left(x, u^{\mathcal{P}}\right) = 0$, $\forall u^{\mathcal{P}} \in U^{\mathcal{P}}(x)$
5: $Count\left(x, u^{\mathcal{P}}\right) = 0$, $\forall u^{\mathcal{P}} \in U^{\mathcal{P}}(x)$
6: **end for**
7: $x_{curr} = x_0$;
8: **while** $\boldsymbol{U}$ are not converged **do**
9: $episode++$;
10: $\epsilon = 1/episode$;
11: **while** $iteration < \tau$ **do**
12: $iteration++$
13: Select $u_{curr}^{\mathcal{P}}$ based on epsilon-greedy selection
14:
15: Execute $u_{curr}^{\mathcal{P}}$ and observer x_{next}, $R(x_{curr})$,$\gamma(x_{curr})$
16: $r \leftarrow R(x_{curr})$ and $\gamma \leftarrow \gamma(x_{curr})$
17: $Count\left(x_{curr}, u_{curr}^{\mathcal{P}}\right)++$
18: $\alpha = 1/Count\left(x_{curr}, u_{curr}^{\mathcal{P}}\right)$
19: $Q\left(x_{curr}, u_{curr}^{\mathcal{P}}\right) \leftarrow (1-\alpha) Q\left(x_{curr}, u_{curr}^{\mathcal{P}}\right) + \alpha\left[r + \gamma \cdot \max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}} Q\left(x_{next}, u^{\mathcal{P}}\right)\right]$
20: $x_{curr} = x_{next}$
21: **end while**
22: **end while**
23: **for** all $x \in X$ **do**
24: $\boldsymbol{\pi}^*(x) = \max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}} Q\left(x, u^{\mathcal{P}}\right)$
25: **end for**
26: **end procedure**

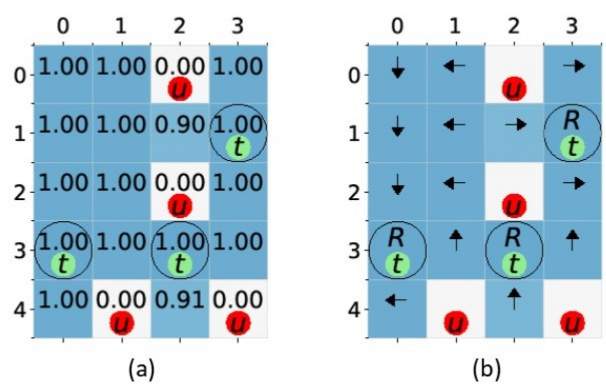

Figure 2: (a) The estimated maximal satisfaction probability., where the targets and unsafe areas are denoted by `t` and `u`, respectively. (b) The optimal policy of satisfying φ_{case1}, where "R" is an action primitive means the robot remains at its current cell.

VI. CASE STUDIES

The developed RL-based control synthesis is implemented in Python. Owl [24] is used to convert LTL specifications to LDGBA. We implement the Alg. 2 to validate the effectiveness of our approach, and we first carry out simulations over grid environments and then validate the approach in a more realistic office scenario with TurtleBot3 robot, which consider the uncertainties of both motion and environment. The software can be find in our Github repository[2].

A. Simulation Results

Consider a mobile robot following the unicycle model, i.e. $\dot{x} = v\sin(\theta)$, $\dot{y} = v\cos(\theta)$, and $\dot{\theta} = \omega$, where x, y, θ indicate the robot positions and orientation. The linear and angular velocities are the control inputs, i.e., $u = (v, \omega)$. The workspace is shown in Fig. 2 and Fig. 3. To model motion uncertainties, we assume the action primitives can not always be successfully executed. For instance, action primitives "N, S, E, W" mean the robot can successfully move towards north, south, east and west (four possible orientations) to adjacent cells with probability 0.9, respectively, and fails by moving sideways with probability 0.1. Action primitive "R" means the robot remains at its current cell.

(1) Maximum Satisfaction Probability: In this case, the objective is to verify that the generated policy satisfies the LTL

[1]Any other model-free RL algorithm can also be adopted with Alg.2
[2]https://github.com/mingyucai/E-LDGBA_RL

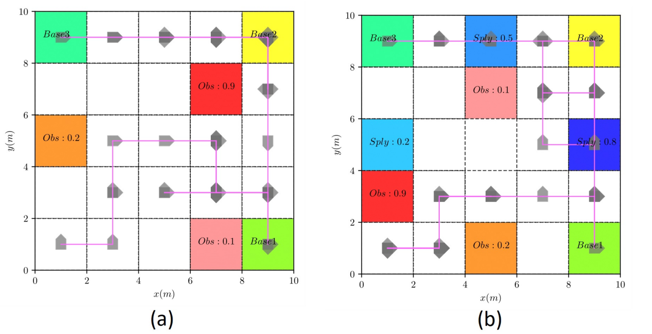

Figure 3: Simulated trajectories of 25 time steps under the corresponding optimal policies.

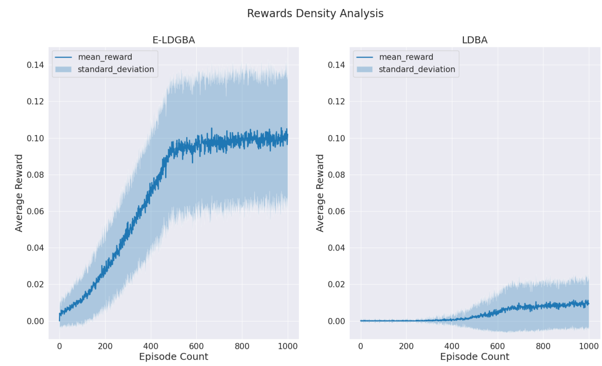

Figure 4: The mean and standard deviation of rewards via E-LDGBA (left) and LDBA (right) respectively

specification with a maximum probability. The package Csrl in [16] is used. The LTL specification is

$$\varphi_{case1} = \Diamond \Box \mathtt{t} \wedge \Box \neg \mathtt{u}, \tag{21}$$

which requires the robot to eventually arrive at one of the targets $\mathtt{t}$ while avoiding unsafe areas $\mathtt{u}$. Each episode terminates after $\tau = 100$ steps. Fig. 2 (a) shows the estimated maximum probability of satisfying φ_{case1} starting from each state. Note that the maximum satisfaction probability starting from $(2, 1)$ is 0.9, since the robot can move sideways with probability 0.1 due to motion uncertainties. Suppose the robot starts from $(0, 0)$, Fig. 2 (b) shows the generated optimal policy at each state, and the robot will complete φ_{case1} with probability one by eventually visiting either $(0, 3)$ or $(2, 3)$. After arriving at the destination, the robot will select "R" to stay at the target.

We then verify more complex LTL specifications over the infinite horizon. As shown in Fig. 3, the cells are marked with different colors to represent different areas of interest, e.g., $\mathtt{Base1}, \mathtt{Base2}, \mathtt{Base3}, \mathtt{Obs}, \mathtt{Sply}$, where $\mathtt{Obs}$ and $\mathtt{Sply}$ are shorthands for obstacle and supply, respectively. To model the environment uncertainties, the number associated with a cell represents the likelihood that the corresponding property appears at that cell. For example, $\mathtt{Obs} : 0.1$ indicates this cell is occupied by the obstacles with probability 0.1. In Fig. 3 (a), we first consider a case that user-specified tasks can all be successfully executed. The desired surveillance task to be performed is formulated as

$$\varphi_{case2} = (\Box\Diamond \mathtt{Base1}) \wedge (\Box\Diamond \mathtt{Base2}) \wedge (\Box\Diamond \mathtt{Base3}) \wedge \Box\neg \mathtt{Obs},$$

which requires the mobile robot to visit all base stations infinitely often while avoiding the obstacles. In this case, each episode terminates after $\tau = 100$ steps. The generated optimal trajectory is shown in Fig. 3 (a), which indicates φ_{case1} is completed. We then validate our approach with more complex task specifications

$$\varphi_{case3} = \varphi_{case1} \wedge \Box \left(\mathtt{Sply} \rightarrow \bigcirc \left((\neg \mathtt{Sply}) \cup \varphi_{\mathtt{one1}}\right)\right),$$

where $\varphi_{one1} = \mathtt{Base1} \vee \mathtt{Base2} \vee \mathtt{Base3}$. φ_{case3} requires the robot to visit the supply station and then go to one of the base stations while avoiding obstacles and requiring all base stations to be visited. In this case, each episode terminates after $\tau = 800$ steps. The generated optimal trajectory is shown in Fig. 3 (b).

Table I: Simulation results of large scale workspaces

Workspace size[cell]	MDP States	$\mathcal{R}$ States	Episode Steps
15×15	225	450	800
25×25	625	1250	2000
40×40	1600	3200	5000

(2) Reward Density: Since LDBA can be considered as a special case of E-LDGBA with only one accepting set, LDBA is compared with E-LDGBA in this case. To show the benefits of applying E-LDGBA over LDBA in overcoming the issues of spare rewards, we perform 100 learning iterations for 1000 episodes and compare the reward collection in the training process. The RL-based policy synthesis is carried out for φ_{case2}. Fig. 4 shows the mean and standard deviations of collected rewards using E-LDGBA and LDBA, respectively. It can be seen that E-LDGBA based method converges faster since the accepting set is tracked every time when the robot visits one of the base stations. In contrast, LDBA only records the time when all base states have been visited. Hence, the sparse reward issue is relaxed in our method.

(3) Scalability: To show the computational complexity, the RL-based policy synthesis is also performed for φ_{case2} over workspaces of various sizes (each grid is further partitioned). The simulation results are listed in Table I, which consists of the number of MDP states, the number of relaxed product MDP states. The steps in Table I indicate the time used to converge to an optimal satisfaction planing when applying reinforcement learning. It is also verified that the given task φ_{case2} can be successfully carried out in larger workspaces.

B. Experimental Results

Consider an office environment constructed in ROS Gazebo as shown in Fig.5, which consists of 7 rooms denoted by $S_0, S_2, S_3, S_5, S_7, S_9, \mathtt{Obs}$ and 5 corridors denoted by S_1, S_6, S_8. The two black dash lines are the dividing lines for corridors S_1, S_6 and S_6, S_8, separately. Starting from room S_0, the TurtleBot3 can follow a collision-free path from the center of one region to another without crossing other regions using obstacle-avoidance navigation. To model motion

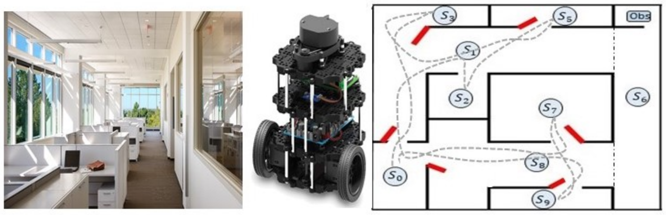

Figure 5: The mock-up office scenario with the TurtleBot3 robot.

uncertainties, it is assumed that the robot can successfully follow its navigation controller moving from corridors to a desired room with probability 0.9 and fail by moving to the adjacent room with probability 0.1. In addition, the robot can successfully moving between corridors with probability 1.0. To model environment uncertainties, we set the door of each room keeps open with probability 0.9 and close with probability 0.1. The service to be performed by TurtleBot3 is expressed as

$$\varphi_{case4} = \varphi_{all} \wedge \square\neg\texttt{Obs}, \tag{22}$$

where $\varphi_{all} = \square\lozenge S_2 \wedge \square\lozenge S_3 \wedge \square\lozenge S_5 \wedge \square\lozenge S_9 \wedge \square\lozenge S_{10}$. In (22), φ_{all} requires the robot to always service all rooms (e.g. pick trash) and return to S_0 (e.g. release trash), while avoiding $\texttt{Obs}$. The optimal policy for the case is generated that each episode terminates after $\tau = 150$ steps. The generated satisfying trajectories (without collision) marked as gray bold dash line are shown in Fig. 5. To maximize the satisfaction probability, it is observed that the optimal policy avoids the corridor S_6, since there is a non-zero probability of entering $\texttt{Obs}$ from S_6 under any policies due to the uncertainties, resulting in the violation of φ_{case3}.

VII. Conclusion

In this paper, the LTL specifications are translated to E-LDGBA to apply the deterministic policy, and a model-free learning-based algorithm is developed to synthesize control policies that maximize the satisfaction of LTL specifications. Future research will focus on deep RL to address continuous state and action spaces.

References

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

[2] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control based motion planning with partially infeasible LTL constrains," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1279–1284, 2020.

[3] Y. Kantaros and M. M. Zavlanos, "Stylus*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 812–836, 2020.

[4] 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, 2014.

[5] 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, 2018.

[6] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1098–1123, 2019.

[7] M. Cai, Z. Li, S. Xiao, and Z. Kan, "Optimal probabilistic motion planning with partially infeasible LTL constraints," *arXiv preprint arXiv:2007.14325*, 2020.

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

[9] 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.

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

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

[12] X. Li, Z. Serlin, G. Yang, and C. Belta, "A formal methods approach to interpretable reinforcement learning for robotic planning," *Sci. Robot.*, vol. 4, no. 37, 2019.

[13] 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.

[14] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, 2020.

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

[16] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *Int. Conf. Robot. Autom.* IEEE, 2020, pp. 10 349–10 355.

[17] 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 *Proc. IEEE Conf. Decis. Control*. IEEE, 2019, pp. 5338–5343.

[18] R. Oura, A. Sakakibara, and T. Ushio, "Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic generalized büchi automata," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 761–766, 2020.

[19] C. Wang, Y. Li, S. L. Smith, and J. Liu, "Continuous motion planning with temporal logic specifications using deep neural networks," *arXiv preprint arXiv:2004.02610*, 2020.

[20] M. Hasanbeig, N. Yogananda Jeppu, A. Abate, T. Melham, and D. Kroening, "DeepSynth: Program synthesis for automatic task segmentation in deep reinforcement learning," in *AAAI Conf. Artif. Intell.* Association for the Advancement of Artificial Intelligence, 2021.

[21] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Int. Conf. Mach. Learn.*, 2018, pp. 2107–2116.

[22] A. Camacho, R. T. Icarte, T. Q. Klassen, R. A. Valenzano, and S. A. McIlraith, "LTL and beyond: Formal languages for reward function specification in reinforcement learning." in *IJCAI*, vol. 19, 2019, pp. 6065–6073.

[23] S. Sickert, J. Esparza, S. Jaax, and J. Křetínskỳ, "Limit-deterministic Büchi automata for linear temporal logic," in *Int. Conf. Comput. Aided Verif.* Springer, 2016, pp. 312–332.

[24] J. Kretínský, T. Meggendorfer, and S. Sickert, "Owl: A library for ω-words, automata, and LTL," in *Autom. Tech. Verif. Anal.* Springer, 2018, pp. 543–550. [Online]. Available: https://doi.org/10.1007/978-3-030-01090-4_34

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

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

[27] R. Durrett and R. Durrett, *Essentials of stochastic processes*. Springer, 1999, vol. 1.