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 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 with unknown transition probabilities and unknown probabilistic label 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 without increasing dimensional and computational complexity. With appropriate dependent reward and discount functions, rigorous analysis shows that any method that optimizes the expected discount return of the RL-based approach is guaranteed to find the optimal policy that maximizes 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 more challenges when 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 sequential decision-making process in which an agent continuously interacts with and learns from the environment [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. 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) is used as reward to facilitate learning. However, only finite horizon motion planning is considered. In [13] and [14], LTL constraints are 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) is employed in [15] and [16] without considering the workspace uncertainties. Since LDBA has only one accepting set, it might lead to spares reward issues during learning. In [17], limit-deterministic generalized Buchi automaton (LDGBA) is used, and a frontier function of rewards is designed to facilitate learning by assigning positive rewards to the accepting sets. However, the memoryless method in [17] cannot track visited or non-visited accepting sets. The works of [18] and [19] overcome this issue by designing binary-valued vectors and Boolean vectors, respectively. However, [18] cannot guarantee the maximum probability of task satisfaction, and the reward function of [19] adds extra computational complexity to the training process due to the dependence on a learning-varying set.

In this paper, we consider motion planning that maximizes the probability of satisfying pre-specified LTL tasks. Due to motion and environment uncertainties, we model the robot motion as a probabilistic labeled Markov decision process (PL-MDP) with transition probabilities and probabilistic label functions that are assumed to be unknown and translate LTL specifications to an (LDGBA) [20] to maintain dense rewards during learning. Inspired by the reward and discount functions from [16], an expected return is designed. In [16], workspace uncertainties are not considered, and the developed expected return is not applicable to LDGBA due to the acceptance condition of LDGBA containing several accepting sets. In this work, a synchronous tracking-frontier function is designed for LDGBA to construct an embedded LDGBA (E-LDGBA) that is capable of recording non-visited accepting sets without increasing dimensional and computational complexity. Rigorous analysis shows that any method that optimizes the expected discount return of the RL-based approach is guaranteed to find the optimal policy that maximizes the satisfaction probability of the LTL specifications. A model-free RL-based motion planning strategy is developed to generate the optimal policy.

The contributions are multi-fold. PL-MDP is introduced to consider both motion and environment uncertainties. We develop an E-LDGBA, based on which the control synthesis problem is converted to an expected return optimization problem. We show that, by leveraging RL, the desired policy is guaranteed to be found to optimize the expected return.

[1]Department of Mechanical Engineering, The University of Iowa, Iowa City, IA, USA.

[5]Department of Automation, University of Science and Technology of China, Hefei, Anhui, China.

Compared with the literature, the proposed RL approach is model-free and can be applied to PL-MDPs with uncertainties. By designing E-LDGBA, the sparse issue in many existing works is relaxed and the framework is capable of recording visited and non-visited accepting sets in each round without increasing the complexity of algorithm computation and automaton graph dimensions. In addition, compared with the existing works that accomplish LTL specifications, the generated optimal policy can maximize the probability of LTL task satisfaction over an uncertain PL-MDP.

II. PRELIMINARIES

A. Probabilistic Labeled MDP

A probabilistic labeled finite MDP (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, $p_S:S\times A\times S\rightarrow[0,1]$ is the transition probability function, Π is a set of atomic propositions, and $L:S\rightarrow 2^{\Pi}$ 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 $\sum_{l\in L(s)}p_L(s,l)=1,\forall s\in S$. 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.

Let ξ be an action function, which can be either deterministic such that $\xi:S\rightarrow A$ maps a state $s\in S$ to an action in $A(s)$, or randomized such that $\xi:S\times A\rightarrow[0,1]$ represents the probability of taking an action in $A(s)$ at 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 $\varLambda:S\rightarrow\mathbb{R}$ denote a reward function over $\mathcal{M}$. Given a discount factor $\gamma\in(0,1)$, the expected return under policy ξ starting from $s'\in S$ can be defined as $U^{\xi}(s')=\mathbb{E}^{\xi}\left[\sum\limits_{i=0}^{\infty}\gamma^i\varLambda(s'_i)\,|s'_0=s'\right]$. The optimal policy ξ^* is a policy that maximizes the expected return for each state $s\in S$ as $\xi^*=\underset{\xi}{\arg\max}\,U^{\xi}(s)$.

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 [20]. 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, $\delta\colon Q\times\Sigma\rightarrow 2^{Q}$ 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 $Q\times(\Sigma\cup\{\epsilon\})\rightarrow 2^{Q}$, 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 states 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 [21].

III. PROBLEM STATEMENTS

The task specification to be performed by the robot is described by an LTL formula ϕ over Π. Given the 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 $p_L(s_i,l_i)>0$. Denote by $L\left(\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}\right)\models\phi$ if the induced $\boldsymbol{s}_{\infty}^{\boldsymbol{\xi}}$ satisfies ϕ. The probabilistic satisfaction under the policy ξ from an initial state s_0 can be defined as

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

where $\boldsymbol{S}_{\infty}^{\boldsymbol{\xi}}$ is a set of admissible paths.

Assumption1. It is assumed that there exists at least one policy that satisfies task ϕ.

Assumption1 is a mild assumption widely employed in the literature (cf. [9], [15], [16]). 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 unknown probabilistic label functions (i.e., workspace uncertainties), the goal is to find the desired policy ξ^* that maximizes the satisfaction probability, i.e., $\xi^*=\underset{\xi}{\arg\max}\Pr{}_M^{\xi}(\phi)$, by interacting with the environment.

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 to keep tracking the non-visited accepting sets. Section IV-B presents the construction of a product MDP between a PL-MDP and an E-LDGBA. The benefits of incorporating E-LDGBA are discussed in Section IV-C.

Algorithm 1 Procedure of E-LDGBA

procedure INPUT: (LDGBA $\mathcal{A}$, f_V, T and length L)
  Output: A valid run $\boldsymbol{q}_{\overline{\mathcal{A}}}$ with length L in $\overline{\mathcal{A}}$
  Set $q_{cur} = q_0$ and $\boldsymbol{q}_{\overline{\mathcal{A}}} = (q_{cur})$;
  Set $T = F$ and $count = 0$;
  while $count \leq L$ **do**
    $q_{next} = \delta(q_{cur}, \alpha)$;
    if $q_{next} \in T$ or $q_{next} \notin F$ **then**
      Add state q_{next} to $\boldsymbol{q}_{\overline{\mathcal{A}}}$;
      $T = f_V(q', T)$;
    else
      No successor found at state q_{cur} and break the loop;
    end if
    $count++$;
    $q_{cur} \leftarrow q_{next}$;
  end while
end procedure

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 tracking the 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. 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 these sets should be removed from T. Based on (2), the E-LDGBA is constructed as follows.

Definition 3 (Embedded LDGBA)**.** Given a state-based LDGBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, its corresponding E-LDGBA is denoted by $\overline{\mathcal{A}} = \left(Q, \Sigma, \overline{\delta}, q_0, F, f_V, T\right)$ where T is initially set as $T = F$. The transition $\overline{\delta}\colon Q \times \Sigma \to 2^Q$ is defined as $q' = \overline{\delta}(q, \alpha)$ satisfying two conditions: 1) $q' = \delta(q, \alpha)$, and 2) $q' \notin F$ or $q' \in T$. Then, T is synchronously updated as $T = f_V(q', T)$ by (2) after each transition $q' = \overline{\delta}(q, \alpha)$.

Algorithm 1 shows the procedure of obtaining a valid run $\boldsymbol{q}_{\overline{\mathcal{A}}}$ (consisting of all visited states) within E-LDBGA. In Definition 3, $\overline{\delta}$ is designed to visit all non-visited accepting sets and avoid repeated visits of the same accepting sets in each round. It's obvious that E-LDGBA and LDGBA accept the same language. Compared with [18], the tracking-frontier function designed in (2) does not increase computational complexity when embedded in LDGBA. In the following analysis, we will use $\overline{\mathcal{A}}_\phi$ to denote the E-LDGBA corresponding to an LTL formula ϕ.

B. Probabilistic Product MDP

Definition 4. Given a PL-MDP $\mathcal{M}$ and an E-LDGBA $\overline{\mathcal{A}}_\phi$, the product MDP is defined as $\mathcal{P} = \mathcal{M} \times \overline{\mathcal{A}}_\phi = \left(X, U^{\mathcal{P}}, p^{\mathcal{P}}, x_0, F^{\mathcal{P}}\right)$, where $X = S \times 2^{\Pi} \times Q$ is the set of labeled states, i.e., $x = (s, l, q) \in X$ 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-actions are only allowed for transitions from Q_N to Q_D; $x_0 = (s_0, l_0, q_0)$ is the initial state; $F^{\mathcal{P}} = \left\{F_1^{\mathcal{P}}, F_2^{\mathcal{P}} \ldots F_f^{\mathcal{P}}\right\}$ where $F_j^{\mathcal{P}} = \left\{(s, l, q) \in X \middle| q \in F_j\right\}$, $j = 1, \ldots f$, 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.

The product MDP $\mathcal{P}$ captures the intersections between all feasible paths over $\mathcal{M}$ and all words accepted to $\mathcal{A}_\phi$, facilitating the identification of admissible agent motions that satisfy the 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 $\Pr_{max}[x \models \mathrm{Acc}_p] = \max\limits_{\boldsymbol{\pi}} \Pr_M^{\boldsymbol{\pi}}(\mathrm{Acc}_p)$.

Consider a sub-product 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, one MEC that does not contain any accepting sets is called a rejecting accepting component (RMEC) and a MEC with only partial accepting sets is called a neutral maximum end component (NMEC) [1].

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 maximum probability, i.e., $\Pr^{\boldsymbol{\pi}^*}[x \models \mathrm{Acc}_p] = \Pr_{max}[x \models \mathrm{Acc}_p]$.

C. Properties of Product MDP

Let $MC_{\mathcal{P}}^{\boldsymbol{\pi}}$ denote the Markov chain induced by a policy π on $\mathcal{P}$, whose states can be represented by a disjoint union of a transient class $\mathcal{T}_\pi$ and n closed irreducible recurrent classes $\mathcal{R}_\pi^j$, $j \in \{1, \ldots, n_R\}$ [22].

Lemma 1. *Given a product 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 [23], for each state in recurrent class, it holds that $\sum\limits_{n=0}^{\infty} p^n(x, x) = \infty$, where $x \in R_\pi^j \cap F_k^{\mathcal{P}}$ and $p^n(x, x)$ 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, neither $q_k \in F_k$ nor $x_k = (s, q_k) \in R_{\boldsymbol{\pi}}^j \cap F_k^{\mathcal{P}}$ with $s \in S$ will occur infinitely, which contradicts the property $\sum_{n=0}^{\infty} p^n(x_k, x_k) = \infty$. ■

Lemma 1 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

Due to the consideration of uncertain PL-MDP, model-free reinforcement learning is leveraged to identify policies for Problem 2.

A. Rewards Design

Let $F_U^{\mathcal{P}}$ denote the union of states, i.e., $F_U^{\mathcal{P}} = \left\{x \in X \left| x \in F_i^{\mathcal{P}}, \forall i \in \{1, \dots f\} \right|\right\}$. Borrowed from [16], the reward function is $R(x) = 1 - r_F$ if $x \in F_U^{\mathcal{P}}$ and $R(x) = 0$ otherwise. The discount function is $\gamma(x) = r_F$ if $x \in F_U^{\mathcal{P}}$, and $\gamma(x) = \gamma_F$ otherwise, where $r_F(\gamma_F)$ is a function of γ_F satisfying $\lim_{\gamma_F \to 1^-} r_F(\gamma_F) = 1$ and $\lim_{\gamma_F \to 1^-} \frac{1-\gamma_F}{1-r_F(\gamma_F)} = 0$.

Given a path $\boldsymbol{x}_t = x_t x_{t+1} \dots$ starting from x_t, the expected return is denoted by

$$\mathcal{D}(\boldsymbol{x}_t) := \sum_{i=0}^{\infty} \left(\prod_{j=0}^{i-1} \gamma(\boldsymbol{x}_t[t+j]) \cdot R(\boldsymbol{x}_t[t+i]) \right) \tag{3}$$

where $\prod_{j=0}^{-1} := 1$ and $\boldsymbol{x}_t[t+i]$ denotes the $(i+1)$th state in $\boldsymbol{x}_t$. Based on (3), the expected return of any state $x \in X$ under policy π can be defined as

$$U^{\boldsymbol{\pi}}(x) = \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t[t] = x \right|\right]. \tag{4}$$

Remark 1. In [16], LDBA was employed to solve a problem similar to Problem 1. However, due to the use of E-LDGBA, the accepting conditions are different and thus there exist three types of MEC, i.e., AMEC, RMEC, NMEC, in the product MDP. Consequently, Theorem 1 of [16] is no longer applicable to this work. Nevertheless, it can be verified Lemma 1 and Lemma 2 of [16] still hold for E-LDGBA. In addition, compared with [19], the reward function only depends on the properties of current states and does not increase computational complexity.

Lemma 2. *For any path $\boldsymbol{x}_t$ and $\mathcal{D}(\boldsymbol{x}_t)$ in (3), it holds that $0 \leq \gamma_F \cdot \mathcal{D}(\boldsymbol{x}_t[t+1:]) \leq \gamma_F \cdot \mathcal{D}(\boldsymbol{x}_t) \leq 1 + r_F + r_F \cdot \mathcal{D}(\boldsymbol{x}_t[t+1:]) \leq 1$, where $\boldsymbol{x}_t[t+1:]$ denotes the suffix of $\boldsymbol{x}_t$ starting from x_{t+1}. Let $AMEC_{\mathcal{P}}^{\boldsymbol{\pi}}(X_{AMEC}^{\boldsymbol{\pi}}, U_{AMEC}^{\boldsymbol{\pi}})$ denote the induced AMCE under a policy π over states $X_{AMEC}^{\boldsymbol{\pi}}$ whose actions are restricted to $U_{AMEC}^{\boldsymbol{\pi}}$. Let $X_{\mathcal{P}}^{\boldsymbol{\pi}}$ denote the set of states belonging to any of accepting sets in $AMEC_{\mathcal{P}}^{\boldsymbol{\pi}}$ such that $X_{\mathcal{P}}^{\boldsymbol{\pi}} := \left\{x \in X \left| x \in F_U^{\mathcal{P}} \cap X_{AMEC}^{\boldsymbol{\pi}} \right|\right\}$. Then, for any states $x \in X_{\mathcal{P}}^{\boldsymbol{\pi}}$, it holds that $\lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}}(x) = 1$.*

The proof of lemma 2 is omitted since it is a straightforward extension of Lemma 1 and Lemma 2 in [16] by replacing LDBA with E-LDGBA. Based on Lemma 2, we can establish the following theorem which is one of the main contributions in this paper.

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

$$\exists i \in \{1, \dots f\}, \lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}}(x) = \Pr^{\boldsymbol{\pi}}\left[\Diamond F_i^{\mathcal{P}}\right], \tag{5}$$

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 (4) can be rewritten as

$$\begin{aligned} U^{\boldsymbol{\pi}}(x) = & \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t \models \Diamond F_i^{\mathcal{P}} \right.\right] \cdot \Pr^{\boldsymbol{\pi}}\left[x \models \Diamond F_i^{\mathcal{P}}\right] \\ & + \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t \neq \Diamond F_i^{\mathcal{P}} \right.\right] \cdot \Pr^{\boldsymbol{\pi}}\left[x \not\models \Diamond F_i^{\mathcal{P}}\right] \end{aligned} \tag{6}$$

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 satisfying and not satisfying $F_i^{\mathcal{P}}$ eventually under policy π starting from state x, respectively.

To find the lower bound of $U^{\boldsymbol{\pi}}(x)$, for any $\boldsymbol{x}_t$ with $\boldsymbol{x}_t[t] = x$, let $t + N_t$ be the index that $\boldsymbol{x}_t$ first intersects a state in $X_{\mathcal{P}}^{\boldsymbol{\pi}} = \left\{x' \in X \left| x' \in F_U^{\mathcal{P}} \cap X_{AMEC}^{\boldsymbol{\pi}} \right|\right\}$, i.e., $N_t = \min\left[i \left| \boldsymbol{x}_t[t+i] \in X_{\mathcal{P}}^{\boldsymbol{\pi}} \right|\right]$. The following holds

$$\begin{aligned} & \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t \models \Diamond F_i^{\mathcal{P}} \right.\right] \\ & \overset{(1)}{\geq} \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t \cap F_U^{\mathcal{P}} \cap X_{AMEC}^{\boldsymbol{\pi}} \neq \emptyset \right.\right] \\ & \overset{(2)}{\geq} \mathbb{E}^{\boldsymbol{\pi}}\left[\gamma_F^{N_t} \cdot \mathcal{D}(\boldsymbol{x}_t[t+N_t:]) \left| \boldsymbol{x}_t[t+N_t] = x \right| \boldsymbol{x}_t \cap X_{\mathcal{P}}^{\boldsymbol{\pi}} \neq \emptyset\right] \\ & \overset{(3)}{\geq} \mathbb{E}^{\boldsymbol{\pi}}\left[\gamma_F^{N_t} \left| \boldsymbol{x}_t \cap X_{\mathcal{P}}^{\boldsymbol{\pi}} \neq \emptyset \right.\right] \cdot U_{\min}^{\boldsymbol{\pi}}(\boldsymbol{x}_t[t+N_t]) \\ & \overset{(4)}{\geq} \gamma_F^{\mathbb{E}^{\boldsymbol{\pi}}\left[N_t \left| \boldsymbol{x}_t[t]=x \right|, \boldsymbol{x}_t \cap F^{\mathcal{P}} \cap X_{AMEC}^{\boldsymbol{\pi}} \neq \emptyset\right]} \cdot U_{\min}^{\boldsymbol{\pi}}(x_{AMEC}) \\ & = \gamma_F^{n_t} \cdot U_{\min}^{\boldsymbol{\pi}}(x_{AMEC}) \end{aligned} \tag{7}$$

where $x_{AMEC} \in X_{\mathcal{P}}^{\boldsymbol{\pi}}$, $U_{\min}^{\boldsymbol{\pi}}(x_{AMEC}) = \min_{x \in X_{\mathcal{P}}^{\boldsymbol{\pi}}} U^{\boldsymbol{\pi}}(x)$, and n_t is a constant. By Lemma 2, one has $\lim_{\gamma_F \to 1^-} U_{\min}^{\boldsymbol{\pi}}(x_{AMEC}) = 1$. In (7), the first inequality holds because $X_{AMEC}^{\boldsymbol{\pi}}$ is one of the cases (e.g., $AMEC_{\mathcal{P}}^{\boldsymbol{\pi}}$ or $NMEC_{\mathcal{P}}^{\boldsymbol{\pi}}$ intersects $F_i^{\mathcal{P}}$ before entering $RMEC_{\mathcal{P}}^{\boldsymbol{\pi}}$) that satisfy $\boldsymbol{x}_t \models \Diamond F_i^{\mathcal{P}}$; the second inequality holds due to Lemma 2; the third inequality holds due to the Markov properties of (3) and (4); the fourth inequality holds due to Jensen's inequality. Based on (7), the lower bound of (6) is $U^{\boldsymbol{\pi}}(x) \geq \gamma_F^{n_t} \cdot U_{\min}^{\boldsymbol{\pi}}(x_{AMEC}) \cdot \Pr^{\boldsymbol{\pi}}\left[x \models \Diamond F_i^{\mathcal{P}}\right]$ from which one has

$$\lim_{\gamma_F \to 1^-} U^{\boldsymbol{\pi}}(x) \geq \gamma_F^{n_t} \cdot \Pr^{\boldsymbol{\pi}}\left[x \models \Diamond F_i^{\mathcal{P}}\right]. \tag{8}$$

Similarly, let $t + M_t$ denote the index that $\boldsymbol{x}_t$ first enters $RMEC_{\mathcal{P}}^{\boldsymbol{\pi}}$. We have

$$\begin{aligned} \mathbb{E}^{\boldsymbol{\pi}}\left[\mathcal{D}(\boldsymbol{x}_t) \left| \boldsymbol{x}_t \not\models \Diamond F_i^{\mathcal{P}} \right.\right] & \overset{(1)}{\leq} \mathbb{E}^{\boldsymbol{\pi}}\left[1 - r_F^{M_t} \left| \boldsymbol{x}_t \not\models \Diamond F_i^{\mathcal{P}} \right.\right] \\ & \overset{(2)}{\leq} 1 - r_F^{\mathbb{E}^{\boldsymbol{\pi}}\left[M_t \left| \boldsymbol{x}_t[t]=x, \boldsymbol{x}_t \not\models \Diamond F^{\mathcal{P}} \right.\right]} = 1 - r_F^{m_t} \end{aligned} \tag{9}$$

where m_t is a constant and (9) holds due to Lemma 2 and Markov properties.

Hence, the upper bound of (6) is obtained as

$$\lim_{\gamma_F\to 1^-} U^{\boldsymbol{\pi}}(x) \le \Pr{}^{\boldsymbol{\pi}}\left[x \models \Diamond F_i^{\mathcal{P}}\right] + (1-r_F^{m_t})\Pr{}^{\boldsymbol{\pi}}\left[x \not\models \Diamond F_i^{\mathcal{P}}\right]. \tag{10}$$

By (8) and (10), we can conclude

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

According to $\lim_{\gamma_F\to 1^-} r_F(\gamma_F) = 1$ in the reward function, (5) 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 (4) 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 π, $MC_{\mathcal{P}}^{\boldsymbol{\pi}} = \mathcal{T}_{\boldsymbol{\pi}} \sqcup \mathcal{R}_{\boldsymbol{\pi}}^1 \sqcup \mathcal{R}_{\boldsymbol{\pi}}^2 \ldots \mathcal{R}_{\boldsymbol{\pi}}^{n_R}$. Let $\boldsymbol{U}_\pi = \begin{bmatrix} U^\pi(x_0) & U^\pi(x_1) & \ldots \end{bmatrix}^T \in \mathbb{R}^{|X|}$ denote the stacked expected return under policy π, which can be reorganized as

$$\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{U}_{\boldsymbol{\pi}}^{rec} \end{bmatrix} = \sum_{n=0}^{\infty}\left(\prod_{j=0}^{n-1}\begin{bmatrix} \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{\mathcal{T}} & \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{0}_{\sum_{i=1}^m N_i\times r} & \boldsymbol{\gamma}_{\boldsymbol{\pi}}^{rec} \end{bmatrix}\right) \\ \cdot \begin{bmatrix} \boldsymbol{P}_{\boldsymbol{\pi}}(\mathcal{T},\mathcal{T}) & \boldsymbol{P}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{0}_{\sum_{i=1}^m N_i\times r} & \boldsymbol{P}_{\boldsymbol{\pi}}(\mathcal{R},\mathcal{R}) \end{bmatrix}^n \begin{bmatrix} \boldsymbol{R}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{R}_{\boldsymbol{\pi}}^{rec} \end{bmatrix}, \tag{11}$$

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 (11), $\boldsymbol{P}_{\boldsymbol{\pi}}(\mathcal{T},\mathcal{T}) \in \mathbb{R}^{r\times r}$ 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 $\boldsymbol{P}_{\boldsymbol{\pi}}(\mathcal{R},\mathcal{R})$ is a diagonal block matrix, where the ith block is a $N_i \times N_i$ matrix containing transition probabilities between states within $\mathcal{R}_{\boldsymbol{\pi}}^i$. Note that $\boldsymbol{P}_{\boldsymbol{\pi}}(\mathcal{R},\mathcal{R})$ is a stochastic matrix since each block matrix is a stochastic matrix [23]. 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 π^* that optimizes the expected return, but not satisfy the accepting condition of $\mathcal{P}$. Based on Lemma 1, 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 π^* 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 $\boldsymbol{P}_{\boldsymbol{\pi}^*}^n(\mathcal{R},\mathcal{R})$ 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 π^* is then obtained from (11) as

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

where $k_1 = \sum_{i=1}^{j-1} N_i$, $k_2 = \sum_{i=j+1}^{n} N_i$. Since $\mathcal{R}_{\pi^*}^j \cap F_i^{\mathcal{P}} = \emptyset, \forall i \in \{1,\ldots f\}$, by the designed reward function, all entries of $\boldsymbol{R}_{\boldsymbol{\pi}^*}^{rec}$ are zero. We can conclude $U_{\boldsymbol{\pi}^*}^{rec}(x_R) = 0$. To show contradiction, the following analysis will show that $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) > U_{\boldsymbol{\pi}^*}^{rec}(x_R)$ for any policy $\bar{\pi}$ that satisfies the accepting condition of $\mathcal{R}$. Thus, it's true that there exists $\mathcal{R}_{\bar{\pi}}^j$ such that $\mathcal{R}_{\pi^*}^j \cap F_k^{\mathcal{P}} \neq \emptyset, \forall k\{1,\ldots f\}$. 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 1, the entries in $\boldsymbol{R}_{\bar{\pi}}^{rec}$ corresponding to the recurrent states in $\mathcal{R}_{\bar{\pi}}^j$ 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 (11), $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R)$ can be lower bounded as

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

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 $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) > U_{\boldsymbol{\pi}^*}^{rec}(x_R)$.

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 [23], for a transient state $x_{tr} \in \mathcal{T}_{\bar{\pi}}$, there always exists an upper bound $\Delta < \infty$ such that $\sum_{n=0}^{\infty} p^n(x_{tr},x_{tr}) < \Delta$, where $p^n(x_{tr},x_{tr})$ 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

$$\sum_{n=0}^{\infty}\gamma^n p^n(x_{rec},x_{rec}) > \frac{1}{1-\gamma^{\overline{n}}}\bar{p}, \tag{12}$$

where there exists $\overline{n}$ such that $p^{\overline{n}}(x_{rec},x_{rec})$ is nonzero and can be lower bounded by $\bar{p}$ [23]. From (11), one has

$$\boldsymbol{U}_{\bar{\boldsymbol{\pi}}}^{tr} > \sum_{n=0}^{\infty}\left(\prod_{j=0}^{n-1}\boldsymbol{\gamma}_{\bar{\boldsymbol{\pi}}}^{tr}\right).\boldsymbol{P}_{\bar{\pi}}^{tr}\boldsymbol{P}_{\bar{\pi}}^{n}(\mathcal{R},\mathcal{R})\boldsymbol{R}_{\bar{\pi}}^{rec} \\ > \underline{\gamma}^n.\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr}\boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{n}(\mathcal{R},\mathcal{R})\boldsymbol{R}_{\bar{\boldsymbol{\pi}}}^{rec}. \tag{13}$$

Let $\max(\cdot)$ and $\min(\cdot)$ represent the maximum and minimum entry of an input vector, respectively. The upper bound $\bar{m} = \left\{\max\left(\overline{M}\right)\middle|\overline{M} < \boldsymbol{P}_{\bar{\pi}}^{tr}\bar{\boldsymbol{P}}\boldsymbol{R}_{\bar{\pi}}^{rec}\right\}$ and $\bar{m} \ge 0$, where $\bar{\boldsymbol{P}}$ is a block matrix whose nonzero entries are derived similarly to $\bar{p}$ in (12). The utility $U_{\bar{\pi}}^{tr}(x_R)$ can be lower bounded from (12) and (13) as

$$U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > \frac{1}{1-\underline{\gamma}^n}\bar{m}. \tag{14}$$

Since $U_{\boldsymbol{\pi}^*}^{rec}(x_R) = 0$, the contradiction $U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > 0$ 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 $U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > \frac{1}{1-\underline{\gamma}^n}\bar{m} \ge 0$. The procedure shows the contradiction of the assumption that π^* 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 (4) 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 (4) of the corresponding product MDP also maximizes the probability of satisfying ϕ, i.e., $\Pr^{\boldsymbol{\pi}^*}\left[x \models Acc_{\mathcal{P}}\right] = \Pr_{max}\left[x \models Acc_{\mathcal{P}}\right]$.*

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

$$\begin{aligned} Q\left(x, u^{\mathcal{P}}\right) \leftarrow & (1-\alpha) Q\left(x, u^{\mathcal{P}}\right) \\ & + \alpha\left[R(x) + \gamma(x) \cdot \max_{u^{\mathcal{P}} \in U^{\mathcal{P}}} Q\left(x, u^{\mathcal{P}}\right)\right], \end{aligned} \tag{15}$$

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. The number of states is $|S| \times |Q|\Big|$, where $|Q|$ is determined by the original LDGBA $\mathcal{A}_{\phi}$ because the construction of E-LDGBA $\overline{\mathcal{A}}_{\phi}$ will not increase the size, and $|S|$ is the size of the PL-MDP model. As for product MDP, the complexity of actions available at each state $x = (s, l, q)$ is $O(|A(s)|)$.

Algorithm 2 Reinforcement Learning

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 $Q\left(x, u^{\mathcal{P}}\right) = 0$ for all x and $u^{\mathcal{P}} \in U^{\mathcal{P}}(x)$
5: $Count\left(x, u^{\mathcal{P}}\right) = 0$ for all $u^{\mathcal{P}} \in U^{\mathcal{P}}(x)$
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} based on epsilon-greedy selection
14: Obtain a_{curr} of $\mathcal{M}$ from u_{curr}
15: Execute u_{curr} and observer x_{next}, $R(x_{curr})$,$\gamma(x_{curr})$
16: $r \leftarrow R(x_{curr})$ and $\gamma \leftarrow \gamma(x_{curr})$
17: $Count(x_{curr}, u_{curr})++$
18: $\alpha = 1/Count(x_{curr}, u_{curr})$
19: $Q(x_{curr}, u_{curr}) \leftarrow (1-\alpha) Q(x_{curr}, u_{curr}) + \alpha\left[r + \gamma \cdot \max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}} Q\left(x_{next}, u^{\mathcal{P}}\right)\right]$
20: $x_{curr} = x_{next}$
21: **end while**
22: **end while**
23: **for** all $x \in X$ **do**
24: $U(x) = \max\limits_{u^{\mathcal{R}} \in U^{\mathcal{R}}} Q\left(x, u^{\mathcal{P}}\right)$
25: $\boldsymbol{\pi}^*(x) = \arg\max\limits_{u^{\mathcal{P}} \in U^{\mathcal{P}}} U(x)$
26: **end for**
27: **end procedure**

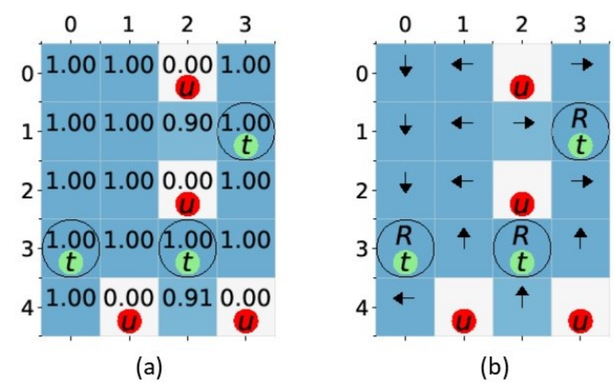

Figure 1. (a) The estimated maximal satisfaction probability., where the targets and unsafe areas are denoted by t and u, respectively. (b) The optimal policy of satisfying φ_{case1}, where "R" is an action primitive means the robot remains at its current cell.

VI. Case Studies

The developed RL-based control synthesis is implemented in Python. Owl [21] is used to convert LTL specifications to LDGBA. All simulations are carried out on a laptop with 2.60 GHz quad-core CPU and 8 GB of RAM. For Q-learning, the optimal policies of each case are generated using 10^5 episodes with random initial states. The discount factors that determine the reward function are selected as $r_F = 0.99$ and $\gamma_F = 0.9999$ for all cases, and the learning rate α is determined by Alg. 2. To validate the effectiveness of our approach, we first carry out simulations over grid environments and then validate the approach in a more realistic office scenario with TurtleBot3 robot.[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. 1 and Fig. 2. 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. The resulting PL-MDP model has 25 states.

[1]Any other model-free RL algorithm can also be adopted with Alg.2

[2]In SectionIV, the examples of LDGBA and LDBA are shown in state-based form for demonstration purposes. As for case studies, the optimized transition-based form via Owl [21] is used.

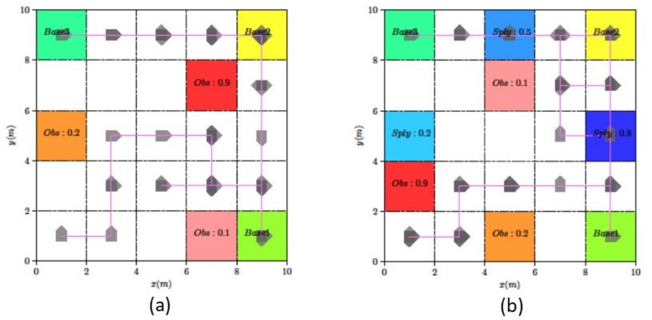

Figure 2. Simulated trajectories of 25 time steps under the corresponding optimal policies.

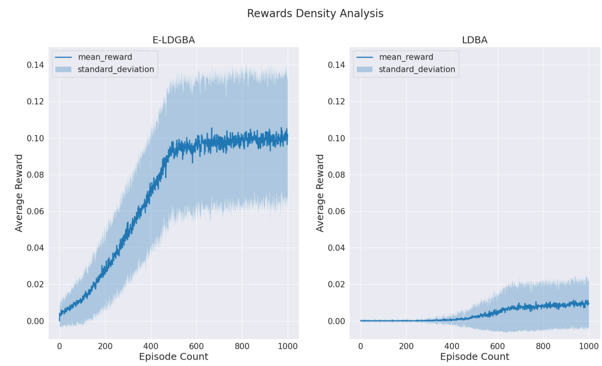

Figure 3. We perform 100 learning iterations for 1000 episodes and each episode terminates after $\tau = 100$ steps. The mean of rewards is obtained using E-LDGBA (left) and LDBA (right) respectively. The ranges of standard deviations are represented in light blue.

(1) Maximum Satisfaction Probability: In this case, the objective is to verify that the policy generated satisfies the given LTL specification with maximum probability. The package Csrl in [16] is used. The LTL specification is

$$\varphi_{case1} = \Diamond \Box \, \mathtt{t} \wedge \Box \neg \mathtt{u}, \tag{16}$$

which requires the robot to eventually arriving at one of the targets $\mathtt{t}$ while avoiding unsafe areas $\mathtt{u}$. The corresponding LDGBA has 1 state with 1 accepting set and the product MDP has 20 states. Each episode terminates after $\tau = 100$ steps. Fig. 1 (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. 1 (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 an infinite horizon. As shown in Fig. 2, 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 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. 2 (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 while avoiding the obstacles. Its corresponding LDGBA has 2 states with 3 accepting sets and the product MDP has 50 states. In this case, each episode terminates after $\tau = 100$ steps. The generated optimal trajectory is shown in Fig. 2 (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. Its corresponding LDGBA has 24 states with 4 accepting sets and the product MDP has 600 states. In this case, each episode terminates after $\tau = 800$ steps. The generated optimal trajectory is shown in Fig. 2 (b).

(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. 3 shows the mean and standard deviations of rewards collected using E-LDGBA and LDBA, respectively, which indicates E-LDGBA based method converges faster since the accepting set is tracked every time the robot visits one of the base stations, while 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.

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

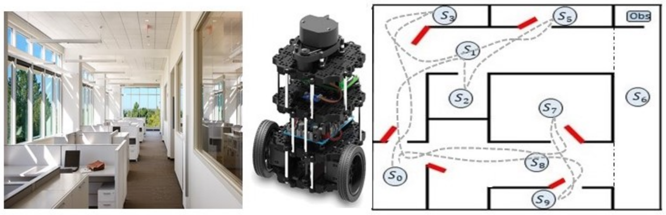

Figure 4. The mock-up office scenario with the TurtleBot3 robot.

B. Experimental Results

Consider an office environment constructed in ROS Gazebo as shown in Fig.4, which consists of 7 rooms denoted by $S_0, S_2, S_3, S_5, S_7, S_9, \texttt{Obs}$ 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. The TurtleBot3 starting from room S_0 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 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, and the robot can successfully moving between corridors with probability 1.0. The resulting PL-MDP has 12 states. The service to be performed by TurtleBot3 is expressed as

$$\varphi_{case4} = \varphi_{all} \wedge \square\neg\texttt{Obs}, \tag{17}$$

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 (17), 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 $\texttt{Obs}$. Its corresponding LDGBA has 5 states with 5 accepting sets and the product MDP has 72 states.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. 4. 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 uncertainties, resulting in the violation of φ_{case3}.

VII. Conclusion

In this paper, a model-free learning-based algorithm is developed to synthesize control policies that maximize the satisfaction of LTL specifications. The LTL specifications are translated to E-LDGBA to synchronously record non–visited accepting sets to improve the performance and maintain dense rewards without increasing dimensional and computational complexity. Future research will focus on deep RL to address continuous state spaces and actions.

References

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

[2] Z. L. H. G. Z. K. Mingyu Cai, Hao Peng, "Receding horizon control based online motion planning with partially infeasible ltl specifications," *arXiv preprint arXiv:2007.12123*, 2020.

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

[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," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1098–1123, 2019.

[7] M. Cai, Z. Li, H. Gao, 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," *Machine learning*, 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," *Science Robotics*, 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," *arXiv preprint arXiv:1909.07299*, 2019.

[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," *arXiv preprint arXiv:1909.05304*, 2019.

[18] R. Oura, A. Sakakibara, and T. Ushio, "Reinforcement learning of control policy for linear temporal logic specifications using limit-deterministic b\" uchi automata," *arXiv preprint arXiv:2001.04669*, 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] 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.

[21] 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

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

[23] R. Durrett, *Essentials of stochastic processes, 2nd ed.* Springer, 2012, vol. 1.