Journal of Automation and Intelligence 4 (2025) 39–51

Contents lists available at ScienceDirect

Journal of Automation and Intelligence

journal homepage: www.keaipublishing.com/en/journals/journal-of-automation-and-intelligence/

Research article

Reinforcement learning with soft temporal logic constraints using limit-deterministic generalized Büchi automaton

Mingyu Cai [a], Zhangli Zhou [b], Lin Li [b], Shaoping Xiao [c], Zhen Kan [b,*]

[a] University of California, Riverside, CA, 92521, USA
[b] University of Science and Technology of China, Hefei, 230026, China
[c] The University of Iowa, IA, 52246, USA

ARTICLE INFO

Keywords:
Formal methods in robotics and automation
Linear temporal logic
Motion planning
Optimal control

ABSTRACT

This paper investigates control synthesis for motion planning under conditions of uncertainty, specifically in robot motion and environmental properties, which are modeled using a probabilistic labeled Markov decision process (PL-MDP). To address this, a model-free reinforcement learning (RL) approach is designed to produce a finite-memory control policy that meets complex tasks specified by linear temporal logic (LTL) formulas. Recognizing the presence of uncertainties and potentially conflicting objectives, this study centers on addressing infeasible LTL specifications. A relaxed LTL constraint enables the agent to adapt its motion plan, allowing for partial satisfaction by accounting for necessary task violations. Additionally, a new automaton structure is introduced to increase the density of accepting rewards, facilitating deterministic policy outcomes. The proposed RL framework is rigorously analyzed and prioritizes two key objectives: (1) satisfying the acceptance condition of the relaxed product MDP, and (2) minimizing long-term violation costs. Simulation and experimental results are presented to demonstrate the framework's effectiveness and robustness.

1. Introduction

Formal logic enables the specification of complex, high-level tasks that go beyond traditional goal-directed navigation in robotic systems. As a formal language, linear temporal logic (LTL) has become increasingly popular for use in robotic motion planning [1–7]. Since robotic systems are often subject to a variety of uncertainties arising from the stochastic behaviors of the motion (e.g., an agent may not exactly follow the control inputs due to potential sensing noise or actuation failures) and uncertain environment properties (e.g., there exist mobile obstacles or time-varying areas of interest), Markov decision processes (MDPs) are often used to model the probabilistic motion of robotic systems [8,9]. Based on probabilistic model checking, control synthesis of MDP with LTL motion specifications has been widely investigated (cf. [10–15]). In particular, the topic of partial satisfaction of high-level tasks in deterministic and stochastic systems is investigated in [14,16–19]. Yet, new challenges arise when considering motion and environment uncertainties. Hence, learning to find a satisfying policy is paramount for the robot to operate in the presence of motion and environment uncertainties.

Reinforcement learning (RL) is a sequential decision-making process in which an agent continuously interacts with and learns from the environment [20–23]. Model-based RL has been employed for motion planning with LTL specifications when full knowledge of MDP is available [24]. The work of [25] extends model-based RL to temporal logic constrained control of stochastic systems with unknown parameters by model approximation. In [26,27], transition probabilities are learned to facilitate the satisfaction of LTL specifications. However, these aforementioned works have to reply on the accuracy of transition probabilities for learning. On the other hand, model-free RL approaches with LTL-based rewards generate desired policies by directly optimizing the Q-values [28–31]. However, these approaches rely on the key assumption that at least one accepting maximum end component (AMEC) exists in a standard product MDP [1], which may not hold in practical scenarios. For example, certain areas of interest may be probabilistically inaccessible to the agent (e.g., areas that become unreachable due to flooding, preventing traversal by a ground robot). This can result in portions of the user-specified tasks being unachievable, leading to the absence of AMECs in the product MDP. Although minimal revision of motion plans in a potentially conflicting environment has been investigated in the works of [32–35], only deterministic transition systems are considered, and it is not yet clear how to address the above-mentioned issues in stochastic systems i.e., MDP.

Peer review under responsibility of Chongqing University.
* Corresponding author.
E-mail addresses: mingyu.cai@ucr.edu (M. Cai), zhangli@ustc.edu.cn (Z. Zhou), linlee1006@mail.ustc.edu.cn (L. Li), shaoping-xiao@uiowa.edu (S. Xiao), zkan@ustc.edu.cn (Z. Kan).

https://doi.org/10.1016/j.jai.2024.12.005
Received 26 August 2024; Received in revised form 28 November 2024; Accepted 23 December 2024
Available online 31 December 2024

Related Works: Compared with most RL-based approaches [26,30,31,36,37], the reward functions are generally designed to enforce the convergence to AMECs so that the acceptance condition can be satisfied. However, these approaches are not applicable in this work due to the fact that, despite the existence of AMECs, they may not be reachable due to environmental uncertainties. As a result, the probability of achieving the specified tasks can become negligible. To address this, the approach in [38] introduces a reduced-variance deep Q-learning technique aimed at approximating optimal state–action values. Additionally, our prior work [39] constructs a relaxed product MDP by converting LTL formulas into a Deterministic Rabin Automaton (DRA) without relying on the existence of AMECs. However, [30] suggests that converting LTL into a DRA may not reliably yield the desired policies. Even though our recent work [40] shows how to leverage planning algorithms to guide minimally-violating policies for infeasible LTL tasks. But multi-objective reinforcement learning without the assistance of planning is still under investigation.

In [41], the LDGBA is applied, and a synchronous frontier function for RL reward is designed to synthesize control policies that mostly fulfill the given task by maximizing the visits of the accepting sets in LDGBA. However, it cannot record the visited or non-visited accepting sets in each round. Compared to the modular framework in [42], which designs deep deterministic policy gradients to handle continuous spaces but lacks explicit mechanisms to record unvisited automaton states, our method leverages E-LDGBA to ensure denser reward signals and more efficient policy optimization. Unlike the mission-driven exploration strategy in [43], which accelerates RL by prioritizing task-relevant state exploration, our approach provides a more robust mechanism to handle task infeasibility through a relaxed product MDP. Furthermore, in contrast to the certified RL algorithm in [44], which shapes rewards based on deterministic automata to ensure probabilistic satisfaction, we optimize multi-objective RL with explicit violation cost analysis, allowing partial satisfaction in stochastic environments. As an extension of [39], this work can handle the above issues by developing an E-LDGBA to record the non-visited accepting sets without adding extra computational complexity. The proposed relaxed product MDP and the designed utility function transforms the control synthesis problem into an expected utility optimization problem, in which a satisfactory policy is guaranteed to be found by leveraging RL to optimize the expected utility. LDGBA reduces the number of states through a simplified automata structure, avoiding the state–space overload problem common in traditional DRA transitions.

LDGBA formulates LTL formulas via limit deterministic structures and generalized Büchi acceptance conditions. Compared to other automaton, LDGBA typically generates a smaller state space, which is computationally more efficient. For many LTL formulas, LDGBA is an exponentially-sized automaton [45]. The size of the state space of LDGBA is typically similar to that of Büchi automaton and significantly smaller than that of Rabin automaton because it mitigates the state space expansion to some extent. The state space of Rabin automaton tends to be larger due to the complexity of their acceptance conditions, which typically need to handle LTL to Rabin automaton transitions on the double exponential level. Besides, LDGBA has more expressive power compared to Büchi automaton and is able to handle more complex LTL tasks. While LDGBA has a smaller state space than Rabin automaton and can express most LTL expressions. Moreover, the model-free RL-based approach is adopted and can synthesize optimal policies on-the-fly without explicitly memorizing structures of relaxed product MDP.

Contributions: Motivated by these challenges, this paper considers the transition probabilities of interactions between the environment and the mobile robot to be unknown. We study learning-based motion planning subject to uncertainties, where control objectives are defined as high-level LTL formulas to express complex tasks. The contributions of this work are multi-fold. (i). From the aspect of automaton theory, we leverage an embedded limit deterministic generalized Büchi automaton (E-LDGBA) that has several accepting sets to maintain a dense reward and allows applying deterministic policies to achieve high-level objectives. (ii) Both motion and workspace uncertainties are considered to be unknown, leading to potentially conflicting tasks (i.e., the pre-specified LTL tasks cannot be fully satisfied). We design a relaxed product MDP from the PL-MDP and the novel automaton so that the RL agent can revise its motion plan without strictly following the desired LTL constraints. (iii) An expected return is defined, incorporating both violation rewards and acceptance rewards. The violation function is designed to measure the deviation between the adjusted and intended motion plans, while the acceptance rewards promote fulfillment of the acceptance condition in the relaxed product MDP. (iv) Rigorous analysis is provided to show how to properly design parameters of accepting and violating rewards for multi-objective RL (MORL). Based on that, the RL agent can find policies that fulfill pre-specified tasks as much as possible.

2. Preliminaries

2.1. Probabilistic labeled MDP

In real-world scenarios, uncertainties in robot motion and environmental characteristics can significantly impact a robotic system's performance and reliability. For instance, motion uncertainties may result in deviations from planned paths, necessitating robust corrective strategies, while environmental uncertainties can affect robots' perception and decision-making. Markov decision processes are often used to model these uncertain interactions between robots and the environment and are defined as follows:

Definition 1. A probabilistic labeled Markov decision process (PL-MDP) is defined as a tuple $\mathcal{M} = (S, A, p_S, (s_0, l_0), \Pi, L, p_L, \Lambda)$, where S represents a finite set of states, and A is the finite set of actions. The transition probability function $p_S : S \times A \times S \to [0,1]$ governs the likelihood of moving between states based on actions. Π denotes a set of atomic propositions, with $L \to 2^{\Pi}$ serving as the labeling function. Let ξ be an action function, which can be either deterministic such that $\xi : S \to A$ maps a state $s \in S$ to an action in $A(s)$, or randomized such that $\xi : S \times A \to [0,1]$ represents the probability of taking an action in $A(s)$ at s. The pair (s_0, l_0) denotes an initial state $s_0 \in S$ with an initial label $l_0 \in L(s_0)$. The function $p_L(s,l)$ denotes the probability of $l \in L(s)$ 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 the agent can fully observe its current state and the associated labels. The PL-MDP can be regarded as an advanced MDP, and we can use the MDP to denote it in the following sections. The MDP $\mathcal{M}$ evolves by taking actions ξ_i at each stage i, where $i \in \mathbb{N}_0$ with $\mathbb{N}$ being the set of natural numbers.

Definition 2. A control policy $\boldsymbol{\mu} = \mu_0 \mu_1 \ldots$ is a sequence of decision rules that generates a path $s = s_0 s_1 s_2 \ldots$ over $\mathcal{M}$, with $p_S(s_i, a_i, s_{i+1}) > 0$ for all i. The policy $\boldsymbol{\mu}$ can be either deterministic, where $\boldsymbol{\mu} : S \to A$, or stochastic, where $\boldsymbol{\mu} : S \times A \to [0,1]$. The control policy $\boldsymbol{\mu}$ is termed memoryless if each μ_i depends only on the current state s_i. Conversely, $\boldsymbol{\mu}$ is called a finite-memory, or history-dependent, policy if each μ_i considers past states.

This work shows how to apply the deterministic and memoryless policy that has more stable decision-making performance. Let $\boldsymbol{\mu}(s)$ denote the probability distribution of actions at state s, and $\boldsymbol{\mu}(s,a)$ represents the probability of generating action a at state s using the policy $\boldsymbol{\mu}$. Let $\Lambda : S \to \mathbb{R}$ 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

$$U^{\xi}(s) = \mathbb{E}^{\xi}\left[\sum_{i=0}^{\infty} \gamma^i \Lambda(s_i) \,\middle|\, s_0 = s\right] \tag{1}$$

The optimal policy ξ^* is a policy that maximizes the expected return for each state $s \in S$ as $\xi^* = \arg\max_{\xi} U^{\xi}(s)$.

Definition 3. Given a PL-MDP $\mathcal{M}$ under policy $\boldsymbol{\pi}$, a Markov chain $MC^{\boldsymbol{\mu}}_{\mathcal{M}}$ of the PL-MDP $\mathcal{M}$ induced by a policy $\boldsymbol{\mu}$ is a tuple $(S, A, p^{\boldsymbol{\mu}}_S, (s_0, l_0), L, p_L)$ where $p^{\boldsymbol{\mu}}_S(s, s') = p_S(s, a, s')$ with $\boldsymbol{\mu}(s, a) > 0$ for all $s, s' \in S$.

A sub-MDP $\mathcal{M}_{(S',A')}$ of $\mathcal{M}$ is a pair (S', A') where $S' \subseteq S$ and A' is a finite action space of S' such that (i) $S' \neq \emptyset$, and $A'(s) \neq \emptyset, \forall s \in S'$; (ii) $\{s' \mid \forall s \in S' \text{ and } \forall a \in A'(x), p^{\mathcal{P}}(s, a, s') > 0\} \subseteq S'$. An induced graph of $\mathcal{M}_{(S',A')}$ is denoted as $\mathcal{G}_{(S',A')}$ that is a directed graph, where if $p_S(s, a, s') > 0$ with $a \in A'(s)$, for any $s, s' \in S'$, there exists an edge between s and s' in $\mathcal{G}_{(S',A')}$. Note the evolution of a sub-MDP $\mathcal{M}_{(S',A')}$ is restricted by the action space A'. A sub-MDP qualifies as a strongly connected component (SCC) if its induced graph is strongly connected, meaning that for any pair of nodes $s, s' \in S'$, a path exists from s to s'. A bottom strongly connected component (BSCC) is an SCC from which no external states can be reached under the restricted action space. For further details on MDP treatments, please refer to [1].

Definition 4 (*[1]*)**.** A sub-MDP $\mathcal{M}_{(S',A')}$ is an end component (EC) of $\mathcal{M}$ if it is a BSCC. An EC $\mathcal{M}_{(S',A')}$ is called a maximal end component (MEC) if there is no other EC $\mathcal{M}_{(S'',A'')}$ such that $S' \subseteq S''$ and $A'(s) \subseteq A''(s)$, $\forall s \in S$.

2.2. LTL and limit-deterministic generalized Büchi automaton

Linear temporal logic (LTL) is a formal language used to specify high-level requirements for a system. An LTL formula is constructed from a set of atomic propositions, e.g., $a \in AP$, standard Boolean operators like $\wedge$ (and), $\neg$ (not), along with temporal operators: $\mathcal{U}$ (until), $\bigcirc$ (next), $\Diamond$ (eventually), and $\Box$ (always). The syntax of an LTL formula is defined inductively as follows:

$$\phi := \text{True} \mid a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \phi \mid \phi_1 \mathcal{U} \phi_2 \,,$$

The semantics of an Linear Temporal Logic formula are evaluated over words, which are infinite sequences $\boldsymbol{o} = o_0 o_1 \ldots$, where $o_i \in 2^{AP}$ for all $i \geq 0$. Here, 2^{AP} denotes the power set of AP, defined as follows:

$$\begin{aligned}
&\boldsymbol{o} \vDash \text{true} \\
&\boldsymbol{o} \vDash \alpha &&\Leftrightarrow \alpha \in L(\boldsymbol{o}[0]) \\
&\boldsymbol{o} \vDash \phi_1 \wedge \phi_2 &&\Leftrightarrow \boldsymbol{o} \vDash \phi_1 \text{ and } \boldsymbol{o} \vDash \phi_2 \\
&\boldsymbol{o} \vDash \neg\phi &&\Leftrightarrow \boldsymbol{o} \not\vDash \phi \\
&\boldsymbol{o} \vDash \phi &&\Leftrightarrow \boldsymbol{o}[1:] \vDash \phi \\
&\boldsymbol{o} \vDash \phi_1 \mathcal{U} \phi_2 &&\Leftrightarrow \exists t \text{ s.t. } \boldsymbol{o}[t:] \vDash \phi_2, \forall t' \in [0, t), \boldsymbol{o}[t':] \vDash \phi_1
\end{aligned}$$

Denote by $\boldsymbol{o} \vDash \varphi$ if the word $\boldsymbol{o}$ satisfies the LTL formula φ. More expressions can be achieved by combining temporal and Boolean operators. Detailed descriptions of the syntax and semantics of LTL can be found in [1]. Given an LTL formula that specifies the missions, the satisfaction of the LTL formula can be evaluated by an LDGBA [45].

Definition 5. 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 a transition function, $q_0 \in Q$ is an initial state, and $F = \{F_1, F_2, \ldots, F_f\}$ is a set of acceptance conditions with $F_i \subseteq Q$, $\forall i \in \{1, \ldots, f\}$.

Denote by $\boldsymbol{q} = q_0 q_1 \ldots$ 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 accepting sets, i.e., $\inf(\boldsymbol{q}) \cap F_i \neq \emptyset$, $\forall i \in \{1, \ldots, f\}$, where $\inf(\boldsymbol{q})$ denotes the set of states that is visited infinitely often.

Definition 6. A Generalized Büchi Automaton (GBA) is called a Limit-deterministic Generalized Büchi Automaton (LDGBA) if the transition function $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ is defined, and the state set Q can be partitioned into a deterministic set Q_D and a non-deterministic set Q_N, such that $Q = Q_D \cup Q_N$ and $Q_D \cap Q_N = \emptyset$, where:

- State transitions within Q_D are total and restricted to Q_D, i.e., $\left|\delta(q, \alpha)\right| = 1$ and $\delta(q, \alpha) \subseteq Q_D$ for every state $q \in Q_D$ and input $\alpha \in \Sigma$,
- ϵ-transitions are only allowed from states in Q_N to states in Q_D and are not permitted within Q_D, i.e., for any $q \in Q_D$, $\delta(q, \epsilon) = \emptyset$,
- All accepting states are contained within Q_D, i.e., $F_i \subseteq Q_D$ for every $F_i \in F$.

In Definition 6, the ϵ-transitions are only defined for state transitions from Q_N to Q_D that do not consume the input atomic proposition. Readers are referred to [46] for algorithms with free implementations to convert an LTL formula to an LDGBA,. Note that the state-based LDGBA is used in this work for demonstration purposes. Transition-based LDGBA can be constructed based on basic graph transformations. More discussions about state-based and transition-based LDGBA in HOA format can be found in Owl [47].

Remark 1. Unlike the widely used deterministic Rabin Automaton (DRA), LDGBA has the Generalized Büchi Accepting condition that purely involves reachability problems. On the other hand, compared with Limit-Deterministic Büchi Automaton (LDBA) applied in [30,31], LDGBA has more accepting sets shown in Fig. 1(a) to increase the density of rewards since the positive rewards are always assigned to the accepting states to enforce the acceptance condition.

3. Problem statement and challenge

The task specification to be performed by the agent is described by an LTL formula ϕ over Π. Given ϕ, $\mathcal{M}$, and $\boldsymbol{\xi} = \xi_0 \xi_1 \ldots$, the induced infinite path is denoted by $\boldsymbol{s}^{\boldsymbol{\xi}}_{\infty} = s_0 \ldots s_i s_{i+1} \ldots$ that satisfies $s_{i+1} \in \left\{ s \in S \middle| p_S(s_i, a_i, s) > 0 \right\}$. Let $L(\boldsymbol{s}^{\boldsymbol{\xi}}_{\infty}) = l_0 l_1 \ldots$ be the sequence of labels associated with $\boldsymbol{s}^{\boldsymbol{\xi}}_{\infty}$ such that $l_i \in L(s_i)$ and $p_L(s_i, l_i) > 0$. Denote by $L(\boldsymbol{s}^{\boldsymbol{\xi}}_{\infty}) \vDash \phi$ if $\boldsymbol{s}^{\boldsymbol{\xi}}_{\infty}$ satisfies ϕ. The satisfaction probability under $\boldsymbol{\mu}$ from an initial state s_0 can be computed as

$$\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = \Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\boldsymbol{s}^{\boldsymbol{\mu}}_{\infty} \in \boldsymbol{S}^{\boldsymbol{\mu}}_{\infty} \mid L(\boldsymbol{s}^{\boldsymbol{\mu}}_{\infty}) \vDash \phi), \tag{2}$$

where $\boldsymbol{S}^{\boldsymbol{\mu}}_{\infty}$ is a set of all admissible paths under policy $\boldsymbol{\mu}$, and the computation of $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\boldsymbol{s}^{\boldsymbol{\mu}}_{\infty})$ can be found in [1].

Definition 7. Given a PL-MDP $\mathcal{M}$, an LTL task ϕ is fully feasible if and only if $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) > 0$ s.t. there exists a path $\boldsymbol{s}^{\boldsymbol{\mu}}_{\infty}$ over the infinite horizons under the policy $\boldsymbol{\mu}$ satisfying ϕ.

Based on Definition 7, an infeasible case means there does not exist any policy μ to satisfy the task i.e., $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = 0$.

Example 1. To illustrate an infeasible case, the environment, its abstracted PL-MDP, and the LDGBA of an LTL formula $\phi = \Box\Diamond \mathfrak{a} \wedge \Box\Diamond \mathfrak{b}$ are shown in Fig. 1(a) and (b), respectively. Since obstacles surround region s_3, the given LTL task is infeasible in this case.

We define the expected discount violation cost of task satisfaction as follows, considering feasible and infeasible tasks.

Definition 8. Given a PL-MDP $\mathcal{M}$ and an LTL task ϕ, the discount expected violation under the policy $\boldsymbol{\mu}$ is defined as

$$J_V(\mathcal{M}^{\boldsymbol{\mu}}, \phi) = \mathbb{E}^{\boldsymbol{\mu}}_{\mathcal{M}} \left[\sum_{i=0}^{\infty} \gamma^i c_V(s_i, a_i, s_{i+1}, \phi) \right], \tag{3}$$

where $c_V(s, a, s', \phi)$ is defined as the violation cost of a transition (s, a, s') with respect to ϕ, and a_i is the action generated based on the policy $\boldsymbol{\mu}(s_i)$.

Most existing results (cf. [26,30,31]) assume the existence of at least one policy μ s.t. $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) > 0$, which may not be true in practice if the task is only partially feasible. In addition, previous work [18] addresses the infeasible cases via an optimization approach that assumes the

transitions probabilities of $\mathcal{M}$ are known. We remove this assumption and apply the RL approach to learn the desired policies in this work. We consider the following problem to account for challenges of both infeasible cases and unknown transitions.

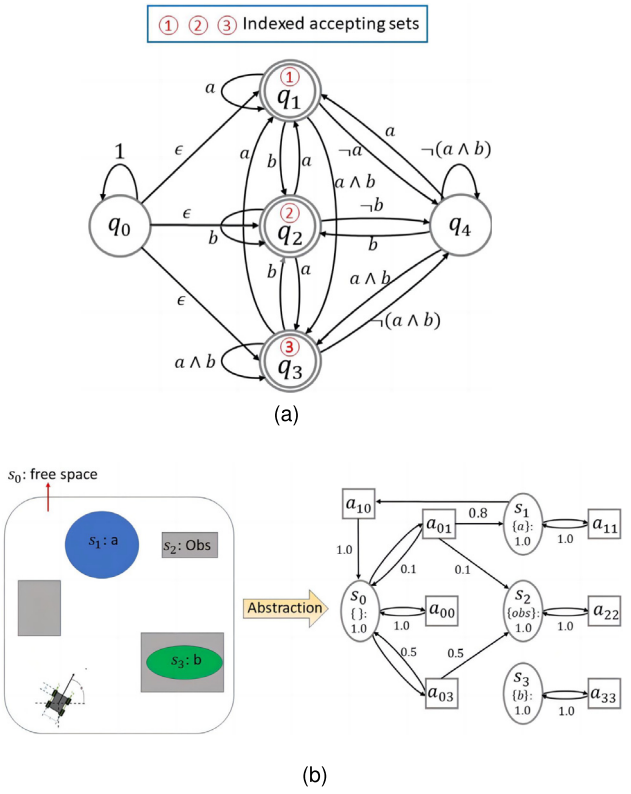


Fig. 1. A running example of an infeasible LTL task with respect to the PL-MDP. (a) The LDGBA of an LTL formula $\phi = \square\lozenge a \wedge \square\lozenge b$. (b) The abstraction process from a robotic system into a PL-MDP $\mathcal{M}$, where the region labeled with b is surrounded by obstacles and cannot be accessed. As a result, ϕ is infeasible in this case.

Problem 1. Given an LTL-specified task ϕ and a PL-MDP with unknown transition probabilities (representing motion uncertainties) and unknown probabilistic label functions (representing workspace uncertainties), the objective is to determine a policy that satisfies the following conditions: (1) if feasible, $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) > 0$; (2) if $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = 0$ for all policies $\boldsymbol{\mu}$, identify the policy $\boldsymbol{\mu}$ that maximally fulfills the task ϕ (i.e., under infeasible constraints) over an infinite horizon by minimizing $J_V(\mathcal{M}^{\boldsymbol{\mu}}, \phi)$.

4. Automaton analysis

To solve Problem 1, Section 4.1 first presents how the LDGBA in Definition 6 can be extended to E-LDGBA to keep tracking the non-visited accepting sets. Section 4.2 shows the traditional approach and the corresponding challenges for infeasible tasks. Section 4.3 presents the construction of a relaxed product MDP to handle soft LTL constraints. The benefits of incorporating E-LDGBA with relaxed product MDP are discussed in Section 4.4.

4.1. E-LDGBA

To identify the desired policy in PL-MDP $\mathcal{M}$ that satisfies the LTL formula ϕ specified by the user, one can construct the standard product MDP between $\mathcal{M}$ and the LDGBA of ϕ as outlined in [1]. This approach reformulates the problem as finding a policy that meets the acceptance condition of the resulting product MDP. However, using the LDGBA directly may lead to unsatisfied LTL specifications when applying deterministic policies, as discussed in our conference paper [36]. To address these limitations, we introduce the E-LDGBA as follows.

Given an LDGBA $\mathcal{A} = (Q, \Sigma, \delta \cup \{\epsilon\}, q_0, F)$, and following [41], a tracking-frontier set T is introduced to monitor unvisited accepting sets. Initially, T is set to F and is updated according to

$$f_V(q,T) = \begin{cases} T \setminus F_j, & \text{if } q \in F_j \text{ and } F_j \in T, \\ F \setminus F_j, & \text{if } q \in F_j \text{ and } T = \emptyset, \\ T, & \text{otherwise.} \end{cases} \tag{4}$$

Each time an accepting set F_j is visited, it is removed from T. When T is empty, it resets to $F \setminus F_j$. Since the LDGBA acceptance condition requires infinite visits to all accepting sets, we define a "round" as a complete visit to all sets (i.e., a round concludes when T becomes empty). If a state q belongs to multiple sets in T, all such sets are removed from T upon visiting q.

Definition 9 (*Embedded LDGBA*). Given an LDGBA $\mathcal{A} = (Q, \Sigma, \delta \cup \{\epsilon\}, q_0, F)$, the corresponding E-LDGBA is denoted by $\overline{\mathcal{A}} = (\overline{Q}, \Sigma, \overline{\delta} \cup \{\epsilon\}, \overline{q_0}, \overline{F}, f_V, T)$, where:

- T is initially set as $T = F$.
- $\overline{Q} = Q \times 2^F$ is the set of augmented states, e.g., $\overline{q} = (q, T)$. The finite alphabet Σ is the same as that in the LDGBA.
- The transition $\overline{\delta} : \overline{Q} \times (\Sigma \cup \{\epsilon\}) \to 2^{\overline{Q}}$ is defined by $\overline{q'} = \overline{\delta}(\overline{q}, \overline{\sigma})$ with $\overline{\sigma} \in (\Sigma \cup \{\epsilon\})$, where $\overline{q} = (q, T)$ and $\overline{q'} = (q', T)$, subject to two conditions: (1) $q' = \delta(q, \overline{\sigma})$, and (2) T is updated as $T = f_V(q', T)$ following the transition $\overline{q'} = \overline{\delta}(\overline{q}, \alpha)$.
- $\overline{F} = \left\{\overline{F_1}, \overline{F_2}, \ldots, \overline{F_f}\right\}$ is the set of accepting states, where $\overline{F_j} = \left\{(q, T) \in \overline{Q} \mid q \in F_j \wedge F_j \subseteq T\right\}$ for $j = 1, \ldots, f$.

In Definition 9, we adjust the tuple structure such that the frontier set T is synchronously updated after each transition. The state space is augmented with the tracking frontier set T, which can be efficiently represented using one-hot encoding based on the indices of the accepting sets. The accepting state is determined by both the current automaton state and the frontier set T. This feature is a key innovation of the E-LDGBA, ensuring that all accepting sets are visited in each round. Alg. 1 demonstrates the procedure of obtaining a valid run $\boldsymbol{q}_{\overline{\mathcal{A}}}$ over E-LDBGA $\overline{\mathcal{A}}_\phi$. Given an input alphabet α at each time step, lines 6–10 show how to generate the next state of $\overline{\mathcal{A}}_\phi$ and update the tracking frontier set T synchronously. Also, lines 6–8 indicate the difference and relationship between $\overline{\mathcal{A}}_\phi$ and its corresponding LDGBA $\mathcal{A}_\phi$.

Given $\overline{\mathcal{A}}_\phi$ and $\mathcal{A}_\phi$ for the same LTL formula, the E-LDGBA $\overline{\mathcal{A}}_\phi$ tracks unvisited accepting sets of $\mathcal{A}_\phi$ through the inclusion of f_V and T, allowing it to distinguish and enforce acceptance satisfaction related to different accepting sets. The set T is reset once all accepting sets of $\mathcal{A}_\phi$ have been visited. Let $\mathcal{L}(\mathcal{A}_\phi) \subseteq \Sigma^\omega$ and $\mathcal{L}(\overline{\mathcal{A}}_\phi) \subseteq \Sigma^\omega$ represent the languages accepted by $\mathcal{A}_\phi$ and $\overline{\mathcal{A}}_\phi$, respectively, over the same alphabet Σ. According to [1], $\mathcal{L}(\mathcal{A}_\phi) \subseteq \Sigma^\omega$ is the set of all infinite words accepted by $\mathcal{A}_\phi$ that satisfy the 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*

$$\mathcal{L}(\overline{\mathcal{A}}_\phi) = \mathcal{L}(\mathcal{A}_\phi). \tag{5}$$

Proof. Details of the proof can be found in our work [36] that has shown $\mathcal{L}(\overline{\mathcal{A}}_\phi) \supseteq \mathcal{L}(\mathcal{A}_\phi)$ and $\mathcal{L}(\overline{\mathcal{A}}_\phi) \subseteq \mathcal{L}(\mathcal{A}_\phi)$. This indicates that the E-LDGBA accepts the same language as the LDGBA and can be used to verify LTL satisfaction. □

Algorithm 1 Synthesizing a valid run of E-LDGBA

procedure INPUT: (LDGBA $\mathcal{A}$, f_V, T and length L)
  Output: A valid run $\overline{q}_{\overline{\mathcal{A}}}$ with length L in $\overline{\mathcal{A}}_\phi$
  set $T = F$ and $count = 1$
  set $\overline{q}_{cur} = (q_0, T)$ and $\overline{q}_{\overline{\mathcal{A}}} = (q_{cur})$
  set $q_{cur} = q$
  while $count \le L$ **do**
    given an input α, $q_{next} = \delta(q_{cur}, \alpha)$
    obtain $\overline{q}_{cur} \leftarrow (q_{next}, T)$
    check if $\overline{q}_{cur}$ is an accepting state
    add state $\overline{q}_{cur}$ to $\overline{q}_{\overline{\mathcal{A}}}$
    update $T = f_V(q_{next}, T)$
    $count++$ and $q_{cur} \leftarrow q_{next}$
  end while
end procedure

4.2. Product MDP and challenges

To find a policy $\boldsymbol{\mu}$ satisfying ϕ i.e., $\Pr^{\mu}_{\mathcal{M}}(\phi) > 0$, one can construct the standard product MDP as follows.

Definition 10. Given the labeled MDP $\mathcal{M}$ and the E-LDGBA $\overline{\mathcal{A}}_\phi$ corresponding to ϕ, the product MDP is defined as $\overline{\mathcal{P}} = (Y, U^{\overline{\mathcal{P}}}, p^{\overline{\mathcal{P}}}, x_0, F^{\overline{\mathcal{P}}})$, where

- $Y = S \times 2^{\Pi} \times \overline{Q}$ is the set of labeled states, i.e., $y = (s, l, \overline{q}) \in Y$ with $l \in L(s)$ satisfying $p_L(s, l) > 0$;
- $x_0 = (s_0, l_0, \overline{q}_0)$ is the initial state;
- $U^{\overline{\mathcal{P}}} = A \cup \{\epsilon\}$ is the set of actions, where the ϵ-actions are only allowed for transitions of E-LDGBA components from $\overline{Q}_N$ to $\overline{Q}_D$;
- $\overline{F}^{\overline{\mathcal{P}}} = \left\{ \overline{F}^{\overline{\mathcal{P}}}_1, \overline{F}^{\overline{\mathcal{P}}}_2 \dots \overline{F}^{\overline{\mathcal{P}}}_f \right\}$ is the set of accepting states. where $\overline{F}^{\overline{\mathcal{P}}}_j = \left\{ (s, l, \overline{q}) \in X \middle| \overline{q} \in \overline{F}_j \right\}$, $j = 1, \dots, f$;
- $p^{\overline{\mathcal{P}}} : Y \times U^{\overline{\mathcal{P}}} \times Y \to [0, 1]$ is transition probability defined as: (1) $p^{\overline{\mathcal{P}}}(y, u^{\overline{\mathcal{P}}}, y') = p_L(s', l') \cdot p_S(s, a, s')$ if $\delta(\overline{q}, l) = \overline{q}'$ and $u^{\overline{\mathcal{P}}} = a \in A(s)$; (2) $p^{\overline{\mathcal{P}}}(y, u^{\overline{\mathcal{P}}}, y') = 1$ if $u^{\overline{\mathcal{P}}} \in \{\epsilon\}$, $\overline{q}' \in \delta(\overline{q}, \epsilon)$, and $(s', l') = (s, l)$; and (3) $p^{\overline{\mathcal{P}}}(y, u^{\overline{\mathcal{P}}}, y') = 0$ otherwise.

The product MDP $\overline{\mathcal{P}}$ represents the intersections between all feasible paths in $\mathcal{M}$ and all words accepted by $\mathcal{A}_\phi$, thereby enabling the identification of admissible agent motions that fulfill the task ϕ. Let $\boldsymbol{\pi}$ denote a policy over $\overline{\mathcal{P}}$, and let $\boldsymbol{y}^{\boldsymbol{\pi}}_{\infty} = y_0 y_1 y_2 \dots$ represent the infinite path generated by $\boldsymbol{\pi}$. A path $\boldsymbol{y}^{\boldsymbol{\pi}}_{\infty}$ is accepted if $\inf(\boldsymbol{y}^{\boldsymbol{\pi}}_{\infty}) \cap \overline{F}^{\overline{\mathcal{P}}}_i \neq \emptyset$ for all $i \in \{1, \dots, f\}$. If $\boldsymbol{y}^{\boldsymbol{\pi}}_{\infty}$ is an accepting run, then there exists a policy $\boldsymbol{\xi}$ in $\mathcal{M}$ that satisfies ϕ.

Consider a sub-product MDP $\overline{\mathcal{P}}'_{(Y', U')}$, where $Y' \subseteq X$ and $U' \subseteq U^{\overline{\mathcal{P}}}$. If $\overline{\mathcal{P}}'_{(Y', U')}$ is a MEC of $\overline{\mathcal{P}}$ and $Y' \cap \overline{F}^{\overline{\mathcal{P}}}_i \neq \emptyset$, $\forall i \in \{1, \dots, f\}$, then $\overline{\mathcal{P}}'_{(Y', U')}$ is called an accepting maximum end component (AMEC) of $\overline{\mathcal{P}}$. Once a path enters an AMEC, The subsequent path remains within the component by taking restricted actions from U'. There exist policies allowing any state $x \in X'$ to be visited infinitely often. Consequently, satisfying the task ϕ is equivalent to reaching an AMEC. Furthermore, a MEC that lacks any accepting sets is termed a rejecting maximum end component (RMEC), while a MEC containing only some accepting sets is referred to as a neutral maximum end component (NMEC).

4.3. Relaxed product MDP

For the product MDP $\overline{\mathcal{P}}$ or $\overline{\mathcal{P}}$ introduced above, the satisfaction of ϕ is based on the assumption that at least one AMEC exists in the product MDP, i.e., at least one policy satisfies the given LTL formula with respect to the PL-MDP. Otherwise, the task is infeasible with respect to the PL-MDP. We treat the LTL task as soft constraints to address the cases where tasks can be potentially infeasible. The relaxed product MDP is designed to allow the agent to revise its motion plan without strictly following the desired LTL constraints.

Definition 11. The relaxed product MDP is constructed from $\overline{\mathcal{P}}$ as a tuple $\overline{\mathcal{R}} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi = (X, U^{\overline{\mathcal{R}}}, p^{\overline{\mathcal{R}}}, x_0, F^{\overline{\mathcal{R}}}, c^{\overline{\mathcal{R}}}_V, f_V, T)$, where

- X, x_0, and $F^{\overline{\mathcal{R}}} = F^{\overline{\mathcal{P}}}$ are the same as in $\overline{\mathcal{P}}$;
- $U^{\overline{\mathcal{R}}}$ is the set of extended actions that jointly consider the actions of $\mathcal{M}$ and the input atomic proposition of $\mathcal{A}_\phi$. Specifically, given a state $x = (s, l, q) \in X$, the available actions are $u^{\overline{\mathcal{R}}}(x) = \left\{ (a, \iota) \middle| a \in A(s), \iota \in (2^{AP} \cup \{\epsilon\}) \right\}$. Given an action $u^{\overline{\mathcal{R}}} = (a, \iota) \in u^{\overline{\mathcal{R}}}(x)$, the projections of $u^{\overline{\mathcal{R}}}$ to $A(s)$ in $\mathcal{M}$ and to $2^{AP} \cup \{\epsilon\}$ in $\mathcal{A}_\phi$ are denoted by $u\big|^{\overline{\mathcal{R}}}_{\mathcal{M}}$ and $u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}}$, respectively;
- $p^{\overline{\mathcal{R}}} : X \times u^{\overline{\mathcal{R}}} \times X \to [0, 1]$ is the transition function. The transition probability $p^{\overline{\mathcal{R}}}$ from a state $x = (s, l, \overline{q})$ to a state $x' = (s', l', \overline{q}')$ is defined as: (1) $p^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x') = p_L(s', l') \cdot p_S(s, a, s')$ with $a = u\big|^{\overline{\mathcal{R}}}_{\mathcal{M}}$, if $\overline{q}$ can be transitioned to $\overline{q}'$ and $\overline{q}' = \overline{\delta}(\overline{q}, u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}})$ s.t $u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}} \neq \epsilon$; (2) $p^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x') = 1$, if $u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}} = \epsilon$, $\overline{q}' \in \overline{\delta}(\overline{q}, \epsilon)$, and $(s', l') = (s, l)$; (3) $p^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x') = 0$ otherwise. Under an action $u^{\overline{\mathcal{R}}} \in U^{\overline{\mathcal{R}}}(x)$, it holds that $\sum_{x' \in X} p^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x') = 1$;
- $c^{\overline{\mathcal{R}}}_V : X \times u^{\overline{\mathcal{R}}} \times X \to \mathbb{R}$ is the violation cost. To define the violation cost $c^{\overline{\mathcal{R}}}_V$, consider $\Pi = \{\iota_1, \iota_2 \dots \iota_M\}$ and an evaluation function $\mathrm{Eval} : 2^{\Pi} \to \{0, 1\}^M$, defined as $\mathrm{Eval}(l) = [v_i]^M$ with $v_i = 1$ if $\iota_i \in l$ and $v_i = 0$ otherwise, where $i = 1, 2, \dots, M$ and $l \in 2^{\Pi}$. To quantify the difference between two elements in 2^{Π}, we introduce $\rho(l, l') = \|v - v'\|_1 = \sum_{i=1}^{M} |v_i - v'_i|$, where $v = \mathrm{Eval}(l)$, $v' = \mathrm{Eval}(l')$, $l, l' \in 2^{\Pi}$, and $\|\cdot\|_1$ is the ℓ_1 norm. The distance from $l \in 2^{\Pi}$ to a set $\mathcal{X} \subseteq 2^{\Pi}$ is then defined as $\mathrm{Dist}(l, \mathcal{X}) = 0$ if $l \in \mathcal{X}$, and $\mathrm{Dist}(l, \mathcal{X}) = \min_{l' \in \mathcal{X}} \rho(l, l')$ if $l \notin \mathcal{X}$. The violation cost of the transition from $x = (s, l, q)$ to $x' = (s', l', q')$ under an action $u^{\overline{\mathcal{R}}}$ is defined as

$$c^{\overline{\mathcal{R}}}_V(x, u^{\overline{\mathcal{R}}}, x') = \begin{cases} p_L(s', l') \cdot w_V(x, x') & \text{if } u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}} \neq \epsilon, \\ 0 & \text{otherwise,} \end{cases}$$

  where $w_V(x, x') = \mathrm{Dist}(L(s), \mathcal{X}(q, q'))$ with $\mathcal{X}(q, q') = \left\{ l \in 2^{\boldsymbol{\pi}} \middle| q \xrightarrow{l} q' \right\}$ being the set of input alphabets that enables the transition from q to q'. Borrowed from [35], the function $\mathrm{Dist}(L(s), \mathcal{X}(q, q'))$ measures the distance from $L(s)$ to the set $\mathcal{X}(q, q')$.

Alg. 2 illustrates executing a given policy and generating a valid run over the relaxed product MDP $\overline{\mathcal{R}}$ on-the-fly. "On-the-fly" means the procedure does not need to construct $\overline{\mathcal{R}}$ completely. In contrast, it tracks the states based on the evolution in Definition 11. In line 8 of Alg. 2, the next automaton state of $\overline{\mathcal{A}}_\phi$ is reached according to lines 6 – 7 of Alg. 1 by assigning the input alphabet as $u\big|^{\overline{\mathcal{R}}}_{\mathcal{A}}$. In lines 5–9, the next state is jointly determined by the transitions of PL-MDP and E-LDGBA (Alg. 1). Line 14 of Alg. 2 also updates the tracking frontier set synchronously as Alg. 1, and line 11 computes its violation cost after generating the valid transition.

The weighted violation function $w_V(x, x')$ quantifies how much the transition from x to x' in a product MDP violates the constraints imposed by ϕ. The transition probability $p^{\overline{\mathcal{R}}}$ jointly considers the probability of an event $p_L(s', l')$ and the transition p_S. Then the measurement of violation in (3) can be reformulated over relaxed product MDP as the expected return of violation cost:

Definition 12. Given a relaxed product MDP $\overline{\mathcal{R}}$ generated from a PL-MDP $\mathcal{M}$ and an E-LDBA $\overline{\mathcal{A}}_\phi$, the measurement of expected violation in (3) can be reformulated over relaxed product MDP under policy $\boldsymbol{\pi}$

Algorithm 2 Policy Execution over $\overline{\mathcal{R}} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi$

procedure INPUT: (policy π, $\mathcal{M}$, $\overline{\mathcal{A}}_\phi$, N)
Output: a valid run $\boldsymbol{x}_N$ with length of N under policy π over $\overline{\mathcal{R}}$
set $t = 0$ and $x_t = x_0$
set $\boldsymbol{x}_N = (x_t)$
while $t < N$ **do**
obtain $u_t^{\overline{\mathcal{R}}}$ based on $\pi(x_t)$
execute $u_t\big|_{\mathcal{M}}^{\overline{\mathcal{R}}}$ for $\mathcal{M}$ and obtain s_{t+1}
observe label l_{t+1} according to $L(s_t)$
obtain $q_{t+1} = \delta(q_t, u\big|_{\overline{\mathcal{A}}}^{\overline{\mathcal{R}}})$ s.t. $\overline{q}_{t+1} = (q_{t+1}, T)$
obtain $x_{t+1} = (s_{t+1}, l_{t+1} q_{t+1}, T)$
check if the x_{t+1} is an accepting state
compute $c_V^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x')$ based on Definition 11
add x_{t+1} to $\boldsymbol{x}_N$
$x_t = x_{t+1}$ and $t{+}{+}$
update $T \leftarrow f_V(q', T)$ for $\overline{\mathcal{A}}_\phi$
end while
end procedure

as:

$$J_V(\overline{\mathcal{R}}^{\boldsymbol{\pi}}) = \mathbb{E}_{\overline{\mathcal{R}}}^{\boldsymbol{\pi}}\left[\sum_{i=0}^{\infty} \gamma^i c_V^{\overline{\mathcal{R}}}(x_i, u_i^{\overline{\mathcal{R}}}, x_{i+1}).\right] \tag{6}$$

Finding a policy to minimize $J_V(\overline{\mathcal{R}}^{\boldsymbol{\pi}})$ will enforce the planned path towards more fulfilling the LTL task ϕ by penalizing w_V. A run $r_{\overline{\mathcal{P}}}^{\pi}$ induced by a policy π that satisfies the accepting condition of the relaxed product MDP $\overline{\mathcal{R}}$ completes the corresponding LTL mission ϕ exactly if and only if the violation return (6) is equal to zero.

4.4. Properties of relaxed product MDP

This section discusses the properties of the relaxed product MDP and standard product MDP. Applying the same process in Definition 10, we can construct the product MDP between LDGBA and PL-MDP denoted as $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}_\phi$. The detailed procedure can be found in [1]. The relaxed product MDP between MDP and LDGBA can also be constructed as the procedure Definition 11 by replacing E-LDGBA with LDGBA denoted as $\mathcal{R} = \mathcal{M} \otimes \mathcal{A}_\phi$. Note, the product and relaxed product are the procedures of constructing the interaction between the automaton structure and the MDP model, which can be applied to any automata.

Similarly, $\mathcal{R}$ and $\mathcal{P}$ for the same $\mathcal{A}_\phi$ and PL-MDP $\mathcal{M}$ share identical states. Thus, $\mathcal{R}$ and $\mathcal{P}$ can be viewed as two distinct directed graphs with the same set of nodes. From a graph-theoretic perspective, a MEC can be considered a bottom strongly connected component (BSCC). Similarly, let ABSCC denote the BSCC that intersects with all accepting sets in $\mathcal{R}$ or $\mathcal{P}$. We then arrive at the following conclusion.

Theorem 1. *Given an MDP $\mathcal{M}$ and an LDGBA $\mathcal{A}_\phi$ for the LTL formula ϕ, let $\mathcal{R} = \mathcal{M} \otimes \mathcal{A}_\phi$ denote the relaxed product MDP and $\mathcal{P}$ the corresponding standard product MDP. The following properties hold:*

1. *The directed graph of the standard product MDP $\mathcal{P}$ is a subgraph of the directed graph of $\mathcal{R}$.*
2. *There always exists at least one AMEC in $\mathcal{R}$.*
3. *If the LTL formula ϕ is feasible over $\mathcal{M}$, then any directed graph of an AMEC in $\mathcal{P}$ is a subgraph of a directed graph of an AMEC in $\mathcal{R}$.*

Proof. The proof is similar as our previous work [18] by replacing the LDBA with LDGBA. □

Since the E-LDGBA is an extension of LDGBA to enable the recording ability regarding accepting sets, which accept the same LTL formulas as in Lemma 1. We can also have the same conclusion for $\overline{\mathcal{R}}$ and $\overline{\mathcal{P}}$.

Lemma 2. *Given an MDP $\mathcal{M}$ and an E-LDGBA $\overline{\mathcal{A}}_\phi$ of ϕ, the relaxed product MDP $\overline{\mathcal{R}} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi$ and its corresponding $\overline{\mathcal{P}}$ have the following properties*

1. *the directed graph of standard product $\overline{\mathcal{P}}$ is a sub-graph of the directed graph of $\overline{\mathcal{R}}$,*
2. *there always exists at least one AMEC in $\overline{\mathcal{R}}$,*
3. *if the LTL formula ϕ is feasible over $\mathcal{M}$, any AMEC of $\overline{\mathcal{P}}$ is the sub-graph of an AEMC of $\overline{\mathcal{R}}$.*

Lemma 2 can be verified by employing the proof of Theorem 1. Both of them demonstrate the advantages of applying the relaxed product MDP, which allows us to handle infeasible situations.

Example 2. Fig. 2(a) and (b) provide examples of a standard product MDP and the corresponding relaxed one, respectively, to illustrate the benefits of designing the relaxed product MDP. Both product MDPs are constructed between the PL-MDP and the LDGBA of LTL formula $\phi = \square\Diamond \mathtt{a} \wedge \square\Diamond \mathtt{b}$, as shown in Fig. 1. Obviously, the product MDP in Fig. 2(a) has no AMEC, while the relaxed product MDP in Fig. 2(b) has one AMEC (blue rectangular) intersecting with all accepting sets. Moreover, the transitions with nonzero violation costs are gray dashed lines. Note that due to the complicated graph structure, we use the LDGBA in this example to illustrate the novelty of the proposed relaxed product MDP. Its advantages for infeasible cases are also applicable to the E-LDGBA.

Given a relaxed product MDP $\overline{\mathcal{R}}$, let $MC_{\overline{\mathcal{R}}}^{\pi}$ represent the Markov chain induced by policy π on $\overline{\mathcal{R}}$. The states of this Markov chain can be expressed as a disjoint union of a transient class T_π and n closed irreducible recurrent sets R_π^j for $j \in \{1, \ldots, n\}$, i.e., $MC_{\overline{\mathcal{R}}}^{\pi} = T_\pi \sqcup R_\pi^1 \sqcup R_\pi^2 \sqcup \ldots \sqcup R_\pi^n$ [48].

Lemma 3. *Given a relaxed product MDP $\overline{\mathcal{R}} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi$, the recurrent class R_π^j of $MC_{\overline{\mathcal{R}}}^{\pi}$, $\forall j \in \{1, \ldots, n\}$, induced by π satisfies one of the following conditions:*

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

Proof. The following proof is based on contradiction. Assume there exists a policy such that $R_\pi^j \cap F_k^{\overline{\mathcal{R}}} \neq \emptyset$, $\forall k \in K$, where K is a subset of $2^{\{1,\ldots,f\}} \setminus \big\{\{1, \ldots, f\}, \emptyset\big\}$. As discussed in [48], for each state in recurrent class, it holds that $\sum_{n=0}^{\infty} p^n(x, x) = \infty$, where $x \in R_\pi^j \cap F_k^{\overline{\mathcal{R}}}$ 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 Definition 9, the tracking set T will not be reset until all accepting sets have been visited. As a result, neither $\overline{q}_k \in \overline{F}_k$ nor $x_k = (s, \overline{q}_k) \in R_\pi^j \cap F_k^{\overline{\mathcal{R}}}$ with $s \in S$ will occur infinitely, which contradicts the property $\sum_{n=0}^{\infty} p^n(x_k, x_k) = \infty$. □

Lemma 3 indicates that, for any policy, all accepting sets will be placed either in the transient class or the recurrent class. As a result, the issue of NMEC, as in many existing methods, can be avoided. Based on Theorem 1 and Lemma 3, Problem 1 can be reformulated as follows.

Problem 2. Given a user-specified LTL task ϕ and an MDP with unknown transition probabilities (representing motion uncertainties) and unknown labeling probabilities (representing environment uncertainties), the objective is to derive a policy that prioritizes, in decreasing order: (1) satisfying the acceptance condition of the relaxed product MDP, and (2) minimizing the violation cost within the expected return.

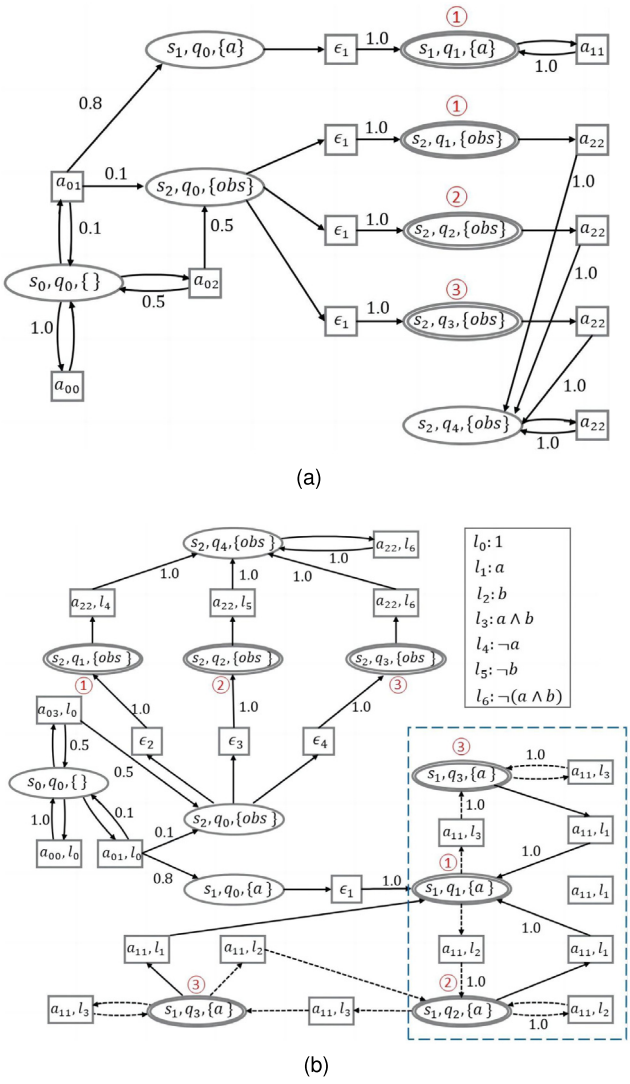


Fig. 2. Two different product MDPs between the PL-MDP and the LDGBA provided in Fig. 1. (a) Standard product MDP. There are no criteria to evaluate a policy for task satisfaction, and we cannot find a path satisfying the acceptance condition. (b) Partial structure of the relaxed product MDP. $l_i, \forall i \in \{0, 1, \ldots, 7\}$ denotes the input alphabet of each automaton transition as shown in the gray rectangular. The gray dashed lines represent the transitions with nonzero violation costs, and the graph component in blue rectangular contains the path that satisfies the acceptance condition.

5. Learning-based control synthesis

In this section, RL is leveraged to identify policies for Problem 2. Specifically, a model-free multi-objective RL (MORL) is designed.

5.1. Reward design

Referring to [41], the accepting reward function $\Lambda : X \to \mathbb{R}$ is designed as

$$\Lambda(x) = \begin{cases} r_\Lambda & \text{if } \exists i \in \{1, \ldots, f\} \text{ such that } x \in F_i^{\overline{\mathcal{R}}}, \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $r_\Lambda > 0$. The violation function $V : X \times U^{\overline{\mathcal{R}}} \times X \to \mathbb{R}$ is designed as

$$V(x, u^{\overline{\mathcal{R}}}, x') = -c_V^{\overline{\mathcal{R}}}(x, u^{\overline{\mathcal{R}}}, x'). \tag{8}$$

The non-negative $\Lambda(x)$ enforces the accepting condition of $\overline{\mathcal{R}}$, while the non-positive function $V(x, u^{\overline{\mathcal{R}}}, x')$ indicates the penalty of violations. Let $\boldsymbol{U}_{\boldsymbol{\pi}} = \begin{bmatrix} U_{\boldsymbol{\pi}}(x_0) & U_{\boldsymbol{\pi}}(x_1) & \ldots \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N$ denote the stacked expected return induced by $\boldsymbol{\pi}$ over $\overline{\mathcal{R}}$ with $N = |X|$, the expected return is designed as

$$\boldsymbol{U} = \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\boldsymbol{\pi}}^n (\boldsymbol{\Lambda}_{\boldsymbol{\pi}} + \beta \boldsymbol{V}_{\boldsymbol{\pi}} \cdot \mathbf{1}_N), \tag{9}$$

where $0 < \gamma < 1$ is a discount factor, $\boldsymbol{P}_{\boldsymbol{\pi}} \in \mathbb{R}^{N \times N}$ is a matrix with entries representing the probabilities $p^{\overline{\mathcal{R}}}(x, \boldsymbol{\pi}(x), x')$ under π for all $x, x' \in X$, $\boldsymbol{\Lambda}_{\boldsymbol{\pi}} = \begin{bmatrix} \Lambda(x_0) & \Lambda(x_1) & \ldots \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N$ is the stacked state rewards, $\beta \in \mathbb{R}^+$ is a weight indicating the relative importance, $\mathbf{1}_N$ is an N-dimensional vector of ones, and $\boldsymbol{V}_{\boldsymbol{\pi}} = \boldsymbol{P}_{\boldsymbol{\pi}} \circ \boldsymbol{V} \in \mathbb{R}^{N \times N}$ is the Hadamard product of $\boldsymbol{P}_{\boldsymbol{\pi}}$ and $\boldsymbol{V}$, i.e., $\boldsymbol{V}_{\boldsymbol{\pi}} = \left[p^{\overline{\mathcal{R}}}(x, a_{\boldsymbol{\pi}}(x), x') \cdot V(x, u^{\overline{\mathcal{R}}}, x') \right]_{N \times N}$ and $a_{\boldsymbol{\pi}}(x)$ represents taking action at x from policy $\boldsymbol{\pi}$.

The objective is to identify a stationary policy π^* that maximizes the expected return

$$\pi^* = \arg\max_{\pi} \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\boldsymbol{\pi}}^n (\boldsymbol{\Lambda}_{\boldsymbol{\pi}} + \beta \boldsymbol{V}_{\boldsymbol{\pi}} \cdot \mathbf{1}_N) \tag{10}$$

$U_{\boldsymbol{\pi}}(x) \le U_{\boldsymbol{\pi}^*}(x)$ for all $x \in X$ if π^* in (10) is optimal.

Theorem 2. *Consider a relaxed MDP product $\overline{\mathcal{R}} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi$. If there exists a policy $\bar{\boldsymbol{\pi}}$ such that an induced run $r_{\bar{\mathcal{P}}}^{\bar{\boldsymbol{\pi}}}$ satisfies the acceptance condition of $\overline{\mathcal{R}}$, any optimization method that solves (10) can find the policy $\bar{\boldsymbol{\pi}}$.*

Proof. Detailed rigorous analysis can be found in Appendix A.1 □

To solve Problem 2 by optimizing the expected return of (9), r_Λ, β, γ can be determined as follows. Firstly, we can choose a fixed r_Λ. Then, γ can be obtained by solving (21) and (22). Finally, the range of β can be determined by solving (18) and (22). In order to minimize the violation cost, a great value of β is preferred.

Remark 2. Problem 2 is a MORL problem, and Theorem 2 shows how it can be addressed to ensure acceptance satisfaction while trying to minimize the long-term violation. In literature, MORL based approaches often seek to identify Pareto fronts [49]. However, our problem considers a trade-off between two possibly conflicting objectives. There are no existing MORL methods to provide formal guarantee of both acceptance satisfaction and violation depreciation, which might result in a safety issue during the execution of optimal policies. To overcome this challenge, we can always divide the task into two parts: hard and soft constraints, with two parallel automatons. The hard task should always be satisfied, and it can be applied with the traditional product MDP as Definition 10. The soft part can be partially infeasible, and it can be relaxed as Definition 11. Such an idea can be found in our previous results [17].

It is worth pointing out that $\boldsymbol{V}_{\pi}$ is a sparse matrix because most transitions have zero violation cost, and Theorem 2 provides a performance guarantee for the worst cases of β for acceptance satisfaction. By applying E-LDGBA, the sparse reward issue can be improved compared with using LDGBA.

5.2. Model-free reinforcement learning

Q-learning is a model-free RL method [20], which can be used to find the optimal policy for a finite MDP. In particular, the agent updates its Q-value from x to x' according to

$$\begin{aligned} Q(x, u^{\overline{\mathcal{R}}}) \leftarrow & (1 - \alpha) Q(x, u^{\overline{\mathcal{R}}}) \\ & + \alpha \left[R(x, u^{\overline{\mathcal{R}}}, x') + \gamma \max_{u_*^{\overline{\mathcal{R}}} \in U^{\overline{\mathcal{R}}}(x)} Q(x, u_*^{\overline{\mathcal{R}}}) \right], \end{aligned} \tag{11}$$

where $Q(x, u^{\overline{\mathcal{R}}})$ is the Q-value of the state–action pair $(x, u^{\overline{\mathcal{R}}})$, $\alpha \in (0, 1)$ is the learning rate, $0 < \gamma \le 1$ is the discount factor, and

$$R(x, u^{\overline{\mathcal{R}}}, x') = \Lambda(x) + \beta V(x, u^{\overline{\mathcal{R}}}, x') \tag{12}$$

Algorithm 3 Model-free RL-based control on MDPs under potentially soft LTL specifications.

procedure INPUT: $(\mathcal{M}, \phi, \Lambda)$
  Output: optimal policy $\boldsymbol{\pi}^*$
  Initialization: Set $episode = 0$, $iteration = 0$ and τ as maximum allowed learning steps
  Initialize r_Λ, β, γ;
  for all $x \in X$ **do**
    $U(x) = 0$ and $Q(x, u^{\overline{R}}) = 0\ \forall x,\ u^{\overline{R}} \in U^{\overline{R}}(x)$
    $Count(x, u^{\overline{R}}) = 0$ for all $u^{\overline{R}} \in U^{\overline{R}}(x)$
  end for
  $x = x_0$;
  while U are not converged **do**
    $episode++$;
    $\epsilon = 1/episode$;
    while $iteration < \tau$ **do**
      $iteration++$
      $u = \arg\max_{u^{\overline{R}} \in U^{\overline{R}}} Q(x, u^{\overline{R}})$ with probability $1-\epsilon$, or select u randomly from $U^{\overline{R}}(x)$
      Obtain and execute a of $\mathcal{M}$ from u
      Observe x', $\Lambda(x), c_V^{\overline{R}}(x, u, x')$
      $r \leftarrow \Lambda(x) - \beta c_V^{\overline{R}}(x, u, x')$
      $Count(x, u)++$
      $\alpha = 1/Count(x, u)$
      $Q(x,u) \leftarrow (1-\alpha)Q(x,u) + \alpha\left[r + \gamma \cdot \max_{u^{\overline{R}} \in U^{\overline{R}}} Q(x', u^{\overline{R}})\right]$
      $x = x'$
    end while
  end while
  for all $x \in X$ **do**
    $U(x) = \max_{u^{\overline{R}} \in U^{\overline{R}}} Q(x, u^{\overline{R}})$
    $\boldsymbol{\pi}^*(x) = \arg\max_{u^{\overline{R}} \in U^{\overline{R}}} U(x)$
  end for
end procedure

denotes the immediate reward from x to x' under $u^{\overline{R}}$. The learning strategy is outlined in Alg. 3 that shows the steps of applying Q-learning into our framework. By applying the off-the-shelf RL, we have the following theorem.

Theorem 3. *Given a finite MDP, i.e., $\overline{R} = \mathcal{M} \otimes \overline{\mathcal{A}}_\phi = (X, U^{\overline{R}}, p^{\overline{R}}, x_0, F^{\overline{R}}, c_V^{\overline{R}}, f_V, T)$, let $Q^*(x, u^{\overline{R}})$ be the optimal Q-function for every pair of state–action $(x, u^{\overline{R}})$. Consider the RL with the updating rule*

$$Q_{k+1}(x, u^{\overline{R}}) \leftarrow (1-\alpha_k) Q_k(x, u^{\overline{R}}) + \alpha_k \left[R(x, u^{\overline{R}}, x') + \gamma \max_{u_*^{\overline{R}} \in U^{\overline{R}}(x)} Q_k(x, u_*^{\overline{R}}) \right],$$

where k is the updating step, $\gamma \in (0, 1]$ is the discount factor, and the learning rate α_k satisfies $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$. Then, $Q_k(x, u^{\overline{R}})$ converges to $Q^*(x, u^{\overline{R}})$ with a probability of 1 as $k \to \infty$.

Theorem 3 is an immediate result of [50]. With standard learning rate α and discount factor γ as Alg. 3, Q-value will converge to a unique limit Q^*. Therefore, the optimal expected utility and policy can be obtained as $U_{\boldsymbol{\pi}}^*(x) = \max_{u^{\overline{R}} \in U^{\overline{R}}(x)} Q^*(x, u^{\overline{R}})$ and $\boldsymbol{\pi}^*(x) = \arg\max_{u^{\overline{R}} \in U^{\overline{R}}(x)} Q^*(x, u^{\overline{R}})$. In (11), the discount γ is tuned to improve the trade-off between immediate and future rewards. Note that our novel design can be also applied with modern advanced RL algorithms.

Complexity Analysis The number of states is $|S| \times |Q|$, 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 labeled MDP model. Due to the consideration of a relaxed product MDP and extended actions in Definition 11, the maximum complexity of actions available at $x = (s, l, q)$ is $O(|A(s)| \times (|\Sigma| + 1))$, since $U^{\overline{R}}(x)$ are created from $A(s)$ and $\Sigma \cup \{\epsilon\}$.

6. Case studies

The developed RL-based control synthesis is implemented in Python. Owl [47] is used to convert LTL specifications into LDGBA, and P_MDP package [13] is used to construct state transition models. 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 learning rate α is determined by Alg. 3 with $\gamma = 0.999$ and $r_\Lambda = 10$. 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 a TIAGo robot.

6.1. Simulation results

Consider a partitioned workspace of 10 m × 10 m, as illustrated in Figs. 3 and 5, where each cell represents an area of 2 m × 2 m. Cells are color-coded to designate specific areas of interest, including Base1, Base2, Base3, Obs, and Sply, where Obs and Sply stand for obstacle and supply zones, respectively. To incorporate environmental uncertainties, each cell is assigned a probability indicating the likelihood of a particular property appearing there. For instance, Obs : 0.1 denotes a 0.1 probability that the cell contains an obstacle. The robot operates based on a unicycle model, with dynamics given by $\dot{x} = v \cdot \sin(\theta)$, $\dot{y} = v \cdot \cos(\theta)$, and $\dot{\theta} = \omega$, where x, y, and θ represent the robot's position and orientation. The control inputs are the linear and angular velocities, denoted as $u = (v, \omega)$.

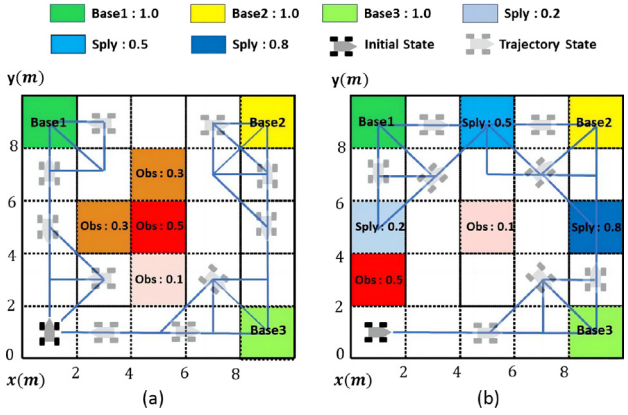


Fig. 3. Simulated trajectories of 50 time steps under corresponding optimal policies in feasible workspaces with (a) a simple task, i.e., φ_{case1}, and (b) a relatively complex task, i.e., φ_{case2}.

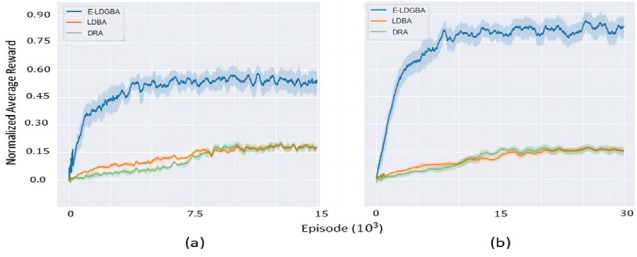


Fig. 4. Normalized average rewards during training using E-LDGBA, LDBA, and DRA for (a) Task φ_{case1} in Fig. 3 and (b) Task φ_{case2} in Fig. 3.

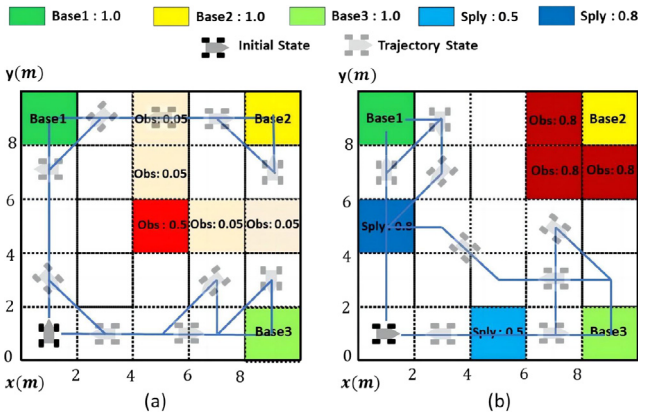


Fig. 5. Simulated trajectories of 50 time steps under corresponding optimal policies for partially infeasible tasks on workspaces where the agent visits the grid labeled as Base2 with (a) a low risk or (b) a high risk.

In addition, we assume the robot cannot consistently successfully execute the action primitives to model motion uncertainties. For instance, action primitives "FR" and "BK" mean the robot can successfully move forward and backward 2 m with a probability of 0.9, respectively, and fail with a probability of 0.1. On the other hand, action primitives "TR" and "TL" mean the robot can successfully turn right and left for an angle of $\frac{\pi}{2}$ exactly with a probability of 0.9, respectively, and fail by an angle of $\frac{\pi}{4}$ (undershoot) with a probability of 0.05 and an angle of $\frac{3\pi}{4}$ (overshoot) with another probability of 0.05. Finally, action primitive "ST" means the robot remains at its current cell. The resulting MDP model has 25 states.

Case 1: As shown in Fig. 3(a), we first consider a case in which user-specified tasks can be successfully executed. The desired surveillance task to be performed is formulated as

$$\varphi_{case1} = (\square\lozenge \text{Base1}) \wedge (\square\lozenge \text{Base2}) \wedge (\square\lozenge \text{Base3}) \wedge \square\neg \text{Obs}, \tag{13}$$

which requires the mobile robot to visit all base stations infinitely while avoiding obstacles. Its corresponding LDGBA has 2 states with 3 accepting sets, and the relaxed product MDP has 50 states. In this case, each episode terminates after $\tau = 100$ steps and $\beta = 8$. Fig. 3(a) shows the generated optimal trajectory, which indicates φ_{case1} is completed.

Case 2: We validate our approach with more complex task specifications in Fig. 3(b). The task is expressed as

$$\varphi_{case2} = \varphi_{case1} \wedge \square\lozenge(\text{Sply} \rightarrow \bigcirc((\neg \text{Sply}) \mathcal{U} \varphi_{\text{one1}})), \tag{14}$$

where $\varphi_{one1} = \text{Base1} \vee \text{Base2} \vee \text{Base3}$. φ_{case2} requires the robot to visit the supply station and then go to one of the base stations while avoiding obstacles. It also requires all base stations to be visited. Its corresponding LDGBA has 24 states with 4 accepting sets, and the relaxed product MDP has 600 states. The generated optimal trajectory is shown in Fig. 3(b).

Case 3: We consider more challenging workspaces in Fig. 5 (a) and (b), where user-specified tasks might not be fully feasible due to potential obstacles (i.e., environment uncertainties). The task specification is

$$\varphi_{case3} = \varphi_{case1} \wedge \square\lozenge(\varphi_{one1} \rightarrow \bigcirc((\neg \varphi_{one1}) \mathcal{U} \text{Sply})), \tag{15}$$

where φ_{case3} requires the robot to visit one of the base stations and then go to one of the supply stations while avoiding obstacles. Its corresponding LDGBA has 24 states with 4 accepting sets, and the product-MDP has 600 states.

For φ_{case2} and φ_{case3}, each episode terminates after $\tau = 1000$ steps and $\beta = 5$. Note that in the case of Fig. 5 (a) and (b), AMECs might not exist since Base2 is surrounded by probabilistic obstacles and may not be visited. The simulated optimal trajectories indicate that the robot takes a risk to accomplish the task in the case of Fig. 5(a). In contrast, the robot decides not to visit Base2 to avoid the high risk of running into obstacles in the case of Fig. 5(b).

Table 1
Simulation results of large scale workspaces.

Workspace size[cell]	MDP States	$\overline{\mathcal{R}}$ States	Episode Steps
15 × 15	225	450	30000
25 × 25	625	1250	50000
40 × 40	1600	3200	100000

Scalability Analysis: RL-based policy synthesis is conducted for φ_{case1} across workspaces of varying sizes (with each grid further subdivided) to illustrate computational complexity. The simulation results are presented in Table 1. The number of episodes in Table 1 indicates the time used to converge to optimal satisfaction planing. It is also verified that the given task φ_{case1} can be successfully carried out in large workspaces.

6.2. Comparison

We compare our algorithm with the existing learning-based works [26,30,31,38,39,41]. From the automaton perspective, DRA has been leveraged in [26,38,39] to follow the Rabin acceptance condition with rejecting sets and accepting sets. In addition, the Limit-Deterministic Büchi Automaton (LDBA) used in [30,31] only has one accepting set. Figs. 7(a) and 1(b) illustrate that (E-)LDGBA and LDBA have different numbers of accepting sets for the same LTL formula $\phi = \square\lozenge a \wedge \square\lozenge b$. On the other hand, the reward is typically designed based on the acceptance condition after integrating LTL with the RL algorithm. Consequently, applying the LDBA and DRA results in more sparse rewards than E-LDGBA (with several accepting sets). Figs. 4 and 6 show the convergence and comparison of learning results.

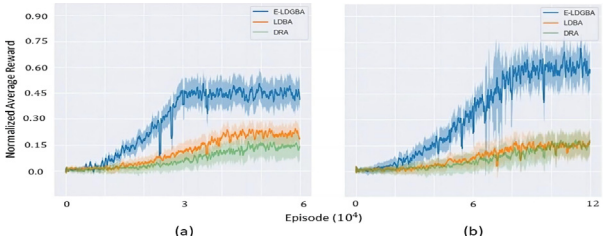


Fig. 6. Normalized average rewards during training using E-LDGBA, LDBA, and DRA for Task φ_{case3} on two environments in Fig. 5, respectively.

Regarding infeasible LTL tasks with respect to environment uncertainties, the most recent work [41] designed the reward to maximize the satisfaction of infeasible tasks by visiting accepting sets in a standard product MDP as much as possible. However, for some specific LTL formulas, e.g., $\phi = \lozenge(a \wedge \lozenge b)$, the corresponding LDGBA only has one accepting set, as shown in Fig. 7(b). After taking the product of this LDGBA and the PL-MDP in Fig. 1(a), it can be observed in Fig. 7(c) that the standard product MDP has no accepting set. Consequently, the work [41] cannot handle this infeasible case. In contrast, the relaxed product MDP in Fig. 7(d) from our work allows finding the policy with the least expected violation costs.

6.3. Experimental results

This section validates our algorithm for high-level decision-making tasks within a real-world environment, demonstrating that the framework can integrate with low-level noisy controllers to establish a hierarchical architecture. Consider an office environment constructed in

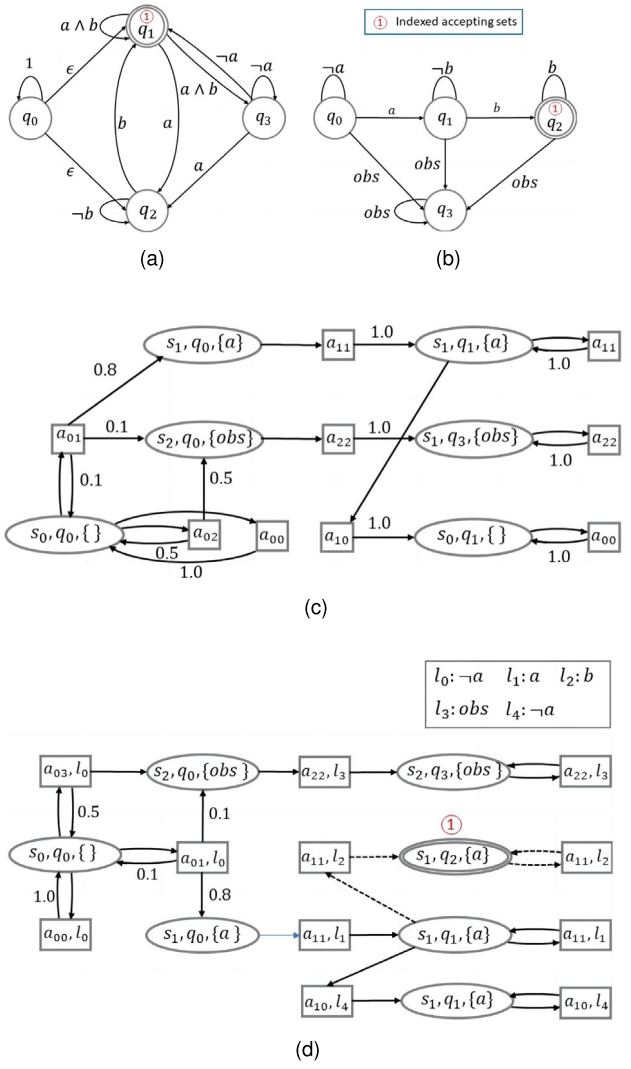


Fig. 7. Examples of the baseline comparisons regarding automaton structures and infeasible cases. (a) The LDBA of the LTL formula $\phi = \square\Diamond a \wedge \square\Diamond b$ only has one accepting set. (b) The LDGBA of the LTL formula $\phi = \Diamond(a \wedge \Diamond b)$. (c) The standard product MDP between the PL-MDP in Fig. 1(a) and the LDGBA of the LTL formula $\phi = \Diamond(a \wedge \Diamond b)$ in Fig. 7(b). There are no accepting states, and this is an infeasible case. (d) The relaxed product MDP (Definition 11) corresponding to the standard product MDP in (c). An accepting state exists, and the dashed arrows represent the transitions with nonzero violation costs.

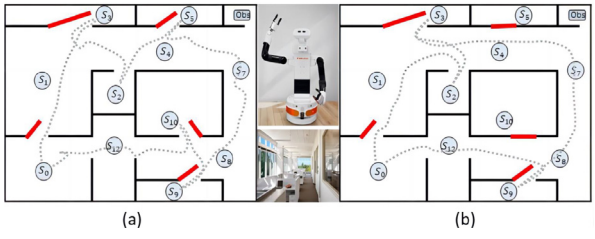


Fig. 8. Experimental trajectories for the task φ_{case4}.

ROS Gazebo, illustrated in Fig. 8, which consists of 7 rooms denoted by $S_0, S_2, S_3, S_5, S_9, S_{10}$, Obs and 5 corridors denoted by $S_1, S_4, S_7, S_8, S_{12}$. The TIAGo robot navigates collision-free from the center of one region to another, avoiding obstacles and ensuring no unintended crossings. To account for motion uncertainties, we assume the robot reaches its target region with a probability of 0.9, while there is a 0.1 probability it will move to an adjacent region instead. The resulting MDP comprises 12 states. The service to be performed by TIAGo is expressed as

$$\varphi_{case4} = \varphi_{all} \wedge \square\neg\text{Obs} \tag{16}$$

where $\varphi_{all} = \square\Diamond S_0 \wedge \square\Diamond S_2 \wedge \square\Diamond S_3 \wedge \square\Diamond S_5 \wedge \square\Diamond S_9 \wedge \square\Diamond S_{10}$. In (16), 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 Obs. Its corresponding LDGBA has 6 states with 6 accepting sets, and the relaxed product MDP has 72 states.

All room doors are open, except the doors of room S_5 S_{10} in Fig. 8(b). As a result, φ_{case4} cannot be fully completed in the case of Fig. 8(b), and the task is infeasible. It is also worth pointing out there do not exist AMECs in the corresponding product automaton $\overline{\mathcal{P}}$ or $\overline{\mathcal{P}}$ in Fig. 8 (a) because the robot has the nonzero probability of entering *Obs* at state S_7 as the motion uncertainties arise. The optimal policies for the two cases are generated, and each episode terminates after $\tau = 150$ steps with $\beta = 4$. The generated satisfying trajectories (without collision) of one round are shown in Fig. 8 (a) and (b), and the robot tries to complete the feasible part of task φ_{case4} in Fig. 8(b). To further illustrate the effectiveness of our approach, we consider the semantics of objects as atomic propositions in temporal logic formulas, following the method presented in [51,52]. Assuming that the environment depicted in Fig. 8 is unknown, the robot must explore the environment while executing the task. The experimental results are shown in Fig. 9. A video of the experiment is available at the following webpage.[1]

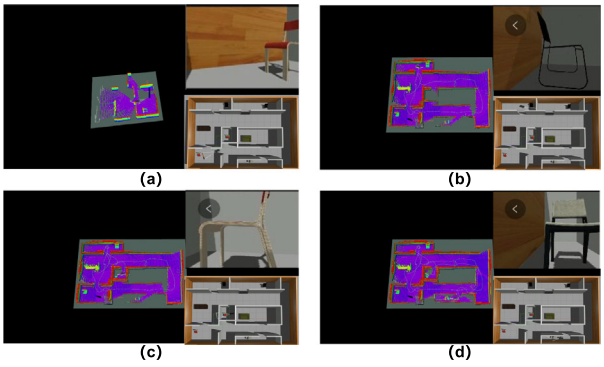


Fig. 9. The figure illustrates a quadrupedal robot navigating the environment while performing a task using the methods outlined in this article. In each sub-figure, the top right corner displays the robot's initial viewpoint, while the bottom right provides a synchronized image from Gazebo.

7. Conclusions

This paper presents a learning-based control synthesis of motion planning subject to motion and environment uncertainties. The LTL formulas are applied to express complex tasks via the automaton theory. Differently, in this work, an LTL formula is converted into a designed E-LDGBA to improve the performance of mission accomplishment and resolve the drawbacks of DRA and LDBA. Furthermore, the innovative relaxed product MDP and utility schemes consisting of the accepting reward and violation reward are proposed to accomplish the satisfaction of soft tasks. In order to provide formal guarantees of achieving the goals of multi-objective optimizations, future research will consider

[1] https://sites.google.com/d/1Gx374qPGlTeinU_8lS6vd6IVnVy-1kJp/p/13rdD73dGargssrmIq0yfXue6-nsiw3qc/edit.

more advanced multi-objective learning methods. Also problems over continuous state and/or action spaces will be studied by incorporating deep neural networks.

CRediT authorship contribution statement

Mingyu Cai: Formal analysis. **Zhangli Zhou:** Formal analysis, Conceptualization. **Lin Li:** Formal analysis, Conceptualization. **Shaoping Xiao:** Formal analysis, Conceptualization. **Zhen Kan:** Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62173314.

Data availability

No data was used for the research described in the article.

Appendix

A.1. Proof of Theorem 2

With reference to [41], we provide a proof for Theorem 2. For any policy $\boldsymbol{\pi}$, let $MC_{\mathcal{R}}^{\boldsymbol{\pi}}$ denote the Markov chain induced by π on $\overline{\mathcal{R}}$. Since $MC_{\mathcal{R}}^{\boldsymbol{\pi}}$ can be written as $MC_{\mathcal{R}}^{\boldsymbol{\pi}} = T_{\boldsymbol{\pi}} \sqcup R_{\boldsymbol{\pi}}^1 \sqcup R_{\boldsymbol{\pi}}^2 \dots R_{\boldsymbol{\pi}}^n$, (9) can be reformulated as

$$\begin{bmatrix} \boldsymbol{U}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{U}_{\boldsymbol{\pi}}^{rec} \end{bmatrix} = \sum_{n=0}^{\infty} \gamma^n \begin{bmatrix} \boldsymbol{P}_{\pi}(\mathcal{T},\mathcal{T}) & \boldsymbol{P}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{0}_{\sum_{i=1}^m N_i \times r} & \boldsymbol{P}_{\boldsymbol{\pi}}(R,R) \end{bmatrix}^n \cdot \left(\begin{bmatrix} \boldsymbol{\Lambda}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{\Lambda}_{\boldsymbol{\pi}}^{rec} \end{bmatrix} + \beta \begin{bmatrix} \boldsymbol{O}_{\boldsymbol{\pi}}^{tr} \\ \boldsymbol{O}_{\boldsymbol{\pi}}^{rec} \end{bmatrix} \right), \tag{17}$$

where $\boldsymbol{U}_{\boldsymbol{\pi}}^{tr}$ and $\boldsymbol{U}_{\boldsymbol{\pi}}^{rec}$ are the utilities of states in the transient and recurrent classes, respectively. In (17), $\boldsymbol{P}_{\pi}(\mathcal{T},\mathcal{T}) \in \mathbb{R}^{r\times r}$ denotes the probability transition matrix between states in $T^{\boldsymbol{\pi}}$. $\boldsymbol{P}_{\pi}^{tr} = \left[\boldsymbol{P}_{\pi}^{tr_1} \dots \boldsymbol{P}_{\pi}^{tr_m} \right] \in \mathbb{R}^{r\times\sum_{i=1}^m N_i}$ is a probability transition matrix where $\boldsymbol{P}_{\pi}^{tr_i} \in \mathbb{R}^{r\times N_i}$ represents the probability of transiting from a transient state in T^{π} to the states of $R_{\boldsymbol{\pi}}^i$. The $\boldsymbol{P}_{\boldsymbol{\pi}}(R,R)$ is a diagonal block matrix, where the ith block is an $N_i \times N_i$ matrix containing transition probabilities between states within $R_{\boldsymbol{\pi}}^j$. $\boldsymbol{P}^{\boldsymbol{\pi}}\left(\overline{\mathcal{R}},\overline{\mathcal{R}}\right)$ is a stochastic matrix since each block matrix is a stochastic matrix [48]. The rewards vector $\boldsymbol{\Lambda}_{\boldsymbol{\pi}}$ can also be partitioned to $\boldsymbol{\Lambda}_{\boldsymbol{\pi}}^{tr}$ and $\boldsymbol{\Lambda}_{\boldsymbol{\pi}}^{rec}$. Similarly, $\boldsymbol{O}_{\boldsymbol{\pi}} = \begin{bmatrix} O_{\boldsymbol{\pi}}(x_0) & O_{\boldsymbol{\pi}}(x_1) & \dots \end{bmatrix}^T = \boldsymbol{V}_{\boldsymbol{\pi}} \cdot \boldsymbol{1}_N$ can be divided into transient class $\boldsymbol{O}_{\boldsymbol{\pi}}^{tr}$ and recurrent class $\boldsymbol{O}_{\boldsymbol{\pi}}^{rec}$.

We prove this theorem by contradiction. Suppose there exists an optimal policy π^* not satisfying the acceptance condition of $\overline{\mathcal{R}}$. Based on Lemma 3, the following is true: $F_k^{\overline{\mathcal{R}}} \subseteq T_{\boldsymbol{\pi}^*}, \forall i \in \{1,\dots,f\}$. As a result, for any $j \in \{1,\dots,n\}$, we have $R_{\boldsymbol{\pi}^*}^j \cap F_i^{\overline{\mathcal{R}}} \neq \emptyset, \forall i \in \{1,\dots,f\}$.

The strategy is to show that there always exists a policy $\bar{\pi}$ with greater utility than $\boldsymbol{\pi}^*$, which contradicts to the optimality of π^*. Let us consider a state $x_R \in R_{\boldsymbol{\pi}^*}^j$ and let $\boldsymbol{P}_{\boldsymbol{\pi}^*}^{x_R R_{\boldsymbol{\pi}^*}^j}$ denote a row vector of $\boldsymbol{P}_{\boldsymbol{\pi}^*}^n(R,R)$ that contains the transition probabilities from x_R to the states in the same recurrent class $R_{\boldsymbol{\pi}^*}^j$ in n steps. The expected return $U_{\boldsymbol{\pi}^*}(x_R)$ of x_R under $\boldsymbol{\pi}^*$ is then obtained from (17) as: $U_{\boldsymbol{\pi}^*}^{rec}(x_R) = \sum_{n=0}^{\infty} \gamma^n \left[\boldsymbol{0}_{k_1}^T \; \boldsymbol{P}_{\boldsymbol{\pi}^*}^{x_R R_{\boldsymbol{\pi}^*}^j} \boldsymbol{0}_{k_2}^T \right] \left(\boldsymbol{\Lambda}_{\boldsymbol{\pi}^*}^{rec} + \beta \boldsymbol{O}_{\boldsymbol{\pi}^*}^{rec} \right)$ where $k_1 = \sum_{i=1}^{j-1} N_i$, $k_2 = \sum_{i=j+1}^{n} N_i$. Since $\overline{\mathcal{R}}_{\boldsymbol{\pi}^*}^j \cap F_i^{\overline{\mathcal{R}}} = \emptyset, \forall i \in \{1,\dots,f\}$, all the elements of $\boldsymbol{\Lambda}_{\boldsymbol{\pi}^*}^{rec}$ are equal to zero and each entry of $\boldsymbol{O}_{\pi^*}^{rec}$ is non-positive. We can conclude $U_{\pi^*}^{rec}(x_R) \leq 0$. To prove that optimal policy π^* raises a contradiction, the following analysis will show that $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) \geq U_{\boldsymbol{\pi}^*}^{rec}(x_R)$ for some policies $\bar{\boldsymbol{\pi}}$ that satisfy the acceptance condition of $\overline{\mathcal{R}}$. Thus we have $R_{\bar{\boldsymbol{\pi}}}^j \cap F_i^{\overline{\mathcal{R}}} \neq \emptyset, \forall i \in \{1,\dots,f\}$.

Case 1: If $x_R \in R_{\bar{\boldsymbol{\pi}}}^j$, by Lemma 3 and (7), there exist a minimum of f accepting states such that $X_{\Lambda} = \left\{ x_{\Lambda} \middle| x_{\Lambda} \in R_{\bar{\boldsymbol{\pi}}}^j \cap F_i^{\overline{\mathcal{R}}}, \forall i \in \{1,\dots,f\} \right\}$ in $R_{\bar{\boldsymbol{\pi}}}^j$ with positive rewards r_{Λ}. From (17), $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R)$ can be lower bounded as $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) \geq \sum_{n=0}^{\infty} \gamma^n \left(P_{\bar{\boldsymbol{\pi}}}^{x_R x_{\Lambda}} r_{\Lambda} + \beta \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{x_R R_{\bar{\boldsymbol{\pi}}}^j} \boldsymbol{V}_{\bar{\boldsymbol{\pi}}}^{rec} \boldsymbol{1}_{N_j} \right)$, where $P_{\bar{\boldsymbol{\pi}}}^{x_R x_{\Lambda}}$ is the transition probability from x_R to x_{Λ} in n steps, and $\boldsymbol{V}_{\bar{\boldsymbol{\pi}}}^{rec} \in \mathbb{R}^{N_j \times N_j}$ represents the violation cost of states in $\overline{\mathcal{R}}_{\bar{\boldsymbol{\pi}}}^j$. Since x_R and x_{Λ} are recurrent states, there always exists a lower bound $\underline{P}_{\bar{\boldsymbol{\pi}}}^{x_R x_{\Lambda}} \in \mathbb{R}^+$ of the transition probability $P_{\bar{\boldsymbol{\pi}}}^{x_R x_{\Lambda}}$. We can select a positive reward r_{Λ} such that

$$U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) \geq \underline{P}_{\bar{\boldsymbol{\pi}}}^{x_R x_{\Lambda}} r_{\Lambda} + \beta N_j^2 \underline{V}_{\bar{\boldsymbol{\pi}}}^{rec} \geq 0 \tag{18}$$

where $\underline{V}_{\bar{\boldsymbol{\pi}}}^{rec} \in \mathbb{R}^-$ represents the minimal entry in $\boldsymbol{V}_{\bar{\boldsymbol{\pi}}}^{rec}$. By selecting r_{Λ} to satisfy (18), we can conclude in this case $U_{\bar{\boldsymbol{\pi}}}^{rec}(x_R) > U_{\boldsymbol{\pi}^*}^{rec}(x_R)$.

Case 2:. If $x_R \in \mathcal{T}_{\bar{\boldsymbol{\pi}}}$, we know $\mathcal{T}_{\bar{\boldsymbol{\pi}}} \cap F_i^{\overline{\mathcal{R}}} = \emptyset, \forall i \in \{1,\dots,f\}$. As demonstrated in [48], for a transient state $x_{tr} \in \mathcal{T}_{\bar{\boldsymbol{\pi}}}$, there always exists an upper bound $\Delta < \infty$ such that $\sum_{n=0}^{\infty} p^n(x_{tr}, x_{tr}) < \Delta$, where $p^n(x_{tr}, x_{tr})$ denotes the probability of returning from a transient state x_{tr} to itself in n time steps. For a recurrent state $x_{rec} \in \overline{\mathcal{R}}_{\bar{\boldsymbol{\pi}}}^j$, it is always true that

$$\sum_{n=0}^{\infty} \gamma^n p^n(x_{rec}, x_{rec}) > \frac{1}{1-\gamma^{\bar{n}}} \bar{p} \tag{19}$$

where there exists $\bar{n}$ such that $p^{\bar{n}}(x_{rec}, x_{rec})$ is nonzero and can be lower bounded by $\bar{p}$ [48]. From (17), one has

$$\begin{aligned} \boldsymbol{U}_{\bar{\boldsymbol{\pi}}}^{tr} > & \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^n(\mathcal{T},\mathcal{T}) \left(\boldsymbol{\Lambda}_{\bar{\boldsymbol{\pi}}}^{tr} + \beta \boldsymbol{O}_{\bar{\boldsymbol{\pi}}}^{tr} \right) \\ & + \sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr} \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^n \left(\overline{\mathcal{R}},\overline{\mathcal{R}} \right) \left(\boldsymbol{\Lambda}^{tec} + \beta \boldsymbol{O}_{\bar{\boldsymbol{\pi}}}^{rec} \right) \end{aligned} \tag{20}$$

Let $\max(\cdot)$ and $\min(\cdot)$ represent the maximum and minimum entry of an input vector, respectively. The upper bound $\bar{m} = \left\{ \max\left(\overline{M}\right) \middle| \overline{M} < \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^{tr} \bar{\boldsymbol{P}} \left(\boldsymbol{\Lambda}_{\bar{\boldsymbol{\pi}}}^{rec} + \beta \boldsymbol{V}_{\bar{\boldsymbol{\pi}}}^{rec} \boldsymbol{1}_{\tilde{N}} \right) \right\}$, where $\tilde{N} = \sum_{j=1}^{m} N_j$ and $\bar{\boldsymbol{P}}$ is a block matrix whose nonzero entries are derived similarly to the $\bar{p}$ in (19). Using the fact $\sum_{n=0}^{\infty} \gamma^n \boldsymbol{P}_{\bar{\boldsymbol{\pi}}}^n(\mathcal{T},\mathcal{T}) \leq \Delta \boldsymbol{1}_{r\times r}$ [48], where $\boldsymbol{1}_{r\times r}$ is a $r \times r$ matrix of all ones, the utility $U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R)$ can be lower bounded from (19) and (20) as

$$U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > \Delta \cdot r \cdot \beta \underline{V}_{\bar{\boldsymbol{\pi}}}^{tr} + \frac{1}{1-\gamma^{\bar{n}}} \bar{m} \tag{21}$$

Since $U_{\boldsymbol{\pi}^*}^{rec}(x_R) = 0$, the contradiction $U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > 0$ in this case will be achieved if $\Delta \cdot r \cdot \beta \underline{V}_{\bar{\boldsymbol{\pi}}}^{tr} + \frac{1}{1-\gamma^{\bar{n}}} \bar{m} \geq 0$. Because $\Delta r \left(w_L + \beta \underline{V}_{\bar{\boldsymbol{\pi}}}^{tr} \right) < 0$, it needs $\bar{m} > 0$ as

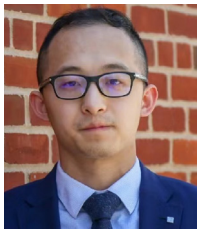
$$\bar{m} > r_{\Lambda} + \beta \tilde{N} \underline{V}_{\bar{\boldsymbol{\pi}}}^{rec} > 0. \tag{22}$$

Thus, by choosing γ to be sufficiently close to 1 with $\bar{m} > 0$, we have $U_{\bar{\boldsymbol{\pi}}}^{tr}(x_R) > 0 \geq U_{\boldsymbol{\pi}^*}^{rec}(x_R)$. The above procedure shows the contradiction of the assumption that $\boldsymbol{\pi}_V^*$ is optimal.

References

[1] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[2] M. Kloetzer, C. Mahulea, LTL-based planning in environments with probabilistic observations, IEEE Trans. Autom. Sci. Eng. 12 (4) (2015) 1407–1420.
[3] Y. Kantaros, M.M. Zavlanos, STyLuS*: A temporal logic optimal control synthesis algorithm for large-scale multi-robot systems, Int. J. Robot. Res. 39 (7) (2020) 812–836.
[4] A. Pacheck, H. Kress-Gazit, Physically feasible repair of reactive, linear temporal logic-based, high-level tasks, IEEE Trans. Robot. (2023).
[5] X. Sun, Y. Shoukry, Neurosymbolic motion and task planning for linear temporal logic tasks, IEEE Trans. Robot. (2024).

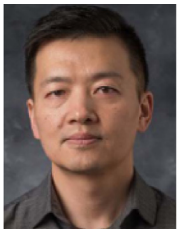
[6] Z. Chen, M. Cai, Z. Zhou, L. Li, Z. Kan, Fast motion planning in dynamic environments with extended predicate-based temporal logic, IEEE Trans. Autom. Sci. Eng. (2024).
[7] Z. Chen, Z. Kan, Real-time reactive task allocation and planning of large heterogeneous multi-robot systems with temporal logic specifications, The International Journal of Robotics Research (2024).
[8] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
[9] M. Guo, T. Liao, J. Wang, Z. Li, Hierarchical motion planning under probabilistic temporal tasks and safe-return constraints, IEEE Trans. Autom. Control 68 (11) (2023) 6727–6742.
[10] X.C. Ding, S.L. Smith, C. Belta, D. Rus, MDP optimal control under temporal logic constraints, in: Proc. IEEE Conf. Decis. Control., 2011, pp. 532–538.
[11] X. Ding, S.L. Smith, C. Belta, D. Rus, Optimal control of Markov decision processes with linear temporal logic constraints, IEEE Trans. Autom. Control 59 (5) (2014) 1244–1257.
[12] A. Ulusoy, T. Wongpiromsarn, C. Belta, Incremental controller synthesis in probabilistic environments with temporal logic constraints, Int. J. Robot. Res. 33 (8) (2014) 1130–1144.
[13] M. Guo, M.M. Zavlanos, Probabilistic motion planning under temporal tasks and soft constraints, IEEE Trans. Autom. Control 63 (12) (2018) 4051–4066.
[14] B. Lacerda, F. Faruq, D. Parker, N. Hawes, Probabilistic planning with formal performance guarantees for mobile service robots, Int. J. Robot. Res. 38 (9) (2019) 1098–1123.
[15] C.I. Vasile, X. Li, C. Belta, Reactive sampling-based path planning with temporal logic specifications, Int. J. Robot. Res. (2020) 0278364920918919.
[16] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, D. Rus, Minimum-violation scltl motion planning for mobility-on-demand, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 1481–1488.
[17] M. Cai, H. Peng, Z. Li, H. Gao, Z. Kan, Receding horizon control-based motion planning with partially infeasible LTL constraints, IEEE Control Syst. Lett. 5 (4) (2020) 1279–1284.
[18] M. Cai, S. Xiao, Z. Li, Z. Kan, Optimal probabilistic motion planning with potential infeasible LTL constraints, IEEE Trans. Autom. Control (2021).
[19] N. Mehdipour, M. Althoff, R.D. Tebbens, C. Belta, Formal methods to comply with rules of the road in autonomous driving: State of the art and grand challenges, Automatica 152 (2023) 110692.
[20] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292.
[21] C. Sun, X. Li, C. Belta, Automata guided semi-decentralized multi-agent reinforcement learning, in: 2020 American Control Conference, ACC, IEEE, 2020, pp. 3900–3905.
[22] H. Wang, H. Zhang, L. Li, Z. Kan, Y. Song, Task-driven reinforcement learning with action primitives for long-horizon manipulation skills, IEEE Trans. Cybern. (2023).
[23] Y. Meng, C. Fan, Signal temporal logic neural predictive control, IEEE Robot. Autom. Lett. (2023).
[24] T. Brázdil, K. Chatterjee, M. Chmelik, V. Forejt, J. Křetínskỳ, M. Kwiatkowska, D. Parker, M. Ujma, Verification of Markov decision processes using learning algorithms, in: Int. Symp. Autom. Tech. Verif. Anal., Springer, 2014, pp. 98–114.
[25] J. Fu, U. Topcu, Probably approximately correct MDP learning and control with temporal logic constraints, 2014, arXiv preprint arXiv:1404.7073.
[26] D. Sadigh, E.S. Kim, S. Coogan, S.S. Sastry, 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.
[27] J. Wang, X. Ding, M. Lahijanian, I.C. Paschalidis, C.A. Belta, Temporal logic motion control using actor–critic methods, Int. J. Robot. Res. 34 (10) (2015) 1329–1344.
[28] D. Aksaray, A. Jones, Z. Kong, M. Schwager, C. Belta, Q-learning for robust satisfaction of signal temporal logic specifications, in: Proc. IEEE Conf. Decis. Control, 2016, pp. 6565–6570.
[29] X. Li, Z. Serlin, G. Yang, C. Belta, A formal methods approach to interpretable reinforcement learning for robotic planning, Science Robotics 4 (37) (2019).
[30] E.M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, D. Wojtczak, Omega-regular objectives in model-free reinforcement learning, in: Int. Conf. Tools Alg. Constr. Anal. Syst., Springer, 2019, pp. 395–412.
[31] A.K. Bozkurt, Y. Wang, M.M. Zavlanos, M. Pajic, Control synthesis from linear temporal logic specifications using model-free reinforcement learning, 2019, arXiv preprint arXiv:1909.07299.
[32] K. Kim, G.E. Fainekos, Approximate solutions for the minimal revision problem of specification automata, in: IEEE Int. Conf. Intell. Robot. and Syst., 2012, pp. 265–271.
[33] K. Kim, G.E. Fainekos, S. Sankaranarayanan, On the revision problem of specification automata, in: Proc. IEEE Int. Conf. Robot. Autom., 2012, pp. 5171–5176.
[34] J. Tumova, G.C. Hall, S. Karaman, E. Frazzoli, D. Rus, Least-violating control strategy synthesis with safety rules, in: Proc. Int. Conf. Hybrid Syst., Comput. Control, 2013, pp. 1–10.
[35] M. Guo, D.V. Dimarogonas, Multi-agent plan reconfiguration under local LTL specifications, Int. J. Robot. Res. 34 (2) (2015) 218–235.
[36] M. Cai, S. Xiao, B. Li, Z. Li, Z. Kan, Reinforcement learning based temporal logic control with maximum probabilistic satisfaction, in: Int. Conf. Robot. Autom., IEEE, 2021, pp. 806–812.
[37] R. Alur, O. Bastani, K. Jothimurugan, M. Perez, F. Somenzi, A. Trivedi, Policy synthesis and reinforcement learning for discounted LTL, in: International Conference on Computer Aided Verification, Springer, 2023, pp. 415–435.
[38] Q. Gao, D. Hajinezhad, Y. Zhang, Y. Kantaros, 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.
[39] M. Cai, H. Peng, Z. Li, Z. Kan, Learning-based probabilistic LTL motion planning with environment and motion uncertainties, IEEE Trans. Autom. Control (2020).
[40] M. Cai, M. Mann, Z. Serlin, K. Leahy, C.-I. Vasile, Learning minimally-violating continuous control for infeasible linear temporal logic specifications, in: American Control Conference, 2023.
[41] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G.J. Pappas, I. Lee, Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees, 2019, arXiv preprint arXiv:1909.05304.
[42] M. Cai, M. Hasanbeig, S. Xiao, et al., Modular deep reinforcement learning for continuous motion planning with temporal logic, IEEE Robot. Autom. Lett. 6 (4) (2021) 7973–7980.
[43] J. Wang, H. Hasanbeig, K. Tan, et al., Mission-driven exploration for accelerated deep reinforcement learning with temporal logic task specifications, 2023, arXiv preprint arXiv:2311.17059.
[44] M. Hasanbeig, A. Abate, D. Kroening, Certified reinforcement learning with logic guidance, 2019, arXiv preprint arXiv:1902.00778.
[45] S. Sickert, J. Esparza, S. Jaax, J. Křetínskỳ, Limit-deterministic büchi automata for linear temporal logic, in: Int. Conf. Comput. Aided Verif., Springer, 2016, pp. 312–332.
[46] E.M. Hahn, G. Li, S. Schewe, A. Turrini, L. Zhang, Lazy probabilistic model checking without determinisation, 2013, arXiv preprint arXiv:1311.2928.
[47] J. Kretínský, T. Meggendorfer, S. Sickert, Owl: A library for ω-words, automata, and LTL, in: Autom. Tech. Verif. Anal., Springer, 2018, pp. 543–550.
[48] R. Durrett, second ed., Essentials of Stochastic Processes, vol. 1, Springer, 2012.
[49] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, E. Dekker, Empirical evaluation methods for multiobjective reinforcement learning algorithms, Mach. Learn. 84 (1–2) (2011) 51–80.
[50] J.N. Tsitsiklis, Asynchronous stochastic approximation and Q-learning, Mach. Learn. 16 (3) (1994) 185–202.
[51] Z. Zhou, S. Wang, Z. Chen, M. Cai, H. Wang, Z. Li, Z. Kan, Local observation based reactive temporal logic planning of human-robot systems, IEEE Trans. Autom. Sci. Eng. (2023) 1–13.
[52] Z. Zhou, Z. Chen, M. Cai, Z. Li, Z. Kan, C.-Y. Su, Vision-based reactive temporal logic motion planning for quadruped robots in unstructured dynamic environments, IEEE Trans. Ind. Electron. (2023) 1–10.



Mingyu Cai received the Ph.D. degree in mechanical engineering from the University of Iowa, Iowa City, IA, USA, in 2021. From 2021 to 2023, he was a Postdoctoral Associate with the Department of Mechanical Engineering at Lehigh University, Bethlehem, USA. In 2023, he worked at Honda Research Institute as a research scientist. From 2024, he is an assistant professor at the University of California, Riverside. His research interests encompass robotics, machine learning, control theory, and formal methods, with a focus on applications in motion planning, decision-making, nonlinear control, and autonomous driving.

Zhangli Zhou received his B.E. degree from Wuhan University of Technology, China, in 2016, and his Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2024. He is currently a Postdoctoral Fellow at the University of Science and Technology of China, with research interests in robotics and reactive task and motion planning.

Lin Li received the B.E. degree in mechanical engineering and automation from the Hefei University of Technology, Hefei, China, in 2021. She is currently pursuing the M.S. degree in control engineering with the University of Science and Technology of China, Hefei. Her current research interests include heterogeneous multirobot system and motion planning.



Shaoping Xiao received the Ph.D. degree in mechanical engineering from Northwestern University in 2003. He is currently an Associate Professor with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA, USA. His primary expertise lies in computational mechanics and materials science, and one of his papers has been cited over 1000 times. In recent years, he has extended his research to artificial intelligence (AI) and its applications in engineering problem-solving. His research interests include machine learning-enhanced numerical modeling of composite materials, reinforcement learning and formal methods for robotics control, AI-powered design of distributed reservoir systems for flood mitigation, intelligent traffic light systems, and quantum computing.

Zhen Kan (Senior Member, IEEE) received his Ph.D. degree in 2011 from the Department of Mechanical and Aerospace Engineering at the University of Florida. He was a Postdoctoral Research Fellow with the Air Force Research Laboratory (AFRL) at Eglin AFB and University of Florida REEF from 2012 to 2016, and was an Assistant Professor in the Department of Mechanical Engineering at the University of Iowa from 2016–2019. He is currently a Professor in the Department of Automation at the University of Science and Technology of China. His research interests include networked control systems, nonlinear control, formal methods, and robotics. He currently serves on program committees of several internationally recognized scientific and engineering conferences and is an Associate Editor of IEEE Transactions on Automatic Control.