Optimal Probabilistic Motion Planning With Potential Infeasible LTL Constraints

Mingyu Cai, *Member, IEEE*, Shaoping Xiao, Zhijun Li, *Senior Member, IEEE*, and Zhen Kan, *Member, IEEE*

***Abstract*—This paper studies optimal motion planning subject to motion and environment uncertainties. By modeling the system as a probabilistic labeled Markov decision process (PL-MDP), the control objective is to synthesize a finite-memory policy, under which the agent satisfies complex high-level tasks expressed as linear temporal logic (LTL) with desired satisfaction probability. In particular, the cost optimization of the trajectory that satisfies infinite horizon tasks is considered, and the trade-off between reducing the expected mean cost and maximizing the probability of task satisfaction is analyzed. The LTL formulas are converted to limit-deterministic Büchi automata (LDBA) with a reachability acceptance condition and a compact graph structure. The novelty of this work lies in considering the cases where LTL specifications can be potentially infeasible and developing a relaxed product MDP between PL-MDP and LDBA. The relaxed product MDP allows the agent to revise its motion plan whenever the task is not fully feasible and quantify the revised plan's violation measurement. A multi- objective optimization problem is then formulated to jointly consider the probability of task satisfaction, the violation with respect to original task constraints, and the implementation cost of the policy execution. The formulated problem can be solved via coupled linear programs. This work first bridges the gap between probabilistic planning revision of potential infeasible LTL specifications and optimal control synthesis of both plan prefix and plan suffix of the trajectory over the infinite horizons. Experimental results are provided to demonstrate the effectiveness of the proposed framework.**

***Index Terms*—Decision-making, formal methods in robotics and automation, linear programming (LP), motion planning, network flow, optimal control, probabilistic model checking.**

I. Introduction

AUTONOMOUS agents operating in complex environments are often subject to a variety of uncertainties. Typical uncertainties arise from the stochastic behaviors of the motion (e.g., potential sensing noise or actuation failures) and uncertain environment properties (e.g., mobile obstacles or time-varying areas of interest). In addition to motion and environment uncertainties, another layer of complexity in robotic motion planning is the feasibility of desired behaviors. For instance, areas of interest to be visited can be found to be prohibitive to the agent in practice (e.g., surrounded by water that the ground robot cannot traverse), resulting in that the user-specified tasks cannot be fully realized. Motivated by these challenges, this work considers motion planning of a mobile agent with potentially infeasible task specifications subject to motion and environment uncertainties, i.e., motion planning and decision-making of stochastic systems.

Linear temporal logic (LTL) is a formal language capable of describing complex missions [1]. For example, motion planning with LTL task specifications has generated substantial interest in robotics (cf. [2]–[4], to name a few). Recently, there has been growing attention in the control synthesis community to address Markov decision process (MDP) with LTL specifications based on probabilistic model checking, such as cosafe LTL tasks [5], [6], computation tree logic tasks [7], stochastic signal temporal logic tasks [8], and reinforcement-learning-based approaches [9]–[13]. However, these aforementioned works only considered feasible specifications that can be fully executed. Thus, a challenging problem is how missions can be successfully managed in a dynamic and uncertain environment, where the desired tasks are only partially feasible.

This work studies the control synthesis of a mobile agent with LTL specifications that can be infeasible. The uncertainties in both robot motion (e.g., potential actuation failures) and workspace properties (e.g., obstacles or areas of interest) are considered. It gives rise to the probabilistic labeled Markov decision process (PL-MDP). Our objective is to generate control policies in decreasing priority order to 1) accomplish the prespecified task with desired probability; 2) fulfill the prespecified task as much as possible if it is not entirely feasible; and 3) minimize the expected implementation cost of the trajectory.

Manuscript received 28 September 2021; revised 24 November 2021; accepted 18 December 2021. Date of publication 28 December 2021; date of current version 28 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62173314, Grant U2013601 and Grant 61625303. Recommended by Associate Editor C. Mahulea. *(Corresponding author: Zhen Kan.)*

Mingyu Cai is with the Department of Mechanical Engineering, Lehigh University, Bethlehem, PA 18015 USA, and also with the Department of Mechanical Engineering, University of Iowa Technology Institute, The University of Iowa, Iowa City, IA 52246 USA (e-mail: mic221@lehigh.edu).

Shaoping Xiao is with the Department of Mechanical Engineering, University of Iowa Technology Institute, The University of Iowa, Iowa City, IA 52246 USA (e-mail: shaoping-xiao@uiowa.edu).

Zhijun Li and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: zjli@ieee.org; zkan@ustc.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2021.3138704.

Digital Object Identifier 10.1109/TAC.2021.3138704

Although the above objectives have been studied individually in the literature, this work considers them together in a probabilistic manner.

Related Works: From the aspect of optimization, the satisfaction of the general form of LTL tasks in stochastic systems involves the lasso-type policies comprised of a plan prefix and a plan suffix [1]. When considering cost optimization subject to LTL specifications with infinite horizons over MDP models, the planned policies generally have a decision-making structure consisting of plan prefix and plan suffix. The prefix policies drive the system into an accepting maximum end component (AMEC), and the suffix policies involve the behaviors within the AMEC [1]. Optimal policies of prefix and suffix structures have been investigated in the literature [14]–[19]. A suboptimal solution was developed in [14], and minimizing the bottleneck cost was considered in [15]. The works of [16]–[19] optimized the total cost of plan prefix and suffix, while maximizing the satisfaction probability of specific LTL tasks. However, the aforementioned works [14]–[19] mainly focused on motion planning in feasible cases and relied on a critical assumption of the existence of AMECs or an accepting run under a policy in the product MDP. Such an assumption may not be valid if desired tasks can not be fully completed in the operating environment.

When considering infeasible tasks, motion planning in a potential conflict situation has been partially investigated via control synthesis under soft LTL constraints [20] and the minimal revision of motion plans [21]–[23]. Recent works [24]–[26] extend the above approaches by considering dynamic or time-bounded temporal logic constraints. The works of [27] and [28] leverage sampling-based methods for traffic environments. However, only deterministic transition systems were considered in [20]–[28]. On the other hand, when considering probabilistic systems, a learning-based approach was utilized in the works of [29] and [30]. However, these works do not provide formal guarantees for multiobjective problems. The iterative temporal planning was developed in [31] and [32] with partial satisfaction guarantees, and the work [33] proposed a minimum violation control for finite stochastic games subject to cosafe LTL. These results are limited to finite horizons. In contrast, the satisfaction of the general LTL tasks in stochastic systems involves the lasso-type policies comprised of prefix and suffix structures [1]. This work considers decision-making over infinite horizons in a stochastic system, where desired tasks might not be fully feasible. In addition, this work also studies probabilistic cost optimization of the agent trajectory, which receives little attention in the works of [20]–[31].

From the perspective of automaton structures, limit-deterministic Büchi automata (LDBA) are often used instead of traditional deterministic Rabin automata (DRA) [1] to reduce the automaton size. It is well known that the Rabin automata, in the worst case, are doubly exponential in the size of the LTL formula, while LDBA only has an exponential-sized automaton [34]. In addition, the Büchi accepting condition of LDBA, unlike the Rabin accepting condition, does not apply rejecting transitions. It allows us to constructively convert the problem of satisfying the LTL formula to an almost-sure reachability problem [35]–[37]. As a result, LDBA-based control synthesis has been increasingly used for motion planning with LTL constraints [35]–[37]. However, in the aforementioned works, cost optimization was not considered, and most of them only considered feasible cases (i.e., with goals to reach AMECs). In this work, the product MDP with LDBA is extended to the relaxed product MDP, which facilitates the optimization process to handle infeasible LTL specifications, reduces the automaton complexity, and improves the computational efficiency,

Contributions: Our work, for the first time, bridges the gap between planning revision for potentially infeasible task specifications and optimal control synthesis of stochastic systems subject to motion and environment uncertainties. In addition, we analyze the finite-memory policy of the PL-MDP that satisfies complex LTL specifications with desired probability and consider cost optimization in both plan prefix and plan suffix of the agent trajectory over infinite horizons. The novelty of this work is the development of a relaxed product MDP between PL-MDP and LDBA to address the cases in which LTL specifications can be potentially infeasible. The relaxed product MDP allows the agent to revise its motion plan whenever the task is not fully feasible and quantify the revised plan's violation measurement. In addition, the relaxed product structure is verified to be an MDP model and a more connected directed graph. Based on such a relaxed product MDP, we are able to formulate a constrained multiobjective optimization process to jointly consider the desired lower bounded satisfaction probability of the task, the minimum violation cost, and the optimal implementation costs. We can find solutions by adopting coupled linear programming (LP) for MDPs relying on the network flow theory [18], [19], which is flexible for any optimal probabilistic model checking problems. We provide a comprehensive comparison with the significant existing methods, i.e., Round-Robin policy [1], PRISM [38], and multiobjective optimization frameworks [17]–[19]. Although the relaxed product MDP is designed to handle potentially infeasible LTL specifications, it is worth pointing out it is also applicable to feasible cases, and, thus, generalizes most existing works. In addition, this framework can be easily adapted to formulate a hierarchical architecture that combines noisy low-level controllers and practical approaches of stochastic abstraction.

II. Preliminaries

A. Notations

$\mathbb{N}$ represents the set of natural numbers. For an infinite path $\boldsymbol{s} = s_0 s_1 \dots$ starting from state s_0, $\boldsymbol{s}[0]$ denotes its first element, $\boldsymbol{s}[t], t \in \mathbb{N}$ denotes the path at step t, $\boldsymbol{s}[t:]$ denotes the path starting from step t to the end. The expected value of a variable x is $\mathbb{E}(x)$. We use abbreviations for several notations and definitions, which are summarized in Table I.

B. Probabilistic Labeled MDP

Definition 1: A PL-MDP is a tuple $\mathcal{M} = (S, A, p_S, (s_0, l_0), L, p_L, c_A)$, where S is a finite state space, A is a finite action space (with a slight abuse of notation,

TABLE I
ABBREVIATION SUMMARY OF NOTATIONS

Notation Name	Abbreviation
Limit-Deterministic Büchi Automaton	LDBA
Strong Connected Component	SCC
Bottom Strong Connect Component	BSCC
Accepting Bottom Strong Connect Component	ABSCC
Maximum End Component	MEC
Accepting Maximum End Component	AMEC
Average Execution Cost per Stage	AEPS
Average Violation Cost per Stage	AVPS
Average Regulation Cost per Stage	ARPS
Linear Programming	LP

$A(s)$ also denotes the set of actions enabled at $s \in S$), $p_S : S \times A \times S \to [0,1]$ is the transition probability function, $\boldsymbol{\pi}$ is a set of atomic propositions, and $L : S \to 2^{\boldsymbol{\pi}}$ is a labeling function. The pair (s_0, l_0) denotes an initial state $s_0 \in S$ and an initial label $l_0 \in L(s_0)$. The function $p_L(s,l)$ denotes the probability of $l \subseteq L(s)$ associated with $s \in S$ satisfying $\sum_{l \in L(s)} p_L(s,l) = 1 \forall s \in S$. The cost function $c_A(s,a)$ indicates the cost of performing $a \in A(s)$ at s. The transition probability p_S captures the motion uncertainties of the agent, while the labeling probability p_L captures the environment uncertainties.

The PL-MDP $\mathcal{M}$ evolves by taking actions a_i selected based on the policy at each step $i \in \mathbb{N}_0$, where $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

Definition 2: The control policy $\boldsymbol{\mu} = \mu_0 \mu_1 \ldots$ is a sequence of decision rules, which yields a path $\boldsymbol{s} = s_0 s_1 s_2 \ldots$ over $\mathcal{M}$. As shown in [39], $\boldsymbol{\mu}$ is called a stationary policy if $\boldsymbol{\mu}_i = \boldsymbol{\mu}$ for all $i \geq 0$, where $\boldsymbol{\mu}$ can be either deterministic, such that $\boldsymbol{\mu} : S \to A$ or stochastic such that $\boldsymbol{\mu} : S \times A \to [0,1]$. The control policy $\boldsymbol{\mu}$ is memoryless if each μ_i only depends on its current state s_i. In contrast, $\boldsymbol{\mu}$ is called a finite memory (i.e., history-dependent) policy if μ_i depends on its past states.

In this work, we consider the stochastic policy. Let $\boldsymbol{\mu}(s)$ denote the probability distribution of actions at state s, and $\boldsymbol{\mu}(s,a)$ represent the probability of generating action a at state s using the policy $\boldsymbol{\mu}$.

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

Definition 4: 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) $\{x' \in X' \mid p^{\mathcal{P}}(x,u,x') > 0 \forall x \in X' \text{ and } \forall u \in U'(x)\}$. 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 $\mathcal{G}_{(S,'A')}$. A sub-MDP is a strongly connected component (SCC) if its induced graph is strongly connected such that for all pairs of nodes $s, s' \in S'$, there is a path from s to s'. A bottom SCC (BSCC) is an SCC from which no state outside is reachable by applying the restricted action space.

Remark 1: Note the evolution of a sub-MDP $\mathcal{M}_{(S,'A')}$ is restricted by the action space A'. Given a PL-MDP and one of its SCCs, there may exist paths starting within the SCC and ending outside of the SCC, whereas all paths starting from a BSCC will always stay within the same BSCC. In addition, a Markov chain $MC_{\mathcal{M}}^{\boldsymbol{\pi}}$ is a sub-MDP of $\mathcal{P}$ induced by a policy $\boldsymbol{\pi}$, and its evolution is restricted by the policy $\boldsymbol{\mu}$.

Definition 5: [1] A sub-MDP $\mathcal{M}_{(S,'A')}$ is called an end component (EC) of $\mathcal{M}$ if it is a BSCC. An EC $\mathcal{M}_{(S,'A')}$ is called a maximal EC (MEC) if there is no other EC $\mathcal{M}_{(S,''A'')}$, such that $S' \subseteq S''$ and $A'(s) \subseteq A''(s), \forall s \in S$.

C. LTL and Limit-Deterministic Büchi Automaton

LTL is a formal language to describe the high-level specifications of a system. The ingredients of an LTL formula are a set of atomic propositions and combinations of several Boolean and temporal operators. The syntax of an LTL formula is defined inductively as

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

where $a \in AP$ is an atomic proposition, True, negation $\neg$, and conjunction $\wedge$ are propositional logic operators, and next $\bigcirc$ and until $\mathcal{U}$ are temporal operators. The satisfaction relationship is denoted as $\models$. The semantics of an LTL formula are interpreted over words, which is an infinite sequence $o = o_0 o_1 \ldots$, where $o_i \in 2^{AP}$ for all $i \geq 0$, and 2^{AP} represents the power set of AP, which are defined as

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

Alongside the standard operators introduced above, other propositional logic operators such as false, disjunction $\vee$, implication $\rightarrow$, and temporal operators always $\Box$, eventually $\Diamond$ can be derived as usual. Thus, an LTL formula describes a set of infinite traces through S. Given an LTL formula that specifies the missions, its satisfaction can be evaluated by an LDBA [34], [40].

Definition 6: An LDBA is a tuple $\mathcal{A} = (Q, \Sigma \cup \{\epsilon\}, \delta, q_0, F)$, where Q is a finite set of states, $\Sigma = 2^{\boldsymbol{AP}}$ is a finite alphabet, $\{\epsilon\}$ is a set of indexed epsilons, each of which is enabled for one $\epsilon-$transition, $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to 2^Q$ is a transition function, $q_0 \in Q$ is an initial state, and F is a set of accepting states. The states Q can be partitioned into a deterministic set Q_D and a nondeterministic set Q_N, i.e., $Q = Q_D \cup Q_N$, where

1) the state transitions in Q_D are total and restricted within it, i.e., $\left|\delta(q,\alpha)\right| = 1$ and $\delta(q,\alpha) \subseteq Q_D$ for every state $q \in Q_D$ and $\alpha \in \Sigma$;
2) the ϵ-transitions are only defined for state transitions from Q_N to Q_D, and are not allowed in the deterministic set, i.e., for any $q \in Q_D$, $\delta(q,\epsilon) = \emptyset \forall \epsilon \in \{\epsilon\}$;
3) the accepting states are only in the deterministic set, i.e., $F \subseteq Q_D$.

An $\epsilon-$transition allows an automaton to change its state without reading any atomic proposition. The run $\boldsymbol{q} = q_0 q_1 \dots$ is accepted by the LDBA, if it satisfies the Büchi condition, i.e., $\inf(\boldsymbol{q}) \cap F \neq \emptyset$, where $\inf(\boldsymbol{q})$ denotes the set of states that is visited infinitely often. As discussed in [41], the probabilistic verification of automaton does not need to be fully deterministic. In other words, the automata-theoretic approach still works if the restricted forms of nondeterminism are allowed. Therefore, LDBA has been applied for the qualitative and quantitative analysis of MDPs [34], [40]–[42]. To convert an LTL formula to an LDBA, see [40]. In the following analysis, we use $\mathcal{A}_\phi$ to denote the LDBA corresponding to an LTL formula ϕ.

III. Problem Statement

Consider an LTL task specification ϕ over $\boldsymbol{\pi}$ and a PL-MDP $\mathcal{M} = (S, A, p_S, (s_0, l_0), \boldsymbol{\pi}, L, p_L, c_A)$. It is assumed that the agent can sense its current state and the associated labels. $\boldsymbol{\mu}(s_k, \mu_k)$ represents the probability of selecting the control input μ_k at time k for state s_k using policy $\boldsymbol{\mu}$. The agent's path $\boldsymbol{s}^{\boldsymbol{\mu}}_\infty = s_0 l_0 \dots s_i l_i s_{i+1} l_{i+1} \dots$ under a control sequence $\boldsymbol{\mu}_\infty = \mu_0 \mu_1 \dots$ is generated based on policy $\boldsymbol{\mu}$ such that $s_{i+1} \in \{s \in S | p_S(s_i, \mu_i, s) > 0\}$, $\boldsymbol{\mu}(s_i, \mu_i) > 0$, and $l_i \in L(s_i)$ with $p_L(s_i, l_i) > 0$. Let $L(\boldsymbol{s}^{\boldsymbol{\mu}}_\infty) = l_0 l_1 \dots l_i l_{i+1} \dots$ be the sequence of labels associated with $\boldsymbol{s}^{\boldsymbol{\mu}}_\infty$, and denote by $L(\boldsymbol{s}^{\boldsymbol{\mu}}_\infty) \models \phi$ if $\boldsymbol{s}^{\boldsymbol{\mu}}_\infty$ satisfies ϕ. The probability measurement of a run $\boldsymbol{s}^{\boldsymbol{\mu}}_\infty$ can be uniquely obtained by

$$\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\boldsymbol{s}^{\boldsymbol{\mu}}_\infty) = \prod_{i=0}^{n} p_L(s_i, l_i) \cdot p_S(s_i, \boldsymbol{\mu}_i(s_i), s_{i+1}) \cdot \boldsymbol{\mu}(s_i, \mu_i). \quad (1)$$

Then, the satisfaction probability under $\boldsymbol{\mu}$ from an initial state s_0 can be computed as

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

where $\boldsymbol{S}^{\boldsymbol{\mu}}_\infty$ is a set of all admissible paths under policy $\boldsymbol{\mu}$.

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.

Note that, according to Definition 7, an infeasible case means there exist no policies to satisfy the task, which can be interpreted as $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = 0$.

Definition 8: The expected average execution cost per stage (AEPS) of a PL-MDP $\mathcal{M}$ under the policy $\boldsymbol{\mu}$ is defined as

$$J_E(\mathcal{M}^{\boldsymbol{\mu}}) = \mathbb{E}^{\boldsymbol{\mu}}_{\mathcal{M}}\left[\limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} c_A(s_i, a_i)\right] \quad (3)$$

where a_i is the action generated based on the policy $\boldsymbol{\mu}(s_i)$.

A common objective in the literature is to find a policy $\boldsymbol{\mu}$, such that $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi)$ is greater than the desired satisfaction probability, while minimizing the expected AEPS. However, when operating in a real-world environment with uncertainties in the dynamic system, the user-specified mission ϕ might not be fully feasible, resulting in $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = 0$ since there may not exist a path $\boldsymbol{s}^{\boldsymbol{\mu}}_\infty$ such that $L(\boldsymbol{s}^{\boldsymbol{\mu}}_\infty) \models \phi$.

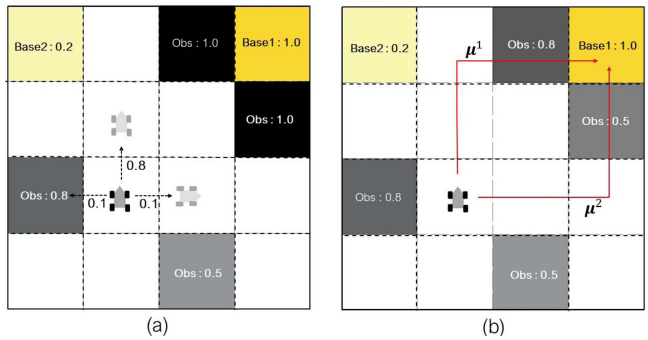


Fig. 1. Example of environments where the LTL task is $\phi_{\text{example}} = \square\lozenge\texttt{Base1} \wedge \square\lozenge\texttt{Base2} \wedge \neg\texttt{Obs}$, and Base 1 is surrounded by obstacles with different probabilities (infeasible). (a) Motion uncertainties and inaccessible Base 1. (b) Base 1 can be visited with different risks under two different policies.

Example 1: Fig. 1 considers the properties of interests $AP = \{\texttt{Base1}, \texttt{Base2}, \texttt{Obs}\}$ that label the environment and represent the regions of Base 1, Base 2, and obstacles, respectively. A robot is tasked to always eventually visit Base 1 and Base 2, while avoiding obstacles. The task can be expressed as an LTL formula $\phi_{\text{example}} = \square\lozenge\texttt{Base1} \wedge \square\lozenge\texttt{Base2} \wedge \neg\texttt{Obs}$. The labels of cells are assumed to be probabilistic, e.g., $\texttt{Obs} : 0.5$ indicates that the likelihood of a cell occupied by an obstacle is 0.5. To model the motion uncertainty, the robot is allowed to transit between adjacent cells or stay in a cell with a set of actions $\{\texttt{Up}, \texttt{Right}, \texttt{Down}, \texttt{Left}, \texttt{Stay}\}$, and the cost of each action is equal to 2. As shown in Fig. 1(a), it is assumed to successfully take the desired action with a probability of 0.8, and there is a probability of 0.2 to take other perpendicular actions following a uniform distribution. There are no motion uncertainties for the action of "Stay."

Fig. 1(a) represents an infeasible case, where Base 1 is surrounded by obstacles, and, thus, cannot be visited, while Base 2 is always accessible. Hence, it is desirable that the robot can revise its motion planning to mostly fulfill the given task (e.g., visit only Base 2 instead) whenever the task over an environment is found to be infeasible. Furthermore, it is essential to analyze the probabilistic violation of two different policies, as shown in Fig. 1(b), due to the environment uncertainties. The generated trajectories have different probabilities of colliding with obstacles and result in different violation costs.

As a result, to consider both feasible and infeasible tasks, a violation of task satisfaction can be defined as follows.

Definition 9: Given a PL-MDP $\mathcal{M}$ and an LTL task ϕ, the expected average violation cost per stage (AVPS) under the policy $\boldsymbol{\mu}$ is defined as

$$J_V(\mathcal{M}^{\boldsymbol{\mu}}, \phi) = \mathbb{E}^{\boldsymbol{\mu}}_{\mathcal{M}}\left[\liminf_{n\to\infty} \frac{1}{n} [i = 0] n \sum c_V(s_i, a_i, s_{i+1}, \phi)\right] \quad (4)$$

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 $\boldsymbol{\mu}(s_i)$.

Motivated by these challenges, the problem considered in this work is stated as follows.

Problem 1: Given an LTL task ϕ and a PL-MDP $\mathcal{M}$, the goal is to find an optimal finite memory policy $\boldsymbol{\mu}$ from the initial state and achieve the multiple objectives with a decreasing order of priority: 1) If ϕ is fully feasible, $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) \geq \gamma$, where $\gamma \in (0,1]$ is the desired satisfaction probability; 2) if ϕ is partially feasible, i.e., $\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) = 0$, minimizing AVPS $J_V(\mathcal{M}^{\boldsymbol{\mu}}, \phi)$ to satisfy ϕ as much as possible; 3) minimizing AEPS $J_E(\mathcal{M}^{\boldsymbol{\mu}})$ over the infinite horizons.

Due to the consideration of infeasible cases, by saying to satisfy ϕ as much as possible in Problem 1, we propose a relaxed structure and its expected average violation function to quantify how much the motion generated from a revised policy deviates from the desired task ϕ and minimize such a deviation. The concrete description of c_V is introduced in Section IV-B.

IV. Relaxed Product MDP Analysis

First, Section IV-A presents the construction of LDBA-based probabilistic product MDP. Then Section IV-B synthesizes how it can be relaxed to handle infeasible LTL constraints, and we concretely introduce the violation measurement of infeasible cases. Finally, the properties of the relaxed product MDP are discussed in Section IV-C, which can be utilized to generate the optimal policy.

A. LDBA-Based Probabilistic Product MDP

We first present the definition of LDBA-based probabilistic product MDP.

Definition 10: Given a PL-MDP $\mathcal{M}$ and an LDBA $\mathcal{A}_\phi$, the product MDP is defined as a tuple $\mathcal{P} = (X, U^{\mathcal{P}}, p^{\mathcal{P}}, x_0, \text{Acc}, c^{\mathcal{P}}_A)$, where $X = S \times 2^{AP} \times Q$ is the set of labeled states s.t. $X = \{(s,l,q) \big| s \in S, l \in L(s), q \in Q\}$; $U^{\mathcal{P}} = A \cup \{\epsilon\}$ is the set of actions, where the ϵ-transitions of LDBA are regarded as actions; $x_0 = (s_0, l_0, q_0)$ is the initial state; $\text{Acc} = \{(s,l,q) \in X \big| q \in F\}$ is the set of accepting states; the cost of taking an action $u^{\mathcal{P}} \in U^{\mathcal{P}}$ at $x = (s,l,q)$ is defined as $c^{\mathcal{P}}_A(x, u^{\mathcal{P}}) = c_A(s,a)$ if $u^{\mathcal{P}} = a \in A(s)$ and $c^{\mathcal{P}}_A(x, u^{\mathcal{P}}) = 0$ otherwise; the transition function $p^{\mathcal{P}} : X \times U^{\mathcal{P}} \times X \to [0,1]$ is defined as: For $x' = (s', l', q')$ in X, 1) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = p_L(s', l') \cdot p_S(s, a, s')$ if $\delta(q, l) = q'$ and $u^{\mathcal{P}} = a \in A(s)$, 2) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = 1$ if $u^{\mathcal{P}} \in \{\epsilon\}$, $q' \in \delta(q, \epsilon)$, and $(s', l') = (s, l)$, and 3) $p^{\mathcal{P}}(x, u^{\mathcal{P}}, x') = 0$ otherwise.

Let $\boldsymbol{\pi}_{\mathcal{P}}$ denote the policy over $\mathcal{P}$. The product MDP $\mathcal{P}$ captures the intersections between all feasible paths over $\mathcal{M}$ and all words accepted to $\mathcal{A}_\phi$, facilitating the identification of admissible motions that satisfy the task ϕ. The path $\boldsymbol{x}^{\boldsymbol{\pi}_{\mathcal{P}}}_{\infty} = x_0 \ldots x_i x_{i+1} \ldots$ under a policy $\boldsymbol{\pi}_{\mathcal{P}}$ is accepted if $\inf(\boldsymbol{x}^{\boldsymbol{\pi}_{\mathcal{P}}}_{\infty}) \cap \text{Acc} \neq \emptyset$. If a subproduct MDP $\mathcal{P}'_{(X', U')}$ is an MEC of $\mathcal{P}$ and $X' \cap \text{Acc} \neq \emptyset$, $\mathcal{P}'_{(X', U')}$ is called an AMEC of $\mathcal{P}$. Details of generating AMEC for a product MDP can be found in [1]. Note that, synthesizing the AMECs does not require finding a set of policies that restrict the selections of actions for each state.

Denote by $\Xi_{\text{acc}} = \{\Xi^i_{\text{acc}}, i = 1 \ldots n^{\mathcal{P}}_{\text{acc}}\}$ the set of all AMECs of $\mathcal{P}$, where $\Xi^i_{\text{acc}} = \mathcal{P}'_{(X'_i, U'_i)}$ with $X'_i \subseteq X$ and $U'_i \subseteq U^{\mathcal{P}}$ and $n^{\mathcal{P}}_{\text{acc}}$ is the number of AMECs in $\mathcal{P}$. Satisfying the LTL task ϕ is equivalent to finding a policy $\boldsymbol{\pi}_{\mathcal{P}}$ that drives the agent enter into one of an AMEC Ξ^i_{acc} in $\mathcal{P}$. Based on that, we can define the feasibility over product MDP.

Lemma 1: Given a product MDP $\mathcal{P}$ constructing from a PL-MDP $\mathcal{M}$ and $\mathcal{A}_\phi$, the LTL task is fully feasible if and only if there exists at least one AMEC in $\mathcal{P}$ [1].

As a result, if an LTL task is feasible with respect to the PL-MDP model, there exits at least one AEMC in corresponding to the product MDP, and satisfying the task ϕ is equivalent to reaching an AMEC in Ξ_{acc}. For the cases that AMECs do not exist in $\mathcal{P}$, most existing works [1], [16], [43], [44], and the work of [17] considered accepting SCCs (ASCC) to minimize the probability of entering bad system states. However, there is no guarantee that the agent will stay within an ASCC to yield satisfactory performance, especially when the probability of entering bad system states is large. Also, the existence of ASCC is based on the existence of an accepting path, returns no solution for the case of Fig. 1(b). Moreover, for the infeasible cases, the work [17] needs first to check the existence of AMECs and then formulate ASCCs, whereas generating of AMECs is computationally expensive. In contrast, this frame designs a relaxed product MDP in the following, which allows us to apply its AMECs addressing both feasible and infeasible cases.

B. Relaxed Probabilistic Product MDP

For the product MDP $\mathcal{P}$ in Definition 10, the satisfaction of ϕ is based on the assumption that there exists at least one AMEC in $\mathcal{P}$. However, such an assumption cannot always be true in practice. To address this challenge, the relaxed product MDP is designed to allow the agent to revise its motion plan whenever the desired LTL constraints cannot be strictly followed.

Definition 11: The relaxed product MDP is constructed from $\mathcal{P}$ as a tuple $\mathcal{R} = (X, U^{\mathcal{R}}, p^{\mathcal{R}}, x_0, \text{Acc}, c^{\mathcal{R}}_A, c^{\mathcal{R}}_V)$, where

1) X, x_0, and Acc are the same as in $\mathcal{P}$;
2) $U^{\mathcal{R}}$ is the set of extended actions that are extended to jointly consider the actions of $\mathcal{M}$ and the input alphabet of $\mathcal{A}_\phi$. Specifically, given a state $x = (s,l,q) \in X$, the available actions are $U^{\mathcal{R}}(x) = \{(a, \iota) \big| a \in A(s), \iota \in (2^{AP} \cup \{\epsilon\})\}$. Given an action $u^{\mathcal{R}} = (a, \iota) \in U^{\mathcal{R}}(x)$, the projections of $u^{\mathcal{R}}$ to $A(s)$ in $\mathcal{M}$ and to $2^{AP} \cup \{\epsilon\}$ in $\mathcal{A}_\phi$ are denoted by $u\big|^{\mathcal{R}}_{\mathcal{M}}$ and $u\big|^{\mathcal{R}}_{\mathcal{A}}$, respectively;
3) $p^{\mathcal{R}} : X \times U^{\mathcal{R}} \times X \to [0,1]$ is the transition function. The transition probability $p^{\mathcal{R}}$ from a state $x = (s,l,q)$ to a state $x' = (s', l', q')$ is defined as: 1) $p^{\mathcal{R}}(x, u^{\mathcal{R}}, x') = p_L(s', l') \cdot p_S(s, a, s')$ with $a = u\big|^{\mathcal{R}}_{\mathcal{M}}$, if q can be transited to q' and $u\big|^{\mathcal{R}}_{\mathcal{A}} \neq \epsilon$ and $\delta(q, u\big|^{\mathcal{R}}_{\mathcal{A}}) = q'$; 2) $p^{\mathcal{R}}(x, u^{\mathcal{R}}, x') = 1$, if $u\big|^{\mathcal{R}}_{\mathcal{A}} = \epsilon$, $q' \in \delta(q, \epsilon)$, and $(s', l') = (s, l)$; 3) $p^{\mathcal{R}}(x, u^{\mathcal{R}}, x') = 0$ otherwise. Under an action $u^{\mathcal{R}} \in U^{\mathcal{R}}(x)$, it holds that $\sum_{x' \in X} p^{\mathcal{R}}(x, u^{\mathcal{R}}, x') = 1$;
4) $c^{\mathcal{R}}_V : X \times U^{\mathcal{R}} \to \mathbb{R}$ is the execution cost. Given a state x and an action $u^{\mathcal{R}}$, the execution cost is defined as

$$c^{\mathcal{R}}_A(x, u^{\mathcal{R}}) = \begin{cases} c_A(s,a) & \text{if } u\big|^{\mathcal{R}}_{\mathcal{M}} \in A(s) \\ 0 & \text{otherwise} \end{cases}$$

5) $c^{\mathcal{R}}_V : X \times U^{\mathcal{R}} \times X \to \mathbb{R}$ is the violation cost. The violation cost of the transition from $x = (s,l,q)$ to $x' =$

$(s,' l,' q')$ under an action $u^{\mathcal{R}}$ is defined as

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

where $w_V(x,x') = \mathrm{Dist}(L(s), \mathcal{X}(q,q'))$ with $\mathcal{X}(q,q') = \{l \in 2^{\boldsymbol{\pi}} | q \xrightarrow{l} q'\}$ being the set of input alphabets that enables the transition from q to q'. Borrowed from [20], the function $\mathrm{Dist}(L(s), \mathcal{X}(q,q'))$ measures the distance from $L(s)$ to the set $\mathcal{X}(q,q')$.

Remark 2: Given a PL-MDP $\mathcal{M}$ and A_ϕ, the relaxed product MDP $\mathcal{R}$ holds the same state space as the corresponding product MDP $\mathcal{P}$. The main difference compared with $\mathcal{P}$ is that the $\mathcal{R}$ has a different action space with revised transition conditions so that $\mathcal{R}$ has a more connected structure. In addition, we propose the violation cost for each transition to measure the AVPS over the relaxed product model $\mathcal{R}$. The complexity analysis of applying the relaxed product MDP is discussed in Section V-E. Note that, the environment uncertainties influence the transition probabilities of a relaxed product MDP, and in turn, affect the probabilities of entering into AMECs.

The weighted violation function $w_V(x,x')$ quantifies how much the transition from x to x' in a product automaton violates the constraints imposed by ϕ. It holds that $c_V^{\mathcal{R}}(x,u^{\mathcal{R}},x') = 0$ if $p^{\mathcal{P}}(x,u^{\mathcal{P}},x') \neq 0$, since a nonzero $p^{\mathcal{P}}(x,u^{\mathcal{P}},x')$ indicates either $\delta(q,L(s)) = q'$ or $\delta(q,\epsilon) = q'$, leading to $w_V(x,x') = 0$. Let $\boldsymbol{\pi}$ denote the policy of $\mathcal{R}$. Consequently, we can transform the measurement of AEPS, and AVPS $J_V(\mathcal{M}^{\boldsymbol{\mu}}, \phi)$ from PL-MDP $\mathcal{M}$ into $\mathcal{R}$.

Definition 12: Given a relaxed product MDP $\mathcal{R}$ generated from a PL-MDP $\mathcal{M}$ and an LDBA A_ϕ, the AEPS of $\mathcal{R}$ under policy $\boldsymbol{\pi}$ can be defined as

$$J_E(\mathcal{R}^{\boldsymbol{\pi}}) = \mathbb{E}_{\mathcal{R}}^{\boldsymbol{\pi}}\left[\liminf_{n\to\infty} \frac{1}{n}\sum_{i=0}^{n} c_A^{\mathcal{R}}(x_i, u_i^{\mathcal{R}})\right]. \tag{5}$$

Similarly, the AVPS of $\mathcal{R}$ can be reformulated as

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

Hence, $J_V(\mathcal{R}^{\boldsymbol{\pi}})$ can be applied to measure how much ϕ is satisfied in Problem 1. It should be pointed out that the violation cost $c_V^{\mathcal{R}}$ jointly considers the probability of an event $p_L(s,' l')$ and the violation of the desired ϕ. For instance, Fig. 1(b) shows the trajectories generated from two different policies that traverse regions labeled Obs with different probabilities. It is obvious that the task of infinitely visiting Base1 and Base2 is infeasible. The paths induced from different policies hold different AVPSs for partial satisfaction. Consequently, a large cost $c_V^{\mathcal{R}}$ can occur if $p_L(s,' l')$ is close to 1 (e.g., an obstacle appears with high probability), or the violation w_V is large, or both are large. Hence, minimizing the AVPS will not only bias the planned path toward more fulfillment of ϕ by penalizing w_V but also toward more satisfaction of mission operation (e.g., reduce the risk of mission failures by avoiding areas with high probability obstacles). This idea is illustrated via simulations in Case 2 in Section VI.

C. Properties of Relaxed Product MDP

Given an LTL formula ϕ and a PL-MDP $\mathcal{M}$, this section verifies properties of the designed relaxed product MDP $\mathcal{R}$, which can be applied to solve feasible cases, where there exists at least one policy $\boldsymbol{\mu}$ such that $\Pr_{\mathcal{M}}^{\boldsymbol{\mu}}(\phi) > 0$, and infeasible cases, where $\Pr_{\mathcal{M}}^{\boldsymbol{\mu}}(\phi) = 0$ for any policy $\boldsymbol{\mu}$. Based on definition 11, the relaxed product MDP $\mathcal{R}$ and its corresponding product MDP $\mathcal{P}$ have the same states. Hence, we can regard $\mathcal{R}$ and $\mathcal{P}$ as two separate directed graphs. Let ABSCC denote the BSCC that contains at least one accepting state in $\mathcal{P}$ or $\mathcal{R}$.

Theorem 1: Given a PL-MDP $\mathcal{M}$ and an LDBA automaton $\mathcal{A}_\phi$ corresponding to the desired LTL task specification ϕ, the relaxed product MDP $\mathcal{R} = \mathcal{M} \otimes \mathcal{A}_\phi$ and corresponding product MDP $\mathcal{P}$ have the following properties.

1) The directed graph of traditional product $\mathcal{P}$ 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 $\mathcal{M}$, any direct graph of AMEC of $\mathcal{P}$ is the subgraph of a direct graph of AMEC of $\mathcal{R}$.

Proof. Property 1: by definition 10, there is a transition between $x = \langle s,l,q\rangle$ and $x' = \langle s,' l,' q'\rangle$ in $\mathcal{P}$, if and only if $p^{\mathcal{P}}(x,u^{\mathcal{P}},x') \neq 0$. There are two cases for $p^{\mathcal{P}}(x,u^{\mathcal{P}},x') \neq 0$: i) $\exists l \in L(s), \delta(q,l) = q'$ and $p_S(s,a,s') \neq 0$ with $u^{\mathcal{P}} = a$; and ii) $q' \in \delta(q,\epsilon)$ and $u^{\mathcal{P}} = \epsilon$. In the relaxed $\mathcal{R}$, for case i), there always exist $u\big|_{\mathcal{A}}^{\mathcal{R}} = L(s)$ and $u\big|_{\mathcal{M}}^{\mathcal{R}} = u^{\mathcal{P}} = a$ with $p_S(s,a,s') \neq 0$, such that $p^{\mathcal{R}}(x,u^{\mathcal{R}},x') \neq 0$. For case ii), based on the fact that $q' \in \delta(q,\epsilon)$, there always exists $u\big|_{\mathcal{A}}^{\mathcal{R}} = \epsilon$, such that $p^{\mathcal{R}}(x,u^{\mathcal{R}},x') \neq 0$. Therefore, any existing transition in $\mathcal{P}$ is also preserved in the corresponding relaxed product MDP $\mathcal{R}$.

Property 2: As indicated in [34], for an LDBA $\mathcal{A}_\phi$, there always exists a BSCC that contains at least one of the accepting states. Without loss of generality, let $Q_B \subseteq Q$ be a BSCC of $\mathcal{A}_\phi$ s.t. $Q_B \cap F \neq \emptyset$. Denote by $\mathcal{M}_{(S,'A')}$ an EC of $\mathcal{M}$. By the definition of the relaxed product MDP $\mathcal{R} = \mathcal{M} \otimes \mathcal{A}_\phi$, we can construct a subproduct MDP $\mathcal{R}_{(X_B,U_B^{\mathcal{R}})}$ such that $x = \langle s,l,q\rangle \in X_B$ with $s \in S'$ and $q \in Q_B$. For each $u_B^{\mathcal{R}}(x) \in U_B^{\mathcal{R}}$, we restrict $u_B^{\mathcal{R}}(x) = (A(s), l_B)$ with $A(s) \in A'$ and $\delta(q,l_B) \in Q_B$. As a result, we can obtain that an EC $\mathcal{R}_{(X_B,U_B^{\mathcal{R}})}$ that contains at least one of the accepting states due to the fact, i.e., $Q_B \cap F \neq \emptyset$. Therefore, there exists at least an AMEC in the relaxed $\mathcal{R}$.

Property 3: If ϕ is feasible over $\mathcal{M}$, there exist AMECs in both $\mathcal{P}$ and $\mathcal{R}$. Let $\Xi_{\mathcal{P}}$ and $\Xi_{\mathcal{R}}$ be an AMEC of $\mathcal{P}$ and $\mathcal{R}$, respectively. From graph perspectives, $\Xi_{\mathcal{P}}$ and $\Xi_{\mathcal{R}}$ can be considered as BSCCs $\mathcal{G}_{(\Xi_{\mathcal{P}})} \subseteq \mathcal{G}_{(X,U^{\mathcal{P}})}$ and $\mathcal{G}_{(\Xi_{\mathcal{R}})} \subseteq \mathcal{G}_{(X,U^{\mathcal{R}})}$ containing accepting states, respectively. According to Property 1, it can be concluded that for any $\mathcal{G}_{(\Xi_{\mathcal{P}})}$, we can find a $\mathcal{G}_{(\Xi_{\mathcal{R}})}$ s.t. $\mathcal{G}_{(\Xi_{\mathcal{P}})}$ is a subgraph of $\mathcal{G}_{(\Xi_{\mathcal{R}})}$ □

Theorem 1 indicates that the directed graph of $\mathcal{R}$ is more connected than the directed graph of the corresponding $\mathcal{P}$. Therefore, there always exists at least one AMEC in $\mathcal{R}$ even for the infeasible cases, which allows us to measure the violation with respect to the original LTL formula. Moreover, if a given task ϕ is fully feasible in $\mathcal{P}$ (there exists a policy $\boldsymbol{\pi}_{\mathcal{P}}$, such that its induced path $\boldsymbol{x}_{\boldsymbol{\pi}^{\mathcal{P}}}$ over $\mathcal{P}$ satisfying ϕ, i.e.,

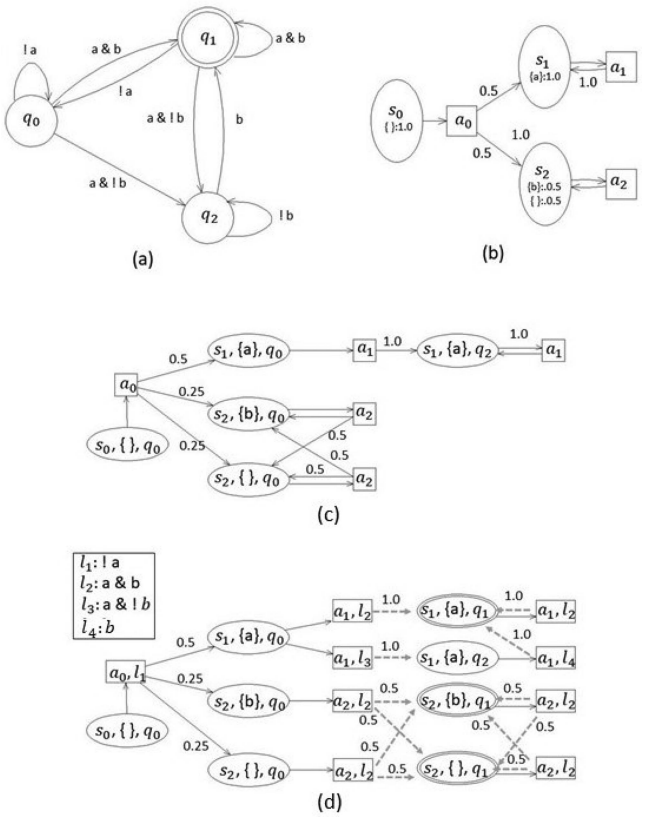


Fig. 2. (a) LDBA $\mathcal{A}_\phi$. (b) MDP $\mathcal{M}$. (c) Constrained product MDP. (d) Relaxed product MDP.

$\Pr^{\boldsymbol{\mu}}_{\mathcal{M}}(\phi) > 0$). Also, there must exist a policy $\boldsymbol{\pi}$. s.t. its induced path $\boldsymbol{x}^{\boldsymbol{\pi}}$ over $\mathcal{R}$ is free of violation cost. In other words, $\mathcal{R}$ can also handle feasible tasks by identifying accepting paths with zero AVPS.

Example 2: To illustrate Theorem 1, a running example is shown here. Consider an LDBA $\mathcal{A}_\phi$ corresponding to $\phi = \square\diamondsuit \mathtt{a} \wedge \square\diamondsuit \mathtt{b}$ and an MDP $\mathcal{M}$, as shown in Fig. 2(a) and (b), respectively. For ease of presentation, partial structures of the product MDPs $\mathcal{P} = \mathcal{M} \otimes \mathcal{A}_\phi$ and $\mathcal{R} = \mathcal{M} \otimes \mathcal{A}_\phi$ are shown in Fig. 2(c) and (d), respectively. Since the LTL formula ϕ is infeasible over $\mathcal{M}$, there is no AMEC in $\mathcal{P}$, whereas there exists one in $\mathcal{R}$. Note that, there is no ϵ-transitions in this case.

Given an accepting path $\boldsymbol{x}^{\boldsymbol{\pi}}_\infty = x_0 \ldots x_i x_{i+1} \ldots$, we propose to regulate the multiobjective optimization objective consisting of implementation cost and violation cost for each transition as

$$c^{\mathcal{R}}\left(x_i, u_i^{\mathcal{R}}, x_{i+1}\right) = c_A^{\mathcal{R}}\left(x_i, u_i^{\mathcal{R}}\right) \cdot \max\left\{e^{\beta c_V^{\mathcal{R}}(x_i, u_i^{\mathcal{R}}, x_{i+1})}, 1\right\} \tag{7}$$

where $\beta \in \mathbb{R}^+$ indicates the relative importance. Based on (7), the expected average regulation cost per stage (ARPS) of $\mathcal{R}$ under a policy $\boldsymbol{\pi}$ is formulated as

$$J(\mathcal{R}^{\boldsymbol{\pi}}) = \mathbb{E}^{\boldsymbol{\pi}}_{\mathcal{R}}\left[\liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} c^{\mathcal{R}}\left(x_i, u_i^{\mathcal{R}}, x_{i+1}\right)\right]. \tag{8}$$

In this work, we aim at generating the optimal policy $\boldsymbol{\pi}$ that minimizes the ARPS $J(\mathcal{R}^{\boldsymbol{\pi}})$, while satisfying the acceptance condition of $\mathcal{R}$.

Lemma 2: By selecting a large parameter $\beta >> 1$ of (7) the first priority of minimizing the AVPS $J_V(\mathcal{R}^{\boldsymbol{\pi}})$ in ARPS $J(\mathcal{R}^{\boldsymbol{\pi}})$ is guaranteed such that minimizing the AEPS with weighting β will never come at the expense of minimizing AVPS, i.e., $J_V(\mathcal{R}^{\boldsymbol{\pi}}) \geq J_V(\mathcal{R}^{\boldsymbol{\pi}'}) \Longrightarrow J(\mathcal{R}^{\boldsymbol{\pi}}) \geq J(\mathcal{R}^{\boldsymbol{\pi}'})$.

Lemma 2 can be directly verified based on formulating the exponential function in (7).

Problem 2: Given an $\mathcal{R}$ from $\mathcal{M}$ and A_ϕ, Problem 1 can be formulated as

$$\begin{aligned} &\min_{\boldsymbol{\pi}\in\bar{\boldsymbol{\pi}}} J(\mathcal{R}^{\boldsymbol{\pi}}) \\ &\text{s.t.} \Pr^{\boldsymbol{\pi}}_{\mathcal{M}}\left(\square\diamondsuit \text{Acc}\right) \geq \gamma \end{aligned} \tag{9}$$

where $\beta >> 1$, $\bar{\boldsymbol{\pi}}$ represents the set of admissible policies over $\mathcal{R}$, $\Pr^{\boldsymbol{\pi}}_{\mathcal{M}}(\square\diamondsuit \text{Acc})$ is the probability of visiting the accepting states of $\mathcal{R}$ infinitely often, and γ is the desired threshold for the probability of task satisfaction.

Remark 3: When the LTL task with respect to the PL-MDP is infeasible, the threshold γ represents the probability of entering into any of an AMEC in $\mathcal{R}$. Furthermore, for the cases where there exist no policies satisfying $\Pr^{\boldsymbol{\pi}}_{\mathcal{M}}(\square\diamondsuit \text{Acc}) \geq \gamma$ for a given γ, the above optimization Problem 2 is infeasible, and returns no solutions. However, we can regard γ as a slack variable, and technical details are explained in Remark 4.

V. Solution

The prefix–suffix structure of LTL satisfaction over an infinite horizon is inspired by the following Lemma.

Lemma 3: Given any Markov chain $MC^{\boldsymbol{\pi}}_{\mathcal{P}}$ under policy $\boldsymbol{\pi}$, its states can be represented by a disjoint union of a transient class $\mathcal{T}_{\boldsymbol{\pi}}$ and n_R closed irreducible recurrent classes $\mathcal{R}^j_{\boldsymbol{\pi}}$, $j \in \{1, \ldots, n_R\}$ [45].

Given any policy, Lemma 3 indicates that the behaviors before entering into AMECs involve the transient class, and a recurrent class represents the decision-making within an AMEC. Note that, Lemma 3 provides a general form of state partition that can be applied to any MDP model. This section shows how to integrate the state partition with a relaxed product MDP. Especially, we analyze states partition to divide Problem 2 into two parts and focus on synthesizing the optimal prefix and suffix policies via LP, which addresses the tradeoff between minimizing the ARPS (Section V-C) over a long term and reaching the probability threshold of task satisfaction.

This solution framework mainly focuses on adopting the ideas of prefix–suffix plans and the method of MDP optimization for relaxed product structures. The details about the intuition, i.e., the analysis of policies over an infinite horizons, computation of AMECs, and linear programming, can be found in [1], [16], and [18].

A. State Partition

According to Property 1 of Theorem 1, let $\Xi^{\mathcal{R}}_i = (X_i, U^{\mathcal{R}}_i)$ denote an AMEC of $\mathcal{R}$ and let $\Xi^{\mathcal{R}}_{\text{acc}} = \{\Xi^{\mathcal{R}}_1, \ldots, \Xi^{\mathcal{R}}_N\}$ denote the set of AMECs. To facilitate the analysis, the state X of $\mathcal{R}$ is divided into a transient class X_T and a recurrent class X_R, where $X_R = \cup_{(X_i, U^{\mathcal{R}}_i) \subseteq \Xi^{\mathcal{R}}_{\text{acc}}} X_i$ is the union of the AMEC

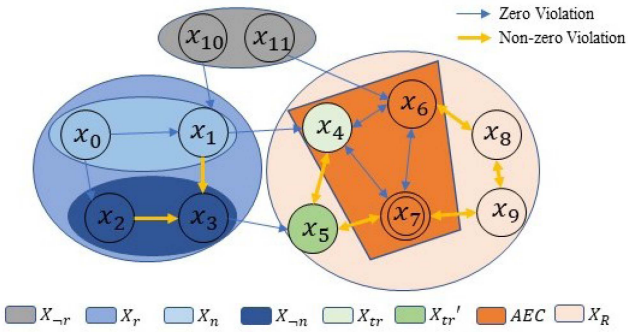


Fig. 3. Illustration of the partition of X in $\mathcal{R}$, where x_7 is an accepting state. The edges with and without violation cost are marked.

states of $\mathcal{R}$ and $X_T = X \setminus X_R$. Let $X_r \subseteq X_T$ and $X_{\neg r} \subseteq X_T$ denote the set of states that can and cannot be reached from the initial state x_0, respectively. Since the states in $X_{\neg r}$ cannot be reached from x_0 (i.e., bad states), we will only focus on X_r, which can be further divided into X_n and $X_{\neg n}$ based on the violation conditions. Let $X_{\neg n}$ and X_n be the set of states that can reach X_R with and without violation edges, respectively. Based on X_n, $X_{\neg n}$ and X_R, let $X_{tr}, X'_{tr} \subseteq X_R$ denote the sets of states that can be reached within one transition from X_n and $X_{\neg n}$, respectively. An example is provided in Fig. 3 to illustrate the partition of states.

B. Plan Prefix

The objective of plan prefix is to construct an optimal policy that drives the agent from x_0 to $X_{tr} \cup X'_{tr}$, while minimizing the combined average cost. To achieve this goal, we construct the prefix MDP model of $\mathcal{R}$ to analyze the prefix behaviors under any policy.

Definition 13: A prefix MDP of $\mathcal{R}$ can be defined as $\mathcal{R}_{\text{pre}} = \{X_{\text{pre}}, U^{\mathcal{R}}_{\text{pre}}, p^{\mathcal{R}}_{\text{pre}}, x_0, c^{\mathcal{R}}_{A_{\text{pre}}}, c^{\mathcal{R}}_{V_{\text{pre}}}\}$ of $\mathcal{R}$, where

1) $X_{\text{pre}} = X_r \cup X_{tr} \cup X'_{tr} \cup v$, where v is a trap state that models the behaviors within the union of AMECs;
2) the set of actions is $U^{\mathcal{R}}_{\text{pre}} = U^{\mathcal{R}} \cup \tau$, where τ represents a self-loop action only enabled at state v s.t. $\tau = U_{\text{pre}}(v)$, and $U^{\mathcal{R}}_{\text{pre}}(x)$ is the actions enabled at $x \in X_{\text{pre}}$;
3) the transition probability $p^{\mathcal{R}}_{\text{pre}}$, is defined as: (i) $p^{\mathcal{R}}_{\text{pre}}(x, u^{\mathcal{R}}, \bar{x}) = p^{\mathcal{R}}(x, u^{\mathcal{R}}, \bar{x}) \forall x \in X_r, \quad \bar{x} \in X_{\text{pre}} \setminus v$, and $\forall u^{\mathcal{R}} \in U^{\mathcal{R}}(x)$; (ii) $p^{\mathcal{R}}_{\text{pre}}(x, u^{\mathcal{R}}, v) = 1$ $\forall x \in X_{tr} \cup X'_{tr}, \quad u^{\mathcal{R}} \in U^{\mathcal{R}}(x)$ and $v \in \mathcal{V}$; (iii) $P^{\mathcal{R}}_{\text{pre}}(v, \tau, v) = 1$;
4) the implementation cost is defined as: (i) $c^{\mathcal{R}}_{A_{\text{pre}}}(x, u^{\mathcal{R}}) = c^{\mathcal{R}}_{A}(x, u^{\mathcal{R}}) \forall x \in X_r$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}(x)$; and (ii) $c^{\mathcal{R}}_{A_{\text{pre}}}(x, u^{\mathcal{R}}) = c^{\mathcal{R}}_{A_{\text{pre}}}(v, \tau) = 0, \forall x \in X_{tr} \cup X'_{tr}$;
5) the violation cost is defined as: $c^{\mathcal{R}}_{V_{\text{pre}}}(x, u^{\mathcal{R}}, \bar{x}) = c^{\mathcal{R}}_{V}(x, u^{\mathcal{R}}, \bar{x}) \forall x \in X_r, \quad \bar{x} \in X_r \cup X_{tr}, \quad u^{\mathcal{R}} \in U^{\mathcal{R}}(x)$; $c^{\mathcal{R}}_{V_{\text{pre}}}(x, u^{\mathcal{R}}, \bar{x}) = 0$ otherwise.

In Definition 13, v is the trap state s.t. there is only self-loop action enabled at the state. The agent's state remains the same once it enters the trap state. Therefore, the optimization process of desired policy over prefix product MDP $\mathcal{R}_{\text{pre}}$ can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\pi} \in \bar{\boldsymbol{\pi}}_{\text{pre}}} \ & \mathbb{E}^{\boldsymbol{\pi}}_{\mathcal{R}_{\text{pre}}} \left[\limsup_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} c^{\mathcal{R}}_{\text{pre}}\left(x_i, u^{\mathcal{R}}_i, x_{i+1}\right) \right], \\ \text{s.t.} \ & \Pr^{\boldsymbol{\pi}}_{x_0, \mathcal{R}_{pre}}(\lozenge v) \geq \gamma \end{aligned} \tag{10}$$

where $\bar{\boldsymbol{\pi}}_{\text{pre}}$ represents a set of all admissible policies over $\mathcal{R}_{\text{pre}}$, $\Pr^{\boldsymbol{\pi}}_{x_0, \mathcal{R}_{pre}}(\lozenge v)$ denotes the probability of $\boldsymbol{x}_{\text{pre}}$ starting from x_0 and eventually reaching the trap state v, and $c^{\mathcal{R}}_{\text{pre}}(x_i, u^{\mathcal{R}}_i, x_{i+1})$ is the regulation cost for each transition, such that

$$\begin{aligned} c^{\mathcal{R}}_{\text{pre}}(x_i, u^{\mathcal{R}}_i, x_{i+1}) &= c^{\mathcal{R}}_{A_{\text{pre}}}(x_i, u^{\mathcal{R}}_i) \\ &\cdot \max\left\{ e^{\beta \cdot c^{\mathcal{R}}_{V_{\text{pre}}}(x_i, u^{\mathcal{R}}_i, x_{i+1})}, 1 \right\}. \end{aligned} \tag{11}$$

In the prefix plan, the policies of staying within AMECs can be modeled by adding the trap state v. Based on the station partition in Section V-A, reaching an AMEC of $\mathcal{R}$ is equivalent to reaching the set $X_{tr} \cup X'_{tr}$. Furthermore, since there exist policies under which paths starting from X_r to $X_{tr} \cup X'_{tr}$ only traverse the transitions with zero violation cost and the cost of staying at v is zero, a large β in $c^{\mathcal{R}}_{\text{pre}}$ is employed to search policies minimizing the AVPS over $\mathcal{R}_{\text{pre}}$ as the first priority. It should be noted that there always exists at least one solution $\boldsymbol{\pi}$ in (10). This is because AMECs in $\mathcal{R}$ always exist by Theorem 1, and we can always obtain a valid prefix MDP $\mathcal{R}_{\text{pre}}$ of $\mathcal{R}$.

Inspired by the network flow approaches [18], [19], (10) can be reformulated as a graph-constrained optimization problem and solved through LP. Especially, let $y_{x,u}$ denote the expected number of times over the infinite horizons such that x is visited with $u \in U^{\mathcal{R}}_{\text{pre}}$. It measures the state occupancy among all paths starting from the initial state x_0 under policy $\boldsymbol{\pi}$ in $\mathcal{R}_{\text{pre}}$, i.e., $y_{x,u} = \sum_{i=0}^{\infty} \Pr^{\boldsymbol{\pi}}_{x_0, \mathcal{R}_{pre}}(x_i = x, u^{\mathcal{R}}_i = u)$. Then, we can solve (10) as the following LP:

$$\begin{aligned} \min_{\{y_{x,u}\}} \ & \left[J_{\text{pre}} \triangleq \sum_{(x,u)} \sum_{\bar{x} \in X_{\text{pre}}} y_{x,u} \cdot p^{\mathcal{R}}_{\text{pre}}(x, u, \bar{x}) \cdot c^{\mathcal{R}}_{\text{pre}}(x, u, \bar{x}) \right] \\ & \text{s.t.} \sum_{(x,u)} \sum_{\bar{x} \in (X_{tr} \cup X'_{tr})} y_{x,u} \cdot p^{\mathcal{R}}_{\text{pre}}(x, u, \bar{x}) \geq \gamma \\ & \sum_{u' \in U^{\mathcal{R}}_{\text{pre}}(x')} y_{x,'u'} = \sum_{(x,u)} y_{x,u} \cdot p^{\mathcal{R}}_{\text{pre}}(x, u, x') + \chi_0(x') \\ & y_{x,u} \geq 0 \forall x' \in X_r \end{aligned} \tag{12}$$

where χ_0 is the distribution of initial state, and $\sum_{(x,u)} := \sum_{x \in (X_r \cup X_{tr} \cup X'_{tr})} \sum_{u \in U^{\mathcal{R}}_{\text{pre}}(x)}$.

Once the solution $y^*_{x,u}$ to (12) is obtained, the optimal stochastic policy $\boldsymbol{\pi}^*_{\text{pre}}$ can be generated as

$$\boldsymbol{\pi}^*_{\text{pre}}(x, u) = \begin{cases} \frac{y^*_{x,u}}{\sum_{\bar{u} \in U^{\mathcal{R}}_{\text{pre}}(x)} y^*_{x,\bar{u}}} & \text{if } x \in X^*_r \\ \frac{1}{\left| U^{\mathcal{R}}_{\text{pre}}(x) \right|} & \text{if } x \in X_{\text{pre}} \setminus X^*_r \end{cases} \tag{13}$$

where $X^*_r = \left\{ x \in X_r \middle| \sum_{u \in U^{\mathcal{R}}_{\text{pre}}(x)} y^*_{x,u} > 0 \right\}$.

Lemma 4: The optimal policy $\boldsymbol{\pi}^*_{\text{pre}}$ in (13) ensures that $\Pr^{\boldsymbol{\pi}}_{x_0, \mathcal{R}_{pre}}(\lozenge v) \geq \gamma$.

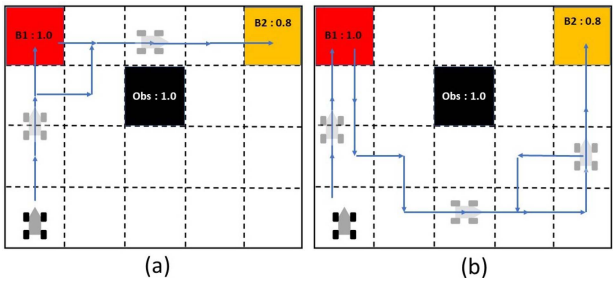


Fig. 4. Simulated trajectories with different γ under optimal prefix policies. (a) $\gamma = 0.3$. (b) $\gamma = 1.0$.

Proof: The proof is similar to [46, Lemma 3.3]. Due to the transient class of X_r, $y_{x,u}$ is finite. In the first constraint of (12), the sum $\sum_{(x,u)\bar{x}\in(X_{tr}\cup X'_{tr})} y_{x,u}\cdot p^{\mathcal{R}}_{\text{pre}}(x,u,\bar{x})$ is the expected number of times that $X_{tr}\cup X'_{tr}$ can be reached for the first time from a given initial state under the policy $\boldsymbol{\pi}^*_{\text{pre}}$. Since the agent remains in v once it enters $X_{tr}\cup X'_{tr}$, the sum is the probability of reaching X_R, which is lower bounded by γ. The second constraint of (12) guarantees the balances of network flow for the distribution of initial states. □

Example 3: As a running example in Fig. 4 , we illustrate the importance of the threshold γ that balances the tradeoff between optimizing the ARPS and reaching the probability of satisfaction. The motion uncertainties and the action cost are set the same as in Example 1. An LTL task is considered as $\phi_{\text{pre}} = \Diamond(\texttt{B1}\wedge\Diamond\texttt{B2})$ that requires visiting region B1 first and then B2 sequentially. ϕ_{pre} is feasible with respect to the corresponding PL-MDP (B2 is surrounded by probabilistic obstacles). Fig. 4 shows the results with two different γ under generated prefix optimal policies. It can be observed how such a parameter impacts the optimization bias since γ represents the quantitative probabilistic satisfaction [1].

Remark 4: The predefined threshold may influence the feasibility of the optimization (12) when there exist no policies s.t. $\Pr^{\boldsymbol{\pi}}_{x_0,\mathcal{R}_{pre}}(\Diamond v)\geq\gamma$. Since LP is a linear convex optimization [18], [19], to alleviate the issue, we can treat γ as a slack variable, such that (12) can be reformulated as

$$\begin{aligned}\min_{\boldsymbol{\pi}\in\bar{\boldsymbol{\pi}}_{\text{pre}}}\ & \mathbb{E}^{\boldsymbol{\pi}}_{\mathcal{R}_{\text{pre}}}\left[\limsup_{n\to\infty}\tfrac{1}{n}\sum_{i=o}^{n}c^{\mathcal{R}}_{\text{pre}}\left(x_i,u^{\mathcal{R}}_i,x_{i+1}\right)\right]+w_\gamma\cdot\gamma\\ \text{s.t.}\ & \Pr^{\boldsymbol{\pi}}_{x_0,\mathcal{R}_{pre}}(\Diamond v)\geq\gamma\end{aligned}\tag{14}$$

where w_γ is a regulation parameter that can be designed based on the users' preference for the multiobjective problems. Then, directly adopting the optimization process as the same as (12) and (13) can find feasible solutions.

Because the optimization of (14) increases the computational complexity, it is only applied when the formulation (12) returns no solution.

C. Plan Suffix

Suppose the prefix optimal policy $\boldsymbol{\pi}^*_{\text{pre}}$ drives the RL-agent into one AMEC $\Xi^{\mathcal{R}}_j$ of $\Xi^{\mathcal{R}}_{\text{acc}}$. This section considers the long-term behavior of the agent inside AMEC $\Xi^{\mathcal{R}}_j$. Since the agent can enter any AMEC, let $\Xi^{\mathcal{R}}_j=(X_j,U^{\mathcal{R}}_j)\subseteq\Xi^{\mathcal{R}}_{\text{acc}}$ denote such an AMEC and $X^{tr}_j=X_j\cap(X_{tr}\cup X'_{tr})$ denote the set of states that can be reached from plan prefix $\boldsymbol{x}_{\text{pre}}$. As a result, X^{tr}_j can be treated as an initial state for plan suffix after entering the AMEC. The objective of suffix policies is to enforce the accepting conditions and consider the optimization of long-term behavior. Therefore, after the agent entering into AMEC $\Xi^{\mathcal{R}}_j$, the optimization process of the desired policy over $\Xi^{\mathcal{R}}_j$ can be formulated

$$\begin{aligned}\min_{\boldsymbol{\pi}\in\bar{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}}\ & \mathbb{E}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}\left[\limsup_{n\to\infty}\tfrac{1}{n}\sum_{i=0}^{n}c^{\mathcal{R}}\left(x_i,u^{\mathcal{R}}_i,x_{i+1}\right)\right]\\ \text{s.t.}\ & \inf(\boldsymbol{x}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j})\cap\text{Acc}\neq\emptyset,\forall\boldsymbol{x}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}\in\boldsymbol{X}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}\end{aligned}\tag{15}$$

where $\bar{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}$ represents a set of all admissible policies over $\Xi^{\mathcal{R}}_j$, $\boldsymbol{X}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}$ is the set of all paths over the finite horizons under the policy $\boldsymbol{\pi}$, and $c^{\mathcal{R}}$ is the regulation transition cost in (7).

Let A_j denote a set of accepting states in X_j of $\Xi^{\mathcal{R}}_j$, i.e., $A_j = X_j\cap\text{Acc}$. Consequently, the acceptance condition of $\mathcal{R}$ can be satisfied s.t. $\inf(\boldsymbol{x}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j})\cap\text{Acc}\neq\emptyset$. One infinite accepting path $\boldsymbol{x}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}$ can be regarded as a concatenation of an infinite number of cyclic paths starting and ending in the set A_j.

Definition 14: A cyclic path $\boldsymbol{x}_c=x_1\ldots x_{N_{x_c}}$ associated with $\Xi^{\mathcal{R}}_j$ is a finite path with horizons N_{x_c} starting and ending at any subset $X'_j\subseteq X_j$, i.e., $x_1,x_{N_{x_c}}\in X'_j$, while actions are restricted to $U^{\mathcal{R}}_j$ to remain within X_j. A cyclic path $\boldsymbol{x}_c$ is called an accepting cyclic path if it starts and ends at A_j, i.e., $x_1,x_{N_{x_c}}\in A_j$.

By definition 14, the optimization problem (15) over the infinite horizons can be reformulated as

$$\begin{aligned}\min_{\boldsymbol{\pi}\in\bar{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j}}\ & \mathbb{E}_{\boldsymbol{x}_c\in\boldsymbol{X}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j,cycle}}\left[\tfrac{1}{N_{x_c}}\sum_{i=0}N_{x_c}\left(c^{\mathcal{R}}\left(x_i,u^{\mathcal{R}}_i,x_{i+1}\right)\right)\right]\\ \text{s.t.}\ & x_1,x_{N_{x_c}}\in A_j,\forall\boldsymbol{x}_c\in\boldsymbol{X}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j,cycle}\end{aligned}\tag{16}$$

where $\boldsymbol{X}^{\boldsymbol{\pi}}_{\Xi^{\mathcal{R}}_j,\text{cycle}}$ is a set of all cyclic paths under policy $\boldsymbol{\pi}$ over the AEMC $\Xi^{\mathcal{R}}_j$, the mean cyclic cost $\frac{1}{\bar{N}}\sum_{i=t}^{t+\bar{N}}(c^{\mathcal{R}}(x_i,u^{\mathcal{R}}_i,x_{i+1}))$ corresponds to the average cost per stage, and the constraint requires all induced cyclic paths under policy π are accepting cyclic paths.

Similarly, inspired from [43] and [44], we can also construct the suffix MDP model of $\mathcal{R}$ based on the state partition. Then (16) can be solved through LP. In order to apply the network flow algorithm to constrain paths starting from and ending at A_j, we need to transform accepting cyclic paths into the form of acyclic paths. To do so, we split A_j to create a virtual copy A^{out}_j that has no incoming transitions from A_j and a virtual copy A^{in}_j that only has incoming transitions from A_j, which allows representing a cyclic path as an acyclic path starting from A^{out}_j and ending in A^{in}_j. To convert the analysis of cyclic paths into acyclic paths, we construct the following suffix MDP of $\mathcal{R}$ for $\Xi^{\mathcal{R}}_j$.

Definition 15: A suffix MDP of $\mathcal{R}$ can be defined as $\mathcal{R}_{\text{suf}}=\{X_{\text{suf}},U^{\mathcal{R}}_{\text{suf}},p^{\mathcal{R}}_{\text{suf}},D^{\mathcal{R}}_{tr},c^{\mathcal{R}}_{A_{\text{suf}}},c^{\mathcal{R}}_{V_{\text{suf}}}\}$ where

1) $X_{\text{suf}}=(X_j\backslash A_j)\cup A^{out}_j\cup A^{in}_j$;

2) $U^{\mathcal{R}}_{\text{suf}} = U^{\mathcal{R}}_j \cup \tau$ with $U^{\mathcal{R}}_{\text{suf}}(x) = \tau \forall x \in A^{in}_j$, where $U^{\mathcal{R}}_{\text{suf}}(x)$ is the actions enabled at $x \in X_{\text{suf}}$;
3) the transition probability $p^{\mathcal{R}}_{\text{suf}}$ can be defined as follows: (i) $p^{\mathcal{R}}_{\text{suf}}(x, u^{\mathcal{R}}, \overline{x}) = p^{\mathcal{R}}(x, u^{\mathcal{R}}, \overline{x}) \forall x, \overline{x} \in (X_j \backslash A_j) \cup A^{\text{out}}_j$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}_j$; (ii) $p^{\mathcal{R}}_{\text{suf}}(x, u^{\mathcal{R}}, \overline{x}) = p^{\mathcal{R}}(x, u^{\mathcal{R}}, \overline{x}) \forall x \in (X_j \backslash A_j) \cup A^{\text{out}}_j$, $\overline{x} \in A^{in}_j$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}_j$; (iii) $p^{\mathcal{R}}_{\text{suf}}(x, \tau, \overline{x}) = 1 \; \forall x, \overline{x} \in A^{in}_j$;
4) the implementation cost is defined as: (i) $c^{\mathcal{R}}_{A_{\text{suf}}}(x, u^{\mathcal{R}}) = c^{\mathcal{R}}_A(x, u^{\mathcal{R}}) \forall x \in X_j \backslash A_j \cup A^{\text{out}}_j$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}_j$; and (ii) $c^{\mathcal{R}}_{A_{\text{suf}}}(x, \tau) = 0 \; \forall x \in A^{in}_j$.
5) the violation cost is defined as: (i) $c^{\mathcal{R}}_{V_{\text{suf}}}(x, u^{\mathcal{R}}, \overline{x}) = c^{\mathcal{R}}_V(x, u^{\mathcal{R}}, \overline{x}) \forall x, \overline{x} \in (X_j \backslash A_j) \cup A^{\text{out}}_j$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}_j$; (ii) $c^{\mathcal{R}}_{V_{\text{suf}}}(x, u^{\mathcal{R}}, \overline{x}) = c^{\mathcal{R}}_V(x, u^{\mathcal{R}}, \overline{x}) \forall x \in (X_j \backslash A_j) \cup A^{\text{out}}_j$, $\overline{x} \in A^{in}_j$ and $u^{\mathcal{R}} \in U^{\mathcal{R}}_j$; and (iii) $c^{\mathcal{R}}_{V_{\text{suf}}}(x, \tau, \overline{x}) = 0 \forall x, \overline{x} \in A^{in}_j$;
6) the distribution of the initial state $D^{\mathcal{R}}_{tr} : X_{\text{suf}} \to \mathbb{R}$ is defined as (i) $D^{\mathcal{R}}_{tr}(x) = \sum_{\hat{x} \in X_n} \sum_{u^{\mathcal{R}} \in U^{\mathcal{R}}} y^*_{\text{pre}}(\hat{x}, u^{\mathcal{R}}) \cdot P^{\mathcal{R}}(\hat{x}, u^{\mathcal{R}}, x)$, if $x \in X^{tr}_j$; (ii) $D^{\mathcal{R}}_{tr}(x) = 0$ if $x \in X_{\text{suf}} \backslash X^{tr}_j$, where X_n is a set of states that can reach X_R in transient class.

Let $z_{x,u} = z(x, u)$ denote the long-term frequency that the state is at $x \in X_{\text{suf}} \setminus A^{in}_j$ and the action u is taken. Then, to solve (16), the following LP is formulated as:

$$\begin{aligned} &\min_{\{z_{x,u}\}} \left[J_{\Xi^{\mathcal{R}}_j} \triangleq \sum_{(x,u)} \sum_{\overline{x} \in X_{\text{suf}}} z_{x,u} p^{\mathcal{R}}_{\text{suf}}(x, u, \bar{x}) \, c^{\mathcal{R}}_{\text{suf}}(x, u, \bar{x}) \right] \\ &\text{s.t.} \sum_{u' \in U^{\mathcal{R}}_{\text{suf}}(x')} z_{x',u'} = \sum_{(x,u)} z_{x,u} \cdot p^{\mathcal{R}}_{\text{suf}}(x, u, x') + D^{\mathcal{R}}_{tr}(x') \\ &\sum_{(x,u)} \sum_{\overline{x} \in A^{in}_j} z_{x,u} \cdot p^{\mathcal{R}}_{\text{suf}}(x, u, \bar{x}) = \sum_{x \in X'_{\text{suf}}} D^{\mathcal{R}}_{tr}(x), \\ &z_{x,u} \geq 0 \forall x' \in X'_{\text{suf}} \end{aligned} \tag{17}$$

where $X'_{\text{suf}} = X_{\text{suf}} \setminus A^{in}_j$, $\sum_{(x,u)} := \sum_{x \in X'_{\text{suf}}} \sum_{u \in U^{\mathcal{R}}_{\text{suf}}(x)}$, and $c^{\mathcal{R}}_{\text{suf}}(x, u, \bar{x}) = c^{\mathcal{R}}_{A_{\text{suf}}}(x, u) \cdot \max\{e^{\beta \cdot c^{\mathcal{R}}_{V_{\text{suf}}}(x,u,\bar{x})}, 1\}$. The first constraint represents the in–out flow balance, and the second constraint ensures that A^{in}_j is eventually reached. Note that, (15) and (16) are defined for the suffix MDP $\Xi^{\mathcal{R}}_j$, whereas (17) is formulated over the suffix MDP $\mathcal{R}_{\text{suf}}$.

Once the solution $z^*_{x,u}$ to (17) is obtained, the optimal policy can be generated by

$$\boldsymbol{\pi}^*_{\text{suf}}(x, u) = \begin{cases} \frac{z^*_{x,u}}{\sum_{\overline{u} \in U^{\mathcal{R}}_{\text{suf}}(x)} z^*_{x,\overline{u}}}, & \text{if } x \in X^*_j \\ \frac{1}{\left| U^{\mathcal{R}}_{\text{suf}}(x) \right|}, & \text{if } x \in X_{\text{suf}} \setminus X^*_j \end{cases} \tag{18}$$

where $X^*_j = \left\{ x \in X_j | \sum_{u \in U^{\mathcal{R}}_{\text{suf}}(x)} z^*_{x,u} > 0 \right\}$.

Lemma 5: The plan suffix $\boldsymbol{\pi}^*_{\text{suf}}$ in (18) solves (15) for the suffix MDP of AMEC $\Xi^{\mathcal{R}}_j$.

Proof: Due to the fact that all input flow from the transient class will eventually end up in A^{in}_j, the second constraint in (17) guarantees the states in A^{in}_j can be eventually reached from $x \in X_{\text{suf}} \backslash A^{in}_j$. Thus, the solution of (17) indicates the accepting states A_j can be visited infinitely often within the AMEC $\Xi^{\mathcal{R}}_j$. Based on the construction of $\mathcal{R}_{\text{suf}}$, the objective function in (17) represents the mean cost of cyclic paths analyzed in (16), which is exactly the ARPS of suffix MDP $\Xi^{\mathcal{R}}_j$ in (15). □

Algorithm 1: Synthesis and Execution of Complete Policy.

procedure Input: $\mathcal{M}$, ϕ, and β
Output: the optimal policy $\boldsymbol{\pi}^*$ and $\boldsymbol{\mu}^*$
Initialization: Construct $\mathcal{A}_\phi$ and $\mathcal{R} = \mathcal{M} \times \mathcal{A}_\phi$
Set $t = 0$ and the execution horizons T.
Construct AMECs $\Xi^{\mathcal{R}}_{\text{acc}} = \{\Xi^{\mathcal{R}}_1, \ldots, \Xi^{\mathcal{R}}_N\}$.
Construct $X_r, X_n, X_{\neg n}, X_{tr}, X'_{tr}$.
if $X_r = \emptyset$ **then**
  $\Xi^{\mathcal{R}}_{\text{acc}}$ cannot be reached from x_0 and no $\boldsymbol{\pi}^*$ exists;
else
  Construct $\mathcal{R}_{\text{pre}}$.
  for each $\Xi^{\mathcal{R}}_j \subseteq \Xi^{\mathcal{R}}_{acc}$ **do**
    Construct $\mathcal{R}_{\text{suf}}$.
  end for
  Obtain $\boldsymbol{\pi}^*$ by solving the coupled LP in 19.
  Set $x_t = x_0 = (s_0, l_0, q_0)$ and $s_t = s_0$.
  Set $\boldsymbol{s}_{\mathcal{M}} = x_t$
  while $t \leq T$ **do**
    Select an action $u^{\mathcal{R}}_t$ according to $\boldsymbol{\pi}^*(x_t)$.
    Obtain s_{t+1} in $\mathcal{M}$ by applying action $a_t = u_t|^{\mathcal{R}}_{\mathcal{M}}$.
    Observe l_{t+1}.
    Set $x_{t+1} = (s_{t+1}, l_{t+1}, q_{t+1})$.
    Update $\boldsymbol{s}$ by concatenating x_{t+1}.
    $t++$.
  end while
  Return $\boldsymbol{\mu}^*(\boldsymbol{s}[:t], L(\boldsymbol{s}[:t])) \forall t = 0, 1 \ldots T$.
end if
end procedure

Remark 5: The above process of constructing suffix MDP in Definition 15, solving optimization problem (12), and synthesizing suffix optimal policies (18) is repetitively applied to every AMEC $\Xi^{\mathcal{R}}_j$ of $\Xi^{\mathcal{R}}_{\text{acc}}$.

To demonstrate the efficiency of our approach, we apply the widely used Round-Robin policy [1] for comparison in the following example and Section VI. After the agent enters into one AMEC $\Xi^{\mathcal{R}}_i = (X_i, U^{\mathcal{R}}_i)$, an ordered sequence of actions from $U^{\mathcal{R}}_i(x) \forall x \in X_i$ is created. The Round-Robin policy guides the agent to visit each state by iterating over the ordered actions, and this ensures all states of the AMEC are visited infinitely often (i.e., satisfying the acceptance condition). For decision-making within an AMEC, the Round-Robin policy does not consider optimality.

Example 4: As another running example in Fig. 5 , we demonstrate the importance of optimizing ARPS to the motion planning after entering into AMECs compared with the Round-Robin policy. The motion uncertainties and the action cost are set as the same as in Example 1 . An LTL task is considered as

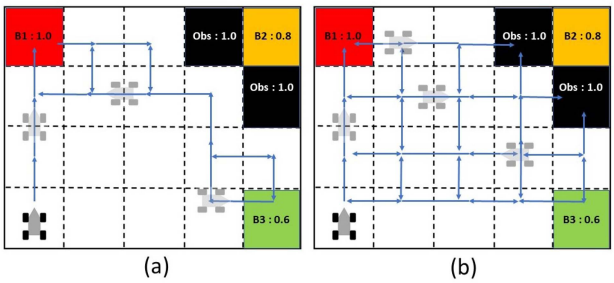


Fig. 5. Simulated trajectories with different suffix policies. (a) Optimal policy generated from our framework. (2) The Round-Robin policy.

$\phi_{\text{suf}} = \square\diamondsuit \texttt{B1} \wedge \square\diamondsuit \texttt{B2} \wedge \square\diamondsuit \texttt{B3}$ that requires to infinitely often visit regions B1, B2, and B3. ϕ_{suf} is infeasible with respect to the corresponding PL-MDP (B3 is surrounded by obstacles). Fig. 5 shows the results with two different policies. Without minimizing the AVPS, the Round-Robin policy cannot be applied to the infeasible cases using the relaxed product MDP $\mathcal{R}$. In addition, the framework in [17] returns no solution for this example.

D. Complete Policy and Complexity

A complete optimal stationary policy $\boldsymbol{\pi}^*$ can be obtained by concatenating the procedure of solving the linear programs in (12) and (17) as

$$\min_{\{y_{x,u}, z_{x,u}\}} \left[(1-\eta) \cdot J_{\text{pre}} + \eta \cdot \sum_{\Xi_j^{\mathcal{R}} \in \Xi_{Acc}^{\mathcal{R}}} J_{\Xi_j^{\mathcal{R}}} \right] \quad \text{s.t. Constraints in (12) and (17)} \tag{19}$$

where $J_{\text{pre}}, y_{x,u}$ and $J_{\Xi_j^{\mathcal{R}}}, z_{x,u}$ are defined in (12) and (17) respectively, and η is a tradeoff parameter to balance the importance of minimizing the ARPS between prefix plan and suffix plan. The equation (19) can be solved via any LP solvers, i.e., Gurobi [47] and CPLEX. [1] Once the optimal solutions $y_{x,u}^*$ and $z_{x,u}^*$ are generated, we can synthesize the optimal policies $\boldsymbol{\pi}_{\text{pre}}^*$ and $\boldsymbol{\pi}_{\text{suf}}^*$ via (13) and (18). The complete optimal policy $\boldsymbol{\pi}^*$ can be obtained by concatenating π_{pre}^* and π_{suf}^* for all states of $\mathcal{R}$.

Since $\boldsymbol{\pi}^*$ is defined over $\mathcal{R}$, to execute the optimal policy over $\mathcal{M}$, we still need to map $\boldsymbol{\pi}^*$ to an optimal finite-memory policy $\boldsymbol{\mu}^*$ of $\mathcal{M}$. Suppose the agent starts from an initial state $x_0 = (s_0, l_0, q_0)$ and the distribution of optimal actions at $t = 0$ is given by $\boldsymbol{\pi}^*(x_0)$. Taking an action $u_0^{\mathcal{R}}$ according to $\boldsymbol{\pi}^*(s_0)$, the agent moves to s_1 and observes its current label l_1, resulting in $x_1 = (s_1, l_1, q_1)$ with $q_1 = \delta(q_0, u_0|_{\mathcal{A}}^{\mathcal{R}})$. Note that, q_1 is deterministic if $u_0|_{\mathcal{A}}^{\mathcal{R}} \neq \epsilon$. The distribution of optimal actions at $t = 1$ now becomes $\boldsymbol{\pi}^*(x_1)$. Repeating this process infinitely will generate a path $\boldsymbol{x}_{\mathcal{R}}^{\boldsymbol{\pi}^*} = x_0 x_{1\ldots}$ over $\mathcal{R}$, corresponding to a path $\boldsymbol{s} = s_0 s_1 \ldots$ over $\mathcal{M}$ with associated labels $L(\boldsymbol{s}) = l_0 l_1 \ldots$. Such a process is presented in Algorithm 1. Since the state x_t is unique given the agent's past path $\boldsymbol{s}[:t]$ and past labels $L(\boldsymbol{s}[:t])$ up to t, the optimal finite-memory policy is designed as

$$\boldsymbol{\mu}^*\left(\boldsymbol{s}[:t], L(\boldsymbol{s}[:t])\right) = \begin{cases} \boldsymbol{\pi}^*(x_t), & \text{for } u_t|_{\mathcal{M}}^{\mathcal{R}} = a \\ 0, & \text{for } u_t|_{\mathcal{A}}^{\mathcal{R}} = \epsilon \end{cases}. \tag{20}$$

From Definition 11, the state s_t in x_t remains the same if $u_t|_{\mathcal{M}}^{\mathcal{R}} = \epsilon$ which gives rise to $\boldsymbol{\mu}^*(\boldsymbol{s}[:t], L(\boldsymbol{s}[:t])) = 0$ in (20).

Theorem 2: Given a PL-MDP and an LTL formula ϕ, the optimal policy $\boldsymbol{\mu}^*$ from (19) and in (20) solves the Problem 1 exactly s.t. achieve multiple objectives in order of decreasing priority: 1) If ϕ is fully feasible, $\Pr_{\mathcal{M}}^{\boldsymbol{\mu}}(\phi) \geq \gamma$ with $\gamma \in (0, 1]$; 2) if ϕ is infeasible, satisfy ϕ as much as possible via minimizing AVPS; and 3) minimize AEPS over the infinite horizons.

Proof: First, the optimal policy $\boldsymbol{\pi}^*$ solves Problem 2 exactly. Such a conclusion can be verified directly based on Theorem 1, Lemma 4, and Lemma 5. Because Problem 1 and Problem 2 are equivalent, the policy projection in (20) finds a policy in $\mathcal{M}$ that solves Problem 1 exactly [1]. □

In Algorithm 1, the overall policy synthesis is summarized in lines 1–12 of Algorithm Note that, the optimization process of suffix plan (line 9–11) is applied to every AMEC of $\mathcal{R}$. After obtaining the complete optimal policies, the process of executing such a policy for PL-MDP $\mathcal{M}$ is outlined in lines 13–24.

Remark 6: The complete policy developed in the work can handle both feasible and infeasible cases simultaneously, and AMECs of relaxed product MDP are computed off-line once based on the algorithms of [1].

E. Complexity Analysis

The maximum number of states is $|X| = |S| \times |L_{\max}(S)| \times |Q|$, where $|Q|$ is determined by the LDBA $\mathcal{A}_\phi$, $|S|$ is the size of the environment, and $L_{\max}(S)$ is the maximum number of labels associated with a state $s \in S$. Due to the consideration of relaxed product MDP and the extended actions, the maximum complexity of actions available at $x_0 = (s_0, l_0, q_0) \in X$ is $O(|\mathcal{A}(s)| \times |\Sigma \cup \{\epsilon\}|)$. From [1], the complexity of computing AMECs for $\mathcal{R}$ is $O(|X|^2)$. The size of LPs in (12) and (17) is linear with respect to the number of transitions in $\mathcal{R}$ and can be solved in polynomial time [48].

VI. Case Studies

Here considers a mobile agent operating in a grid environment, which is a commonly used benchmark for probabilistic model checking in the literature [9]–[11], and [17]. There are properties of interest associated with the cells. To model environment uncertainties, these properties are assumed to be probabilistic. We consider the same motion uncertainties as Example 1. The agent is allowed to transit between adjacent cells or stay in a cell, i.e., the action space is $\{\texttt{Up}, \texttt{Right}, \texttt{Down}, \texttt{Left}, \texttt{Stay}\}$, and the action costs are $[3, 4, 2, 3, 1]$. To model the agent's motion uncertainty caused by actuation noise and drifting, the agent's motion is also assumed to be probabilistic. For instance, the robot may successfully take the desired action with a probability of 0.85, and there is a probability of 0.15 to take other perpendicular actions based on uniform distributions. There is no motion uncertainty for the action of "Stay." In the following

[1] [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

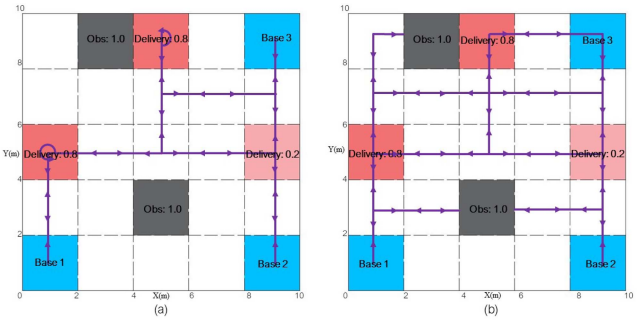


Fig. 6. Simulated trajectories by (a) the optimal policy and (b) the Round-Robin policy. The line arrows represent the directions of movement and the circle arrows represent the `Stay` action.

cases, the algorithms developed in Section V are implemented, where $\beta = 100$ is employed to encourage a small violation of the desired task if the task is infeasible. The desired satisfaction probability is set as $\gamma = 0.9$. Gurobi [47] is used to solve the linear program problems in (12) and (17). All algorithms are implemented in Python 2.7, and Owl [49] is used to convert LTL formulas into LDBA. All simulations are carried out on a laptop with a 2.60 GHz quad-core CPU and 8 GB of RAM.

A. Case 1: Feasible Tasks

This case considers motion planning in an environment, where the desired task can be completely fulfilled. Suppose the agent is required to perform a surveillance task in a workspace, as shown in Fig. 6, and the task specification is expressed in the form of LTL formula as

$$\begin{aligned}\varphi_{\text{case1}} = \; & (\square\lozenge \texttt{base1}) \wedge (\square\lozenge \texttt{base2}) \wedge (\square\lozenge \texttt{base3}) \\ & \wedge \square\left(\varphi_{\text{one}} \rightarrow \bigcirc\left((\neg\varphi_{\text{one}}) \cup \texttt{Delivery}\right)\right) \\ & \wedge \square\neg \texttt{Obs} \end{aligned} \tag{21}$$

where $\varphi_{\text{one}} = \texttt{base1} \vee \texttt{base2} \vee \texttt{base3}$. The LTL formula in (21) means that the agent visits one of the base stations and then goes to one of the delivery stations, while avoiding obstacles. All base stations need to be visited. Based on the environment and motion uncertainties, the LTL formula φ_{case1} with respect to PL-MDP is feasible. The corresponding LDBA has 35 states and 104 transitions, and the PL-MDP has 28 states. It took 11.2 s to construct the relaxed product MDP and 0.15 s to synthesize the optimal policy via Algorithm 1. To demonstrate the efficiency, we also compare the optimal policies generated from this with the Round-Robin policy.

Fig. 6(a) and (b) shows the trajectories generated by our optimal policy and the Round-Robin policy, respectively. The arrows represent the directions of movement, and the circles represent the `Stay` action. Clearly, the optimal policy is more efficient in the sense that fewer cells were visited during mission operation. In Fig. 7, 1000 Monte Carlo simulations were conducted. Fig. 7(a) shows the distribution of the plan suffix cost. It indicates that, since the task is completely feasible, the optimal policy in this work can always find feasible plans with zero AVPS. Since Round-Robin policy would select all available

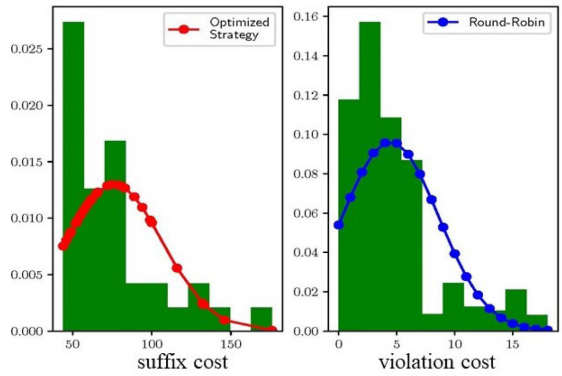


Fig. 7. (a) Normalized distribution of the plan suffix cost under the optimal policy. (b) Normalized distribution of the violation cost under the Round-Robin policy.

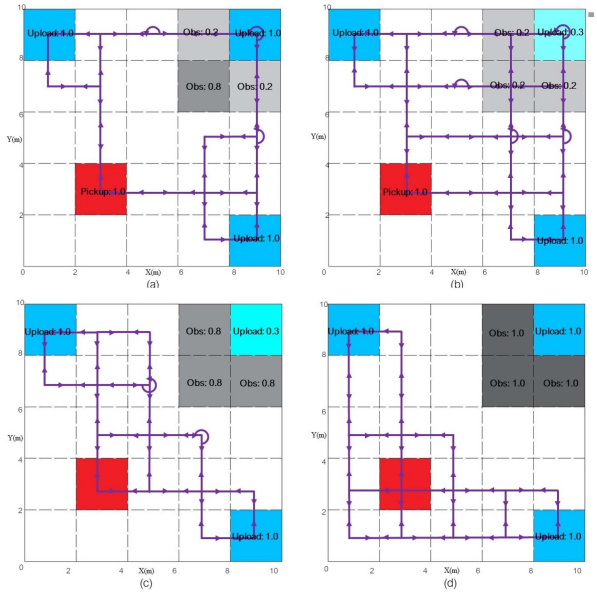


Fig. 8. Simulated trajectories by the optimal policy for different environments.

actions enabled at each state of AMEC, Fig. 7(b) shows the distribution of the violation cost under Round-Robin policy.

B. Case 2: Infeasible Tasks

This case considers motion planning in an environment, where the desired task might not be fully executed. In Fig. 8, suppose the agent is tasked to visit the `pickup` station and then goes to one of the `upload` stations, while avoiding obstacles. In addition, the agent is not allowed to visit the `pickup` station before visiting an `upload` station, and all `upload` stations need to be visited. Such a task can be written in an LTL formula as

$$\begin{aligned}\varphi_{\text{case2}} = \; & \square\lozenge \texttt{Pickup} \wedge \square\neg \texttt{Obs} \\ & \wedge \square\left(\texttt{Pickup} \rightarrow \bigcirc\left((\neg \texttt{Pickup}) \cup \varphi_{\text{one}}\right)\right) \\ & \wedge \square\lozenge \texttt{Upload1} \wedge \square\lozenge \texttt{Upload2} \wedge \square\lozenge \texttt{Upload3} \end{aligned} \tag{22}$$

where $\varphi_{\text{one}} = \texttt{Upload1} \vee \texttt{Upload2} \vee \texttt{Upload3}$. Fig. 8(a)–(c) shows infeasible tasks since the cells surrounding `Upload2` are

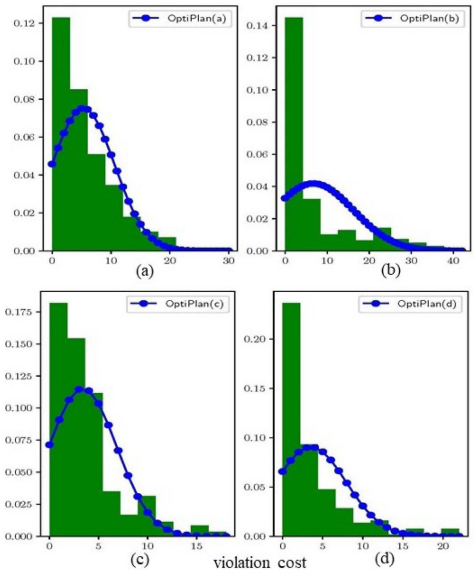


Fig. 9. Normalized distributions of the violation cost for different environments.

occupied by obstacles probabilistically. Fig. 8(d) shows an infeasible environment since `Upload2` is surrounded by obstacles for sure and can never be reached.

Simulation results show how the relaxed product MDP can synthesize an optimal plan when no AMECs or no ASCCs exist. Note that, the algorithm in [17] returns no solution if no ASCCs exist. The resulting LDBA has 43 states and 136 transitions, and it took 0.15 s on average to synthesize the optimal policy. The simulated trajectories are shown in Fig. 8 with arrows indicating the directions of movement. In Fig. 8(a), since the probability of `Upload2` is high and the probabilities of surrounding obstacles are relatively low, the planning tries to complete the desired task φ_{case2}. In Fig. 8(b) and (c), the probability of `Upload2` is 0.3, while the probabilities of surrounding obstacles are 0.2 and 0.8, respectively. The agent still tries to complete φ_{case2} by visiting `Upload2` in Fig. 8(b), while the agent is relaxed to not visit `Upload2` in Fig. 8(c) due to the high risk of running into obstacles and low probability of `Upload2`. Since φ_{case2} is completely infeasible in Fig. 8(d), the motion plan is revised to not visit `Upload2` and select paths with the minimum violation and implementation cost to mostly satisfy φ_{case2}. To illustrate the ability to minimize AVPS for infeasible cases, we analyze the violations such that Fig. 9 shows the distribution of AVPS for 1000 Monte Carlo simulations corresponding to the four different infeasible cases in Fig. 8, respectively. It can be observed that there is a high probability of obtaining a small AVPS with this framework.

C. Parameter Analysis and Results Comparison

In this section, we first apply the feasible task ϕ_{case1} to analyze the effect of η in (19) on the tradeoff between the optimal expected execution costs of prefix and suffix plans. The results are shown in Table II. Then we compared our framework referred as "LDBA" with widely used model-checking tool PRISM [38].

TABLE II
EXPECTED EXECUTION COSTS USING DIFFERENT PARAMETERS η FOR ϕ_{CASE1}

η parameter	Total cost Prefix	Cyclic cost Suffix	Mean Cost Suffix
0	36.4	178.5	2.823
0.2	36.7	66.1	2.545
0.4	36.7	65.8	2.540
0.6	39.4	62.1	2.538
0.8	50.9	57.2	2.520
1.0	115.6	55.9	2.512

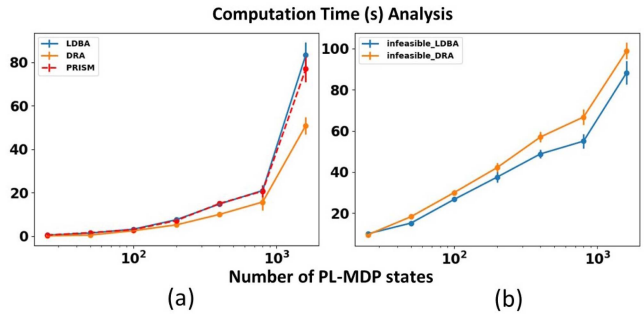


Fig. 10. Computation time for different methods. (a) Computation time of solving the optimization process using "LDBA," "DRA," PRISM for feasible cases. (b) Computation time of overall process (construct models and solve the optimization) using "LDBA," "DRA" for infeasible cases.

To implement PRSIM with the PL-MDP, we use the package [17] to translate the relaxed product automaton into PRISM language and verify the LDBA accepting condition. We select the option of PRISM "multiobjective property" that finds the policies satisfying task with the risk lower bounded by $1-\gamma$, while minimizing cumulative reward. The tool PRSIM can only synthesize the optimal prefix policy and does not support the optimization of suffix structure to handle infeasible cases. Thus, we compare the computation time for the feasible LTL task ϕ_{case1} with its environments, divide each grid of the environment to construct various workspace sizes, and run 10 times for each environment with the same size, where the initial locations are selected from uniform distributions. We compare such complexity of the feasible case ϕ_{case1} with the works [17]–[19] named as "DRA" that uses the seterministic Rabin automaton and standard product MDP to synthesize solutions. The results are shown in Fig. 10(a). We can see the computation time of the optimization process with PRISM is almost the same. And since the relaxed product MDP is more connected than standard product MDP, the computation time of our work is a little higher than the works [17]–[19] for feasible cases.

As for infeasible cases, even though the algorithm [17] returns no solutions for cases described in Figs. 1(a), 5, and 8(d), it still has solutions for case ϕ_{case2} in Fig. 8(a)–(c). We refer to the algorithm [17] as "infeasible DRA," and our framework as "infeasible LDBA". To analyze the computational time, we select the environment of Fig. 8(b). We also divide each grid to generate various workspace sizes and run 10 times for each environment with the same size, where the initial locations are

TABLE III
COMPARISON OF WORKSPACE SIZE AND COMPUTATION TIME

Workspace size[cell]	$\mathcal{M}$ Time[s]	$\mathcal{R}$ Time[s]	AMECs Time[s]	π^* Time[s]
5×5	0.14	0.56	0.64	0.45
10×10	1.59	1.34	1.88	3.20
30×30	25.4	5.20	7.41	20.71
50×50	460.1	28.95	25.89	124.03
100×100	843.9	41.47	39.80	276.05

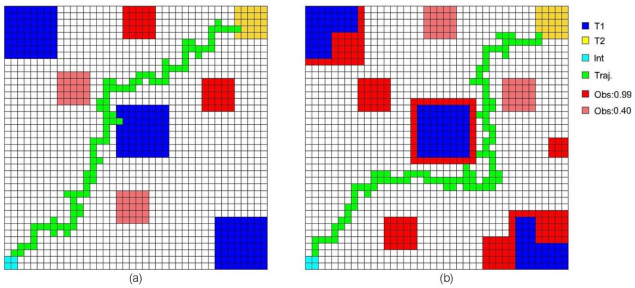


Fig. 11. Simulation results for the specification φ_{case3}. (a) shows a feasible case and (b) shows an infeasible case where $T1$ can not be visited.

selected from uniform distributions. The results of computation time for the overall process are shown in Fig. 10(b). It shows that our algorithm has better computational performance because the work [17] needs to construct AMECs first to check the feasibility and then construct ASCCs to start the optimization process, which is computationally expensive.

D. Case 3: Large-Scale Analysis

This case considers motion planning in a larger scale problem. To show the efficiency of using LDBA, we first repeat the task of Case 1 for different workspace sizes. Table III lists the computation time for the construction of PL-MDP, the relaxed product MDP, AMECs, and the optimal plan π^* in different workspace sizes.

To demonstrate the scalability and computational complexity, consider a 40×40 workspace as in Fig. 11. The desired task expressed in an LTL formula is given by

$$\varphi_{\text{case3}} = \square\neg\texttt{Obs} \wedge \lozenge\texttt{T1} \wedge \square\left(\texttt{T1} \rightarrow \bigcirc\left(\neg\texttt{T1}\,\mathcal{U}\,\texttt{T2}\right)\right)$$

where T1 and T2 represent two targets properties to be visited sequentially. The agent starts from the left corner (i.e., the light blue cell). The LDBA associated with φ_{case3} has 6 states and 17 transitions, and it took 27.3 s to generate an optimal plan. The simulation trajectory is shown in Fig. 11. Note that, AMECs of $\mathcal{P}$ only exist in Fig. 11(a). Neither AMECs nor ASCCs of $\mathcal{P}$ exist in Fig. 11(b), since T1 is surrounded by obstacles and cannot be reached. Clearly, the desired task φ_{case3} can be mostly and efficiently executed whenever the task is feasible or not.

E. Mock-Up Office Scenario

In this section, we verify our algorithm for high-level decision-making problems in a real-world environment, and

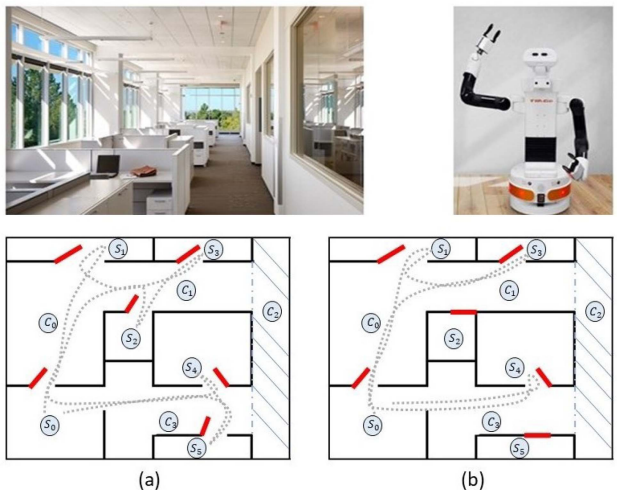


Fig. 12. Generated trajectories for the task using φ_{case4} in the mock-up office scenario with the TIAGo robot.

show that the framework can be adopted with any stochastic abstractions and low-level noisy controllers to formulate a hierarchical architecture. Consider a TIAGo robot operating in an office environment, as shown in Fig. 12, which can be modeled in ROS Gazebo in Fig. 12. The mock-up office consists of 6 rooms S_i, $i = 0, \ldots, 5$, and 4 corridors C_i, $i = 0, \ldots, 3$. The TIAGo robot can follow a collision-free path from the center of one region to another without crossing other regions using obstacle-avoidance navigation. The marked area C_2 represents an inclined surface, where more control effort is required by TIAGo robot to walk through. To model motion uncertainties, it is assumed that the robot can follow its navigation controller moving to the desired region with a probability of 0.9 and fail by moving to the adjacent region with a probability of 0.1. The resulting MDP has 10 states.

The LTL task is formulated as

$$\varphi_{\text{case4}} = \square\lozenge S_0 \wedge \square\lozenge S_1 \wedge \square\lozenge S_2 \wedge \square\lozenge S_3 \wedge \square\lozenge S_4 \wedge \square\lozenge S_5$$

which requires the robot to periodically serve all rooms. Its corresponding LDBA has 6 states with 6 accepting states and the relaxed product MDP has 60 states. The simulated trajectories are shown in Fig. 12(a) and (b). The task is satisfied exactly in Fig. 12(a) because all rooms are accessible. In Fig. 12(b), φ_{case4} is only feasible since rooms S_2 and S_5 are closed. Hence, the robot revises its plan to only visit rooms S_0, S_1, S_3, and S_4. In both Fig. 12(a) and (b), C_2 is avoided for energy efficiency.

VII. CONCLUSION

A plan synthesis algorithm for probabilistic motion planning is developed for both feasible and infeasible tasks. LDBA is employed to evaluate the LTL satisfaction with the Büchi acceptance condition. The extended actions and relaxed product MDP are developed to allow probabilistic motion revision if the workspace is not fully feasible to the desired mission. Cost optimization is studied in both plan prefix and plan suffix of the trajectory. Inspired by the existing works, e.g., [13], future research will consider the optimization of multiobjective over

continuous space with safety-critical constraints. Additional in-depth research includes extending this work to multiagent systems with cooperative tasks.

Acknowledgment

The authors would like to thank M. Guo for the open-source software. The authors would also like to thank the editor and anonymous reviewers for their time and efforts in helping improve the paper.

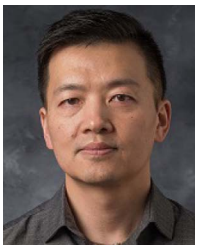
References

[1] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT press, 2008.
[2] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans. Autom. Control*, vol. 53, no. 1, pp. 287–297, Feb. 2008.
[3] Y. Kantaros and M. M. Zavlanos, "Sampling-based optimal control synthesis for multirobot systems under global temporal tasks," *IEEE Trans. Autom. Control*, vol. 64, no. 5, pp. 1916–1931, May 2019.
[4] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 363–374, Apr. 2021.
[5] A. Ulusoy, T. Wongpiromsarn, and C. Belta, "Incremental controller synthesis in probabilistic environments with temporal logic constraints," *Int. J. Robot. Res.*, vol. 33, no. 8, pp. 1130–1144, 2014.
[6] P. Jagtap, S. Soudjani, and M. Zamani, "Formal synthesis of stochastic systems via control barrier certificates," *IEEE Trans. Autom. Control*, vol. 66, no. 7, pp. 3097–3110, Jul. 2020.
[7] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Trans. Robot.*, vol. 28, no. 2, pp. 396–409, Apr. 2012.
[8] P. Nuzzo, J. Li, A. L. Sangiovanni-Vincentelli, Y. Xi, and D. Li, "Stochastic assume-guarantee contracts for cyber-physical system design," *ACM Trans. Embed Comput. Syst.*, vol. 18, no. 1, 2019, Art. no. 2.
[9] D. Sadigh, E. S. Kim, S. Coogan, S. S. Sastry, and S. A. Seshia, "A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications," in *Proc. 53rd IEEE Conf. Decis. Control* 2014, pp. 1091–1096.
[10] M. Hasanbeig, A. Abate, and D. Kroening, "Logically-constrained neural fitted Q-iteration," 2018, *arXiv:1809.07823*.
[11] M. Hasanbeig, A. Abate, and D. Kroening, "Certified reinforcement learning with logic guidance," 2019, *arXiv:1902.00778*.
[12] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021.
[13] M. Cai and C.-I. Vasile, "Safety-critical modular deep reinforcement learning with temporal logic through Gaussian processes and control barrier functions," 2021, *arXiv:2109.02791*.
[14] M. Svoreňová, I. Černá, and C. Belta, "Optimal control of MDPs with temporal logic constraints," in *Proc. IEEE Conf. Decis. Control*, 2013, pp. 3938–3943.
[15] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Robot. Res.*, vol. 30, no. 14, pp. 1695–1708, 2011.
[16] X. Ding, S. L. Smith, C. Belta, and D. Rus, "Optimal control of Markov decision processes with linear temporal logic constraints," *IEEE Trans. Autom. Control*, vol. 59, no. 5, pp. 1244–1257, May 2014.
[17] M. Guo and M. M. Zavlanos, "Probabilistic motion planning under temporal tasks and soft constraints," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4051–4066, Dec. 2018.
[18] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu, "Quantitative multi-objective verification for probabilistic systems," in *Proc. Int. Conf. Tool. Algorithm. Const. Anal. Syst*, 2011, pp. 112–127.
[19] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *Proc. Int. Symp. Autom. Tech. Verification Anal.* Springer, 2012, pp. 317–332.
[20] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.
[21] K. Kim and G. E. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 265–271.
[22] K. Kim, G. E. Fainekos, and S. Sankaranarayanan, "On the revision problem of specification automata," in *Proc. Int. Conf. Robot. Automat.*, 2012, pp. 5171–5176.
[23] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus, "Least-violating control strategy synthesis with safety rules," in *Proc. Int. Conf. Hybrid Syst. Comput. Control*, 2013, pp. 1–10.
[24] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control-based motion planning with partially infeasible LTL constraints," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1279–1284, Oct. 2020.
[25] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazicioğlu, "Distributed safe planning for satisfying minimal temporal relaxations of TWTL specifications," *Robot. Auton. Syst.*, vol. 142, 2021, Art. no. 103801.
[26] Z. Li, M. Cai, S. Xiao, and Z. Kan, "Online motion planning with soft timed temporal logic in dynamic and unknown environment," 2021, *arXiv:2110.09007*.
[27] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, "Minimum-violation scLTL motion planning for mobility-on-demand," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1481–1488.
[28] T. Wongpiromsarn, K. Slutsky, E. Frazzoli, and U. Topcu, "Minimum-violation planning for autonomous systems: Theoretical and practical considerations," in *Proc. IEEE Amer. Control Conf.*, 2021, pp. 4866–4872.
[29] M. Cai, H. Peng, Z. Li, and Z. Kan, "Learning-based probabilistic LTL motion planning with environment and motion uncertainties," *IEEE Trans. Autom. Control*, vol. 66, no. 5, pp. 2386–2392, May 2021.
[30] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Reinforcement learning based temporal logic control with soft constraints using limit-deterministic generalized Buchi automata," 2021, *arXiv:2101.10284*.
[31] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 583–599, Jun. 2016.
[32] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *Int. J. Robot. Res.*, vol. 38, no. 9, pp. 1098–1123, 2019.
[33] L. Niu, J. Fu, and A. Clark, "Optimal minimum violation control synthesis of cyber-physical systems under attacks," *IEEE Trans. Autom. Control*, vol. 66, no. 3, pp. 995–1008, Mar. 2021.
[34] S. Sickert, J. Esparza, S. Jaax, and J. Křetínskỳ, "Limit-deterministic Büchi automata for linear temporal logic," in *Proc. Int. Conf. Comput. Aided Verification*, 2016, pp. 312–332.
[35] M. Hasanbeig, Y. Kantaros, A. Abate, D. Kroening, G. J. Pappas, and I. Lee, "Reinforcement learning for temporal logic control synthesis with probabilistic satisfaction guarantees," in *Proc. IEEE 58th Conf. Decis. Control*, 2019, pp. 5338–5343.
[36] A. K. Bozkurt, Y. Wang, M. M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 10349–10355.
[37] M. Cai, S. Xiao, B. Li, Z. Li, and Z. Kan, "Reinforcement learning based temporal logic control with maximum probabilistic satisfaction," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 806–812.
[38] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *Proc. Int. Conf. Comput. Aided Verification*, 2011, pp. 585–591.
[39] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA:Wiley, 2014.
[40] E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang, "Lazy probabilistic model checking without determinisation," 2013, *arXiv:1311.2928*.
[41] M. Y. Vardi, "Automatic verification of probabilistic concurrent finite state programs," in *Proc. IEEE 26th Annu. Symp. Found. Comput. Sci.*, 1985, pp. 327–338.
[42] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," *J. Assoc. Comput. Machinery*, vol. 42, no. 4, pp. 857–907, 1995.
[43] M. Randour, J.-F. Raskin, and O. Sankur, "Percentile queries in multi-dimensional Markov decision processes," in *Proc. Int. Conf. Comput. Aided Verification*, 2015, pp. 123–139.
[44] T. Br'zdil, V. Brozek, K. Chatterjee, V. Forejt, and A. Kucera, "Two views on multiple mean-payoff objectives in Markov decision processes," in *Proc. IEEE Symp. Log. Comput. Sci.*, 2011, pp. 33–42.
[45] R. Durrett, *Essentials of Stochastic Processes*, vol. 1, 2nd ed. Berlin, Germany: Springer, 2012.

[46] K. Etessami, M. Kwiatkowska, M. Y. Vardi, and M. Yannakakis, "Multi-objective model checking of Markov decision processes," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, 2007, pp. 50–65.
[47] Gurobi Optimizer Reference Manual. Gurobi Optimization LLC., Houston, TX, USA, 2021. [Online]. Available: https://www.gurobi.com
[48] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 1998.
[49] J. Kretínský, T. Meggendorfer, and S. Sickert, "Owl: A library for ω-words, automata, and LTL," in *Proc. Autom. Tech. Verification. Anal.*, 2018, pp. 543–550. [Online]. Available: https://doi.org/10.1007/978-3-030-01090-4_34

Mingyu Cai received the B.Eng. degree in aerospace engineering from the Beijing Institute of Technology, Beijing, China in 2015, the M.S.E degree in mechanical and aerospace engineering from the University of Florida, in 2017, the Ph.D. degree in mechanical engineering with the University of Iowa, Iowa City, IA, USA, in 2021.

He is currently a Postdoctoral Associate with the Department of Mechanical Engineering, Lehigh University, Bethlehem, USA. His research interests include robotics, machine learning, control theory, formal methods, with applications to motion planning, decision-making, nonlinear control, and autonomous driving.



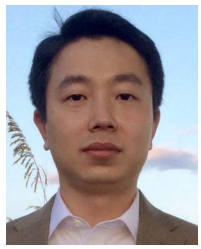
Shaoping Xiao received the Ph.D. degree in mechanical engineering from Northwestern University, in 2003.

He is an Associate Professor with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA, USA. His original expertise lies in computational mechanics and materials science, and one of his papers has been cited over 1000 times. In the past several years, he has extended his efforts 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 to mitigate the flood, intelligent traffic light, and quantum computing.

Zhijun Li (Senior Member, IEEE) received the Ph.D. degree in mechatronics engineering from Shanghai Jiao Tong University, Shanghai, China, in 2002.

From 2003 to 2005, he was a Postdoctoral Fellow with the Department of Mechanical Engineering and Intelligent systems, University of Electro-Communications, Tokyo, Japan. From 2005 to 2006, he was a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, and Nanyang Technological University, Singapore. From 2017, he has been a Professor with the Department of Automation, University of Science and Technology, Hefei, China. From 2019, he is the Vice Dean of School of Information Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include wearable robotics, teleoperation systems, nonlinear control, neural network optimization, etc.

Dr. Li's is serving as an Editor-at-large for *Journal of Intelligent & Robotic Systems*, and Associate Editors of several IEEE Transactions. He has been the Co-Chair of IEEE Systems, Man, and Cybernetics Society Technical Committee on Bio-mechatronics and Bio-robotics Systems (B^2S), and IEEE-Robotics & Automation Society Technical Committee on Neuro-Robotics Systems, from 2016.



Zhen Kan (Member, IEEE) received the Ph.D. degree in mechanical engineering from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2011.

He was a Postdoctoral Research Fellow with the Air Force Research Laboratory (AFRL), Eglin AFB, FL, USA, and the University of Florida Research and Engineering Education Facility, Shalimar, FL. USA, from 2012 to 2016, and an Assistant Professor with the Department of Mechanical Engineering, University of Iowa, Iowa City, IA, USA, from 2016 to 2019. He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. His research interests include networked control systems, nonlinear control, formal methods, and robotics.

Dr. Kan is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL. He currently serves on program committees of several internationally recognized scientific and engineering conferences.