Proceedings of Machine Learning Research vol 283:1–12, 2025 7th Annual Conference on Learning for Dynamics and Control

Probabilistic Satisfaction of Temporal Logic Constraints in Reinforcement Learning via Adaptive Policy-Switching

Xiaoshan Lin LIN00668@UMN.EDU
University of Minnesota, Minnesota, USA

Sadık Bera Yüksel YUKSEL.SA@NORTHEASTERN.EDU
Yasin Yazıcıoğlu Y.YAZICIOGLU@NORTHEASTERN.EDU
Derya Aksaray D.AKSARAY@NORTHEASTERN.EDU
Northeastern University, Massachusetts, USA

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

Constrained Reinforcement Learning (CRL) is a subset of machine learning that introduces constraints into the traditional reinforcement learning (RL) framework. Unlike conventional RL which aims solely to maximize cumulative rewards, CRL incorporates additional constraints that represent specific mission requirements or limitations that the agent must comply with during the learning process. In this paper, we address a type of CRL problem where an agent aims to learn the optimal policy to maximize reward while ensuring a desired level of temporal logic constraint satisfaction throughout the learning process. We propose a novel framework that relies on switching between pure learning (reward maximization) and constraint satisfaction. This framework estimates the probability of constraint satisfaction based on earlier trials and properly adjusts the probability of switching between learning and constraint satisfaction policies. We theoretically validate the correctness of the proposed algorithm and demonstrate its performance through comprehensive simulations.

Keywords: Reinforcement Learning, Formal Methods in Robotics and Automation, Constraint Satisfaction

1. Introduction

Reinforcement learning (RL) relies on learning optimal policies through trial-and-error interactions with the environment. However, many real-life systems need to not only maximize some objective function but also satisfy certain constraints on the system's trajectory. Conventional formulations of constrained RL (e.g., Achiam et al. (2017); Chow et al. (2018); Garcıa and Fernández (2015)) focus on maximizing reward functions while keeping some cost function below a certain threshold. In contrast, robotic systems often require adherence to more intricate spatial-temporal constraints. For instance, a robot should "pick up from region A and deliver to region B within a specific time window, while avoiding collisions with any object".

Temporal logic (TL) is a formal language that can express spatial and temporal specifications. In recent years, RL subject to TL constraints has gained significant interest, especially in the robotics community. One common approach involves encoding constraint satisfaction into the reward function and learning a policy by maximizing the cumulative reward (e.g., Li et al. (2017); Aksaray et al. (2016)). Another approach focuses on modifying the exploration process during RL, such as the shielded RL proposed in Alshiekh et al. (2018) that corrects unsafe actions to satisfy Linear Temporal Logic (LTL) constraints. Similarly, Hasanbeig et al. (2020) constructs a safe padding based on maximum likelihood estimation and Bellman update, combined with a state-adaptive reward function, to maximize the probability of satisfying LTL constraints. A model-based approach is introduced for safe exploration in deep RL by Cai and Vasile (2021), which employs Gaussian process estimation and control barrier functions to ensure a high likelihood of satisfying LTL constraints. Although these approaches focus on maximizing the probability of satisfaction, they do not provide guarantees on satisfying TL constraints with a *desired probability* during the learning process. Moreover, Jansen et al. (2020) proposes a method based on probabilistic shield and model checking to ensure the satisfaction of LTL specifications under model uncertainty. However, this method lacks guarantees during the early stages of the learning process. Finally, Aksaray et al. (2021) and Lin et al. (2023) assume partial knowledge about the system model and leverage it to prune unsafe actions, thus ensuring the satisfaction of Bounded Temporal Logic (BTL) with a desired probabilistic guarantee throughout the learning process. However, both methods require learning over large state spaces, causing scalability issues. Furthermore, Aksaray et al. (2021), which is the closest work to this paper, is only applicable to a more restrictive family of BTL formulas.

Driven by the need for a computationally efficient solution that offers probabilistic constraint satisfaction guarantees throughout the learning process, even in the first episode, we propose a novel approach that enables the RL agent to alternate between two policies during the learning process. The first policy is a stationary policy that prioritizes satisfying the BTL constraint, while the other employs RL to learn a policy on the MDP that only maximizes the cumulative reward. The proposed algorithm estimates the satisfaction rate of following the first policy and adaptively updates the switching probability to balance the need for constraint satisfaction and reward maximization. We theoretically show that, with high confidence, the proposed approach satisfies the BTL constraint with a probability greater than the desired threshold. We also validate our approach via simulations.

2. Preliminaries: Bounded Temporal Logic

We denote the set of positive integers by $\mathbb{Z}^+$, the set of atomic propositions by AP, and the power set of a finite set Σ by 2^Σ. Bounded temporal logics (BTL) (e.g., Bounded Linear Temporal Logic Zuliani et al. (2010), Interval Temporal Logic Cau and Zedan (1997), and Time Window Temporal Logic (TWTL) Vasile et al. (2017)) are expressive languages that enable users to define specifications with explicit time-bounds (e.g., "visit region A and then region B within a desired time interval"). In this paper, we focus on BTL that can be translated into a finite-state automaton.

Definition 1. *(Finite State Automaton) A finite state automaton (FSA) is a tuple $\mathscr{A} = (Q, q_{init}, 2^\Sigma, \delta, F)$, where Q is a finite set of states, q_{init} is the initial state, 2^Σ is the input alphabet, $\delta : Q \times 2^\Sigma \to Q$ is a transition function, F is the set of accepting states.*

While our proposed methods can be applied to any BTL that can be translated into an FSA, we will use TWTL specifications in our examples. Hence, we also provide some relevant preliminaries here. A TWTL Vasile et al. (2017) formula over a set of atomic propositions Σ is defined as follows:

$$\phi ::= H^d s \mid H^d \neg s \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg \phi_1 \mid \phi_1 \cdot \phi_2 \mid [\phi_1]^{[a,b]}.$$

Here, s either represents the constant "true" or an atomic proposition in AP; ϕ_1 and ϕ_2 are TWTL formulas; $\wedge$, $\vee$, and $\neg$ denote the conjunction, disjunction, and negation Boolean operators, respectively; $\cdot$ is the concatenation operator; operator H^d where $d \in \mathbb{Z}^+$, represents the hold operator; $[]^{[a,b]}$ denotes the within operator with $a, b \in \mathbb{Z}^+$ and $a \le b$. For example, the statement "stop at location

A for 3 seconds" can be represented as H^3A, and "take the customer to A within 20 minutes, and then pick up food from B within 60 minutes" can be written as $[H^0A]^{[0,20]} \cdot [H^0B]^{[0,60]}$. Detailed syntax and semantics of TWTL can be found in Vasile et al. (2017).

In this paper, we consider temporal relaxations of TWTL specifications that can be encoded as a compact FSA representation. Temporally relaxed TWTL formulas accommodate tasks that may be completed before or after their original deadlines. For instance, a formula $\phi = [H^0A]^{[0,20]} \cdot [H^0B]^{[0,60]}$ can be temporally relaxed as $\phi(\tau) = [H^0A]^{[0,20+\tau_1]} \cdot [H^0B]^{[0,60+\tau_2]}$, where $\tau = (\tau_1, \tau_2)$. Specifically, we consider relaxed formulas $\phi(\tau)$ whose time bound $\|\phi(\tau)\|$ does not exceed that of ϕ (i.e., any delay in achieving tasks must be compensated by the others to ensure the overall duration is not exceeded). The time bound of ϕ is defined as the maximum time needed to satisfy ϕ.

3. Problem Statement

We consider a labeled-Markov Decision Process (MDP) denoted as $\mathscr{M} = (S, A, \Delta_M, R, l)$, where S represents the state space, and A denotes the set of actions. The probabilistic transition function is defined as $\Delta_M : S \times A \times S \to [0,1]$, while $R : S \to \mathbb{R}$ represents the reward function. Additionally, $l : S \to 2^{AP}$ is a labeling function that maps each state to a set of atomic propositions. Given a trajectory $\mathbf{s} = s_1 s_2 \ldots$ over the MDP, the output word $\mathbf{o} = o_1 o_2 \ldots$ is a sequence of elements from 2^{AP}, where each element $o_i = l(s_i)$. The subword $o_i \ldots o_j$ is denoted by $\mathbf{o}_{i,j}$

Definition 2 (Deterministic Policy)**.** *Given a labeled-MDP $\mathscr{M} = (S, A, \Delta_M, R, l)$, a deterministic policy is a mapping $\pi : S \to A$ that maps each state to a single action.*

We address the problem of learning a policy that maximizes the reward while ensuring the satisfaction of a BTL specification with a probability greater than a desired threshold *throughout the learning process*. In our setting, the reward function is decoupled from the BTL specification and captures a secondary objective that may not always align with the primary goal of satisfying the BTL specification. For example, consider a scenario where a robot must complete a pickup-and-delivery task with at least 95% probability, while also being encouraged to monitor high-priority areas to maximize information gain. Such a probabilistic guarantee cannot be achieved without any prior knowledge about the transition probabilities. In this paper, we assume that while the actual transition probabilities may be unknown, for each state s and action a, the agent knows which states have a non-zero probability and which states have a sufficiently large probability of being observed as the next state s'.

Problem 1. *Suppose that the following are given: (1) a labeled-MDP $\mathscr{M} = (S, A, \Delta_M, R, l)$ with unknown transition function Δ_M and reward function R, (2) a BTL formula ϕ with time bound $\|\phi\| = T$, (3) a desired probability threshold $Pr_{des} \in (0,1]$, (4) some $\varepsilon \in [0,1)$ such that for each MDP state s and action a, the states s' for which $\Delta_M(s,a,s') > 0$ and the states s'' for which $\Delta_M(s,a,s'') \geq 1-\varepsilon$ are known, find the optimal policy*

$$\pi^* = \arg\max_{\pi} E^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t] \tag{1}$$

such that, for every episode j in the learning process,

$$\Pr\left(\mathbf{o}_{jT,jT+T} \models \phi(\tau_j)\right) \geq Pr_{des}, \ \|\phi(\tau_j)\| \leq T, \quad \forall j \geq 0 \tag{2}$$

where γ is a discount factor, $\mathbf{o}_{jT,jT+T}$ is the output word in episode j, τ_j is the time relaxation in episode j, $\phi(\tau_j)$ is a temporally-relaxed BTL constraint, and $\|\phi(\tau_j)\|$ is the time bound of $\phi(\tau_j)$.

4. Proposed Algorithm

We propose a solution to Problem 1 by introducing a switching-based algorithm that allows switching between two policies: 1) a stationary policy derived from the product of the MDP and FSA for maximizing the probability of constraint satisfaction based on the available prior information, and 2) a policy learned over the MDP to maximize rewards. Before each episode, the RL agent determines whether to follow the constraint satisfaction policy or the reward maximization policy based on a computed switching probability. The proposed approach, separating constraint satisfaction from reward maximization, eliminates the need for a time-product MDP often used in the state-of-the-art, offering a more computationally efficient solution.

4.1. Policy for Constraint Satisfaction

Consider a task "eventually visit A and then B". Suppose that the agent is at C. The agent must select an action that steers it towards 1) B if A has been visited before; or 2) A if A has not been visited yet. Hence, the selection of actions is determined by the agent's current state and the progress of constraint satisfaction, which can be encoded by a Product MDP.

Definition 3 (Product MDP). *Given a labeled-MDP $\mathscr{M} = (S, A, \Delta_M, R, l)$ and an FSA $\mathscr{A} = (Q, q_{init}, O, \delta, F)$, a product MDP is a tuple $\mathscr{P} = \mathscr{M} \times \mathscr{A} = (S_P, S_{P,init}, A, \Delta_P, R_P, F_P)$, where $S_P = S \times Q$ is a finite set of states; $S_{P,init} = \{(s, \delta(q_{init}, l(s)) \,|\, \forall s \in S\}$ is the set of initial states; A is the set of actions; $\Delta_P : S_P \times A \times S_P \to [0,1]$ is the probabilistic transition relation, such that for any two states, $p = (s,q) \in S_P$ and $p' = (s',q') \in S_P$, and any action $a \in A$, $\Delta_P(p,a,p') = \Delta_M(s,a,s')$ if $\delta(q, l(s)) = q'$; $R_P : S_P \to \mathbb{R}$ is the reward function such that $R_P(p) = R(s)$ for $p = (s,q) \in S_P$; $F_P = (S \times F_A) \subseteq S_P$ is the set of accepting states.*

The constraint satisfaction policy is designed to maximize the probability of reaching F_P from any state of the product MDP. We can obtain such a policy by selecting the action to minimize the expected distance to F_P from each state. Thus, we define ε-stochastic transitions and distance-to-F_P.

Definition 4 (ε-Stochastic Transitions). *Given a product MDP and some $\varepsilon \in [0,1)$, any transition (p_i, a, p_j) such that $\Delta_P(p_i, a, p_j) \geq 1 - \varepsilon$ is defined as an ε-stochastic transition.*

If transition (p_i, a, p_j) is a 0-stochastic transition, the agent will move to state p_j with probability 1 after taking action a at state p_i. Oppositely, as ε approaches 1, any feasible transition becomes a ε-stochastic transition. Next, we will use ε-stochastic transitions to define *Distance-To-F_P*.

Definition 5 (Distance-To-F_P). *Given a product MDP, for any product MDP state p, the distance from p to the set of accepting states F_P is $D^\varepsilon(p) = \min_{p' \in F_P} dist^\varepsilon(p, p')$ where $dist^\varepsilon(p, p')$ represents the minimum number of ε-stochastic transitions to move from state p to state p'.*

The distance-to-F_P, $D^\varepsilon(p)$, represents the minimum number of ε-stochastic transitions needed for reaching the set of accepting states from a state p. We will use this metric to design a policy for constraint satisfaction (reaching the accepting states) and derive a lower bound on the probability of constraint satisfaction within a time bound.

Definition 6 (π^ε_{GO} Policy). *Given a product MDP and $\varepsilon \in [0,1)$, $\pi^\varepsilon_{GO} : S_P \to A$, is a stationary policy over the product MDP such that*

$$\pi^\varepsilon_{GO}(p) = \arg\min_{a \in A} D^\varepsilon_{min}(p, a), \tag{3}$$

where $D^{\varepsilon}_{min}(p,a) = \min_{p':\Delta_P(p,a,p')\geq 1-\varepsilon} D^{\varepsilon}(p')$*, i.e., the minimum distance-to-*F_P *among the states reachable from* p *under action* a *with probability of at least* $1-\varepsilon$*.*

In order to achieve probabilistic constraint satisfaction in each episode, our approach builds a conservative estimation of the probability of reaching an accepting state under the π^{ε}_{GO} policy for every initial state. In particular, we present two methods for computing a lower bound on this satisfaction probability. *The first method* involves a closed-form equation outlined in Theorem 1. To derive this equation, we introduce an additional assumption on the product-MDP. *The second method*, which relaxes this assumption, utilizes a recursive approach.

Assumption 1. *The MDP and FSA (constraint) are such that any feasible transition in the resultant product MDP cannot increase the distance-to-*F_P *by more than some* $\delta_{max} \in \mathbb{Z}^+$*.*

Assumption 1 is the relaxed version of the assumption made in Aksaray et al. (2021), which only allows $\delta_{max} = 1$. This relaxed assumption poses fewer constraints on the BTL formulas that can be handled by our proposed approach. For example, we can accommodate formulas such as "eventually stay at A for 3 time steps". The distance-to-F_P can increase by 3 if the robot leaves A right before it stays at A for 3 time steps.

Theorem 1. *Let Assumption 1 hold. For any* $p \in S_P$ *of the given product MDP* $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$*, let integer* $k > 0$ *denote the remaining time steps,* $d = D^{\varepsilon}(p)$ *denote the distance-to-*F_P *from* p*, and* $Pr(p \xrightarrow{k} F_P; \pi^{\varepsilon}_{GO})$ *be the probability of reaching* F_P *from* p *within the next* k *time steps under the policy* π^{ε}_{GO}*. If* $0 < d < \infty$*, then*

$$Pr(p \xrightarrow{k} F_P; \pi^{\varepsilon}_{GO}) \geq lb^c[p][k], \tag{4}$$

where

$$lb^c[p][k] = \sum_{m=1}^{k} P(T_m), \quad P(T_m) = \left[C_m^{\frac{m-d}{1+\delta_{max}}} \varepsilon^{\frac{m-d}{1+\delta_{max}}} (1-\varepsilon)^{\frac{m\delta_{max}+d}{1+\delta_{max}}} - \sum_{m'=1}^{m-1} C_{m-m'}^{\frac{m-m'}{1+\delta_{max}}} \varepsilon^{\frac{m-m'}{1+\delta_{max}}} (1-\varepsilon)^{\frac{(m-m')\delta_{max}}{1+\delta_{max}}} P(T_{m'}) \right],$$

$$C_m^n = \begin{cases} 0 & \text{if } n > m \text{ or } n \notin \mathbb{Z}^+ \\ \frac{m!}{n!(m-n)!} & \text{otherwise.} \end{cases}, \quad T_m = \{(X_1,\ldots,X_k) | \sum_{i=1}^{m} X_i = d, \text{ and } \sum_{i=1}^{m'} X_i < d, \forall m' < m\}.$$

X_i *is a random variable representing the distance reduction to* F_p *at time step* i*, which takes the value 1 with a probability of* $1-\varepsilon$*, and* $-\delta_{max}$ *with a probability of* ε*.*

Corollary 1.1. *Let Assumption 1 hold. For any initial state* $p_0 \in P_{init}$ *such that* $0 < D^{\varepsilon}(p_0) < \infty$ *and* $lb^c[p_0][T] \geq Pr_{des}$*,*

$$Pr(p_0 \xrightarrow{T} F_P; \pi^{\varepsilon}_{GO}) \geq Pr_{des}, \tag{5}$$

where T *is the time bound of the BTL constraint.*

Proof The proof of Theorem 1 and Corollary 1.1 is provided in Lin et al. (2024).

While the lower bound lb^c can be efficiently computed, it can be overly conservative due to the assumptions made in deriving (4). To address this limitation, we employ a method proposed in Lin et al. (2023) and present Alg. 1, which computes another lower bound based on recursive computations over the product MDP. While this recursive approach is computationally more demanding, it provides a much less conservative lower bound than (4).

Algorithm 1: Recursive algorithm for computing the proposed lower bound in (7)

Input : product MDP $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$; time bound $\|\phi\| = T$
Output: lower bound of satisfaction probability $lb^r[\cdot][\cdot]$

1 **for** k = 0, 1, ..., T **do**
2   **for** *each* $p \in S_P$ **do**
3   **if** k = 0 **then**
4             $lb^r[k][p] \leftarrow 1$ **if** *p is an accepting state*, **else** $lb^r[k][p] \leftarrow 0$
5   **else if** $D^{\varepsilon}(p) > k$ **then** $lb^r[k][p] \leftarrow 0$
6   **else if** *p is accepting state* **then** $lb^r[k][p] \leftarrow 1$
7   **else**
8             $lb^r[k][p] \leftarrow$ solve (6)

$$
\begin{aligned}
lb^r[p][k] = & \min_{\Delta_1, \Delta_2, \ldots, \Delta_n} \sum_{i=1}^{n} lb^r[p_i'][k-1]\Delta_i && \text{(6a)}\\
\text{s.t.} \quad & \sum_{i=1}^{n} \Delta_i = 1 && \text{(6b)}\\
& 1 - \varepsilon \leq \Delta_j \leq 1, \quad j = 1, 2, \ldots, m \qquad (\varepsilon-\text{stochastic transitions}) && \text{(6c)}\\
& 0 < \Delta_k \leq \varepsilon, \quad k = m+1, \ldots, n \qquad (\text{transitions that are not } \varepsilon-\text{stochastic transitions}). && \text{(6d)}
\end{aligned}
$$

Alg. 1 computes $lb^r[p][k]$, the lower bound of the satisfaction probability from any product automaton state p within k time steps, under policy π^{ε}_{GO}. The algorithm leverages the fact that the lower bound $lb^r[p][k]$ depends on $lb^r[p'][k-1]$, where p' represents the reachable states from p under π^{ε}_{GO}. If the values of $lb^r[p'][k-1]$ are known, we can compute $lb^r[p][k]$, by solving the optimization problem in (6). Starting from $k = 0$ (lines 3-4), we can iteratively solve the optimization problem for $lb^r[p][k]$ up to $k = T$ (lines 5-8).

Theorem 2. *For any $p \in S_P$ of a given product MDP $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$, let integer $k > 0$ denote the remaining time steps, and $Pr(p \xrightarrow{k} F_P; \pi^{\varepsilon}_{GO})$ be the probability of reaching the set of accepting states from p within the next k time steps under the policy π^{ε}_{GO}. Then*

$$Pr(p \xrightarrow{k} F_P; \pi^{\varepsilon}_{GO}) \geq lb^r[p][k]. \tag{7}$$

Corollary 2.1. *Given a product MDP $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$, any initial state $p_0 \in P_{init}$ such that $lb^r[p_0][T] \geq Pr_{des}$ satisfies*

$$Pr(p_0 \xrightarrow{T} F_P; \pi^{\varepsilon}_{GO}) \geq Pr_{des}. \tag{8}$$

Proof The proof of Theorem 2 and Corollary 2.1 is provided in Lin et al. (2024).

4.2. A Switching-based RL Algorithm

The switching algorithm allows a probabilistic transition between two distinct policies: the π^{ε}_{GO} policy for constraint satisfaction and a learned policy for reward maximization. We hereby define the switching policy as follows.

Definition 7 (Switching Policy)**.** *For any episode starting with initial state $p \in P_{init}$, the switching policy is defined as adopting π^{ε}_{GO} policy with a probability of $Pr_{switch}(p)$ and RL with a probability of $1 - Pr_{switch}(p)$ throughout that episode, where $Pr_{switch}(p)$ is the switching probability for state p.*

To determine the switching probability $Pr_{switch}(p)$ for each initial state p, we initialize $Pr_{switch}(p)$ to 1 so that the agent adopts π_{GO}^{ε} policy with probability 1 in the early stage of the learning process. Let $Pr_{GO}(p)$ denote the probability of satisfaction under policy π_{GO}^{ε} starting from an initial state p. By estimating $Pr_{GO}(p)$, we can adjust the switching probability such that: 1) $Pr_{switch}(p)$ is lower than 1 (to allow exploration for reward maximization) if we are confident that $Pr_{GO}(p)$ is greater than the desired threshold Pr_{des}; 2) $Pr_{switch}(p)$ remains 1 (to maximize constraint satisfaction) if we are not confident that $Pr_{GO}(p)$ is greater than Pr_{des}. Since π_{GO}^{ε} is a stationary policy, for any initial state p, the outcome of following π_{GO}^{ε} (either satisfies or violates the constraint) is a Bernoulli trial with the probability of success (constraint satisfaction) equal to $Pr_{GO}(p)$. Thus, we use Wilson score interval Wilson (1927), to compute a confidence bound $[Pr_{low}(p), Pr_{up}(p)]$ that contains $Pr_{GO}(p)$ up to some given confidence level, where

$$Pr_{low}(p) = \frac{n_S(p) + \frac{1}{2}z^2}{n(p) + z^2} - \frac{z}{n(p) + z^2}\sqrt{\frac{n_S(p)n_F(p)}{n(p)} + \frac{z^2}{4}}. \tag{9}$$

Here, $n(p)$ denotes the total number of episodes the agent started at p and adopted π_{GO}^{ε}, $n_S(p)$ is the number of those episodes that satisfied the constraint under π_{GO}^{ε}, $n_F(p)$ is the number of episodes that violated the constraint under π_{GO}^{ε}, i.e., $n(p) = n_F(p) + n_S(p)$. The value of z is determined by the desired confidence level (e.g., 99% confidence level corresponds to a z value of 2.58), i.e, the probability that $[Pr_{low}(p), Pr_{up}(p)]$ contains $Pr_{GO}(p)$. Accordingly, we can select a sufficiently high value of z to ensure that $Pr_{GO}(p) \geq Pr_{low}(p)$ with high confidence. We update the lower bound $Pr_{low}(p)$ at the end of each episode. The resulting $Pr_{low}(p)$ is then used to update the switching probability $Pr_{switch}(p)$ of the initial state p.

If $Pr_{low}(p)$ is less than the desired threshold Pr_{des} (indicating a risk of violating the constraint), the switching probability should be set to 1 to ensure that the algorithm always employs π_{GO}^{ε} when starting at the initial state p. If $Pr_{low}(p)$ exceeds Pr_{des}, indicating a high likelihood of constraint satisfaction by π_{GO}^{ε} policy, the switching probability can be set lower than 1. This adjustment allows for executing RL with a non-zero probability to enhance reward maximization. The estimation of the satisfaction probability becomes more accurate as the number of episodes increases, and the algorithm reduces the switching probability as needed to achieve reward maximization.

Note that in any episode starting at p, the probability of satisfying the constraint is lower bounded by $Pr_{switch}(p)Pr_{GO}(p)$, i.e., the probability of choosing π_{GO}^{ε} in that episode times the probability of satisfying the constraint from that initial state via π_{GO}^{ε}. Accordingly, we propose to update the switching probability as

$$Pr_{switch}(p) = min\left(1, \frac{Pr_{des}}{Pr_{low}(p)}\right), \tag{10}$$

which ensures that the product, $Pr_{switch}(p)Pr_{GO}(p)$, a lower bound on the probability of satisfying the constraint starting at p, is at least Pr_{des} as long as 1) $Pr_{GO}(p) \geq Pr_{des}$, and 2) $Pr_{GO}(p) \geq Pr_{low}(p)$, which can be achieved with very high confidence via a proper selection of z in (9).

At the beginning of each episode, the agent decides whether to adhere to the π_{GO}^{ε} policy for constraint satisfaction or to employ RL for maximizing rewards. This decision is based on the calculated switching probability $Pr_{switch}(\cdot)$. The design of Pr_{switch} in (10) ensures that: 1) the agent exclusively follows the π_{GO}^{ε} policy when the confidence lower bound Pr_{low} is lower than the desired threshold Pr_{des}; 2) the agent is allowed to engage in RL for reward maximization when Pr_{low} exceeds Pr_{des} (as presented in Alg. 2).

Algorithm 2: Switching-based RL

Input : product MDP $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$; initial MDP state s_{init}; π^{ε}_{GO} policy; time bound of BTL constraint ϕ, i.e., T

Output: $\pi : S_P \rightarrow A$; $Pr_{switch}(\cdot)$

1 **Initialization:** $n(p) \leftarrow 0, n_S(p) \leftarrow 0, n_F(p) \leftarrow 0$ for all $p \in P_{init}$

2 **Initialization:** $p \leftarrow$ *find* $\bar{p} \in P_{init}$ *s.t.* $mdp\_state(\bar{p}) = s_{init}$

3 **for** $j = 0 : N_{episode} - 1$ **do**

4     $p_0 \leftarrow p$

5   **if** $n(p_0) < N_{sample}$ *or random()* $< Pr_{switch}(p_0)$ **then** $flag_{RL} \leftarrow 0$

6   **else** $flag_{RL} \leftarrow 1$

7   **for** $t = 0 : T - 1$ **do**

8   **if** ϕ *is not satisfied* ***and*** $flag_{RL} = 0$ **then**

9   Action $a \leftarrow \pi^{\varepsilon}_{GO}(p)$

10   Take action a, observe the next state p'

11   **else if** *constraint satisfied* ***or*** $flag_{RL} = 1$ **then**

12   Update π via a selected RL algorithm

13   **if** ϕ *is satisfied* **then**

14   `update`$(p_0, n(p_0), n_S(p_0), n_F(p_0),$ '*success*'$)$

15   **else**

16   `update`$(p_0, n(p_0), n_S(p_0), n_F(p_0),$ '*failure*'$)$

17 **Function** `update`$(p_0, n(p_0), n_S(p_0), n_F(p_0), result)$ **:**

18     $n(p_0) \leftarrow n(p_0) + 1$

19     $n_S(p_0) \leftarrow n_S(p_0) + 1$ **if** *result* = 'success', **else** $n_F(p_0) \leftarrow n_F(p_0) + 1$

20     $Pr_{low}(p_0) \leftarrow$ equation (9)

21     $Pr_{switch}(p_0) \leftarrow$ equation (10)

Algorithm 2 begins by initializing the numbers of trials, successes, and failures for every initial state in P_{init} (line 1). Line 2 sets the initial product MDP state. Before each episode, the algorithm records the initial state of the current episode (line 4) and determines whether to follow the π^{ε}_{GO} policy or to adopt RL (lines 5-6). In line 5, the condition $n(p_0) < N_{sample}$ is included to ensure enough samples have been collected for accurate estimation of the confidence lower bound. In situations where π^{ε}_{GO} policy is selected but the constraint has not yet been satisfied, the agent will take actions from the π^{ε}_{GO} policy (lines 8-10). Conversely, if the constraint has been satisfied or RL is selected, the agent uses a chosen RL algorithm to update the reward maximization policy π (line 12). Some examples of RL algorithms that can be used in line 12 include Tabular Q-learning Watkins and Dayan (1992) and Deep Q-learning Mnih (2013). At the end of each episode, the algorithm will check if the BTL constraint is satisfied, and update the numbers of trials, successes, failures, and the switching probability for the initial state p_0 (lines 13-16), using the function `update()`.

Theorem 3. *Given a BTL constraint ϕ with a desired threshold Pr_{des} and a product MDP $\mathscr{P} = (S_P, P_{init}, A, \Delta_P, R_P, F_P)$, if $Pr(p_0 \xrightarrow{T} F_P; \pi^{\varepsilon}_{GO}) \geq Pr_{des}$ [1] for every initial state $p_0 \in P_{init}$, then Alg. 2 guarantees that the probability of satisfying ϕ in each episode is at least Pr_{des} with high confidence[2].*

Proof The proof of this theorem is provided in Lin et al. (2024).

5. Simulation Results

We present case studies to validate the proposed algorithm and compare it with Aksaray et al. (2021), which serves as a baseline. We consider a robot operating on an 8×8 grid, with its action set and possible transitions under each action shown in Fig. 1. The action "Stay" keeps the robot at its current position with probability 1, while any other action leads to the intended transition with 90% probability and distributes the remaining 10% equally among all unintended transitions. While this exact transition model is unknown, the robot has partial knowledge through a conservative uncertainty $\varepsilon \geq 0.1$, where the actual transition uncertainty is 0.1.

1. In practice, the proposed bounds, lb^r or lb^c, can be used to verify this inequality as shown in Corollaries 1.1 and 2.1.
2. Theorem 3 does not claim that the probability of satisfaction is always greater than or equal to Pr_{des}. Instead, we ensure this probabilistic satisfaction guarantee with high confidence. This is because Pr_{low} was estimated using the Wilson score method, which means that $Pr_{GO}(p_0) \geq Pr_{low}(p_0)$ (needed in the proof for Theorem 3) holds true with a high confidence level depending on the chosen parameter z in (9).

We consider a scenario where the robot periodically performs a pickup and delivery task while being incentivized to remain in high-reward regions for information gathering. In Fig. 2, the light gray, dark gray, and all other cells yield a reward of 1, 10, and 0, indicating their levels of importance. The pickup and delivery task is formalized using a TWTL formula: $[H^1 P]^{[0,20]} \cdot ([H^1 D_1]^{[0,20]} \vee [H^1 D_2]^{[0,20]}) \cdot [H^1 Base]^{[0,20]}$.

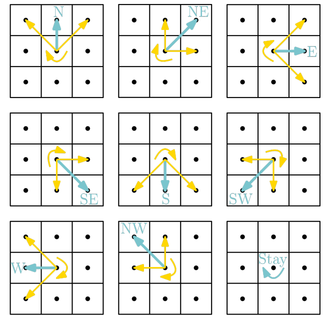


Figure 1: Transitions (intended - blue, unintended - yellow) under each action.

Case 1. We illustrate sample trajectories in Fig. 2 using the policy learned by the proposed algorithm (with tabular-Q learning). The proposed algorithm adaptively switches between two decoupled behaviors—constraint satisfaction and reward maximization—and also learns the switching probability between them. For example, we run 1000 episodes using the proposed algorithm, which outputs the reward maximizing policy (blue in Fig. 2) and the switching probability of 0.907 between the reward maximizing and constraint satisfying (black in Fig. 2) policies, for $Pr_{des} = 0.9$, $\varepsilon = 0.2$.

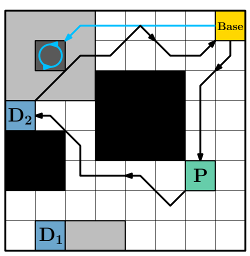


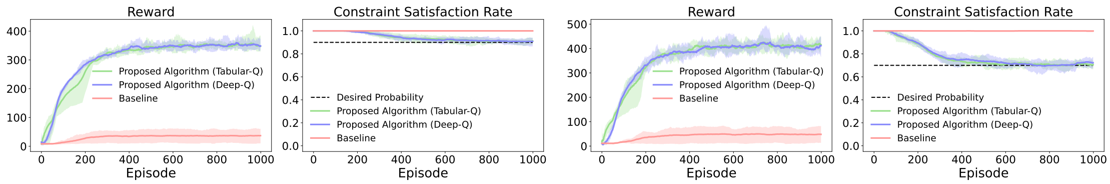
Figure 2: An environment where yellow, green, blue, and black cells are respectively the base station (start state), the pick-up region, the delivery regions, and the obstacles. The gray cells are reward regions (darker shades - higher reward). The arrows denote sample trajectories by the proposed algorithm (blue: reward maximization policy π, black: π^{ε}_{GO} policy).

In Cases 2 and 3, we compare the performance of the proposed algorithm during training with respect to a baseline algorithm. In particular, the baseline is the algorithm in Aksaray et al. (2021), which learns a single static policy that jointly maximizes reward and ensures constraint satisfaction in every episode. For the benchmark analysis, we use a fixed number of episodes ($N_{episode} = 1000$) and a decaying ε-greedy RL policy ($\varepsilon_{init} = 0.7$ and $\varepsilon_{final} = 0.0001$). The learning rate and discount factor are set to 0.1 and 0.95, respectively. We use a z-score of 2.58 to ensure the probabilistic constraint satisfaction with high confidence. All results are averaged over 10 independent runs.

Case 2. We evaluated the proposed algorithm (with tabular and deep Q-learning) and the baseline under varying probability thresholds Pr_{des}. As shown in Fig. 3, the baseline tends to be overcautious and enforces a higher satisfaction rate, while our proposed algorithm effectively balances constraint satisfaction with reward maximization, adaptively aligning the satisfaction rate with the desired threshold. As Pr_{des} increases, we notice a decrease in the collected rewards in both algorithms, due to a more restrictive constraint.

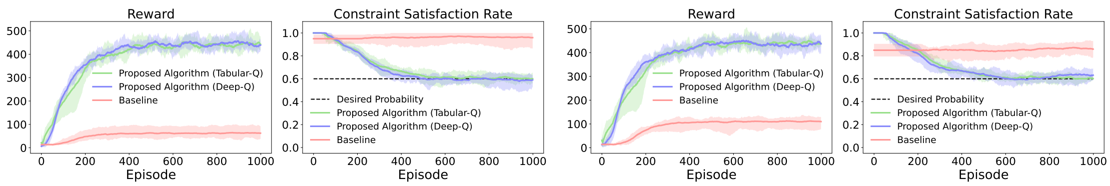
Case 3. This case study examines the impact of the parameter ε on the performance of both algorithms. As shown in Fig. 4, the baseline is highly sensitive to ε; larger values reduce reward collection but increase satisfaction rate. In contrast, the proposed algorithm adapts its switching behavior based on collected data and is independent of ε. As a result, it remains largely unaffected by changes in ε.

The proposed algorithm consistently outperforms the baseline in maximizing rewards for two main reasons: 1) The proposed solution separates reward maximization from constraint satisfaction, allowing the agent to learn the reward-maximizing policy on a significantly smaller MDP, whereas the baseline operates on a much larger time-product MDP. 2) While the baseline tends to over-satisfy the TWTL constraint, the proposed solution adaptively aligns with the desired prob-



$(a)\ Pr_{des} = 0.9, \varepsilon = 0.2$ $(b)\ Pr_{des} = 0.7, \varepsilon = 0.2$

Figure 3: Reward and constraint satisfaction rate under various desired probabilities Pr_{des}.



$(a)\ Pr_{des} = 0.6, \varepsilon = 0.2$ $(b)\ Pr_{des} = 0.6, \varepsilon = 0.1$

Figure 4: Reward and constraint satisfaction rate under different uncertainties ε.

ability by adjusting the switching probability, providing more freedom to explore and maximize the rewards. In general, the proposed algorithm is also more efficient, requiring significantly fewer training episodes to converge compared to the baseline.

Case 4. We compare the closed-form solution (4) with the recursive one (Alg. 1) in terms of lower bound evaluation and computational efficiency. In Table 1, we present the computed lower bounds for selected product MDP states p at time step $k = 17$ for a TWTL task $[H^1 P]^{[0,8]} \cdot [H^1 D_1]^{[0,8]}$. The results indicate that the recursive solution consistently produces higher (less conservative) lower bounds than the closed-form solution. On the other hand, the closed-form solution is more computationally efficient than the recursive solution. For example, for task $[H^1 P]^{[0,20]} \cdot ([H^1 D_1]^{[0,20]} \vee [H^1 D_2]^{[0,20]}) \cdot [H^1 Base]^{[0,20]}$, the recursive solution requires 21.14 seconds to compute $lb^r[p][k]$ for all initial product MDP states p at a time step $k = 62$, while the closed-form solution only requires 0.089 seconds to compute $lb^c[p][k]$. A detailed scalability analysis of the recursive method for larger state-action spaces and more complex temporal logic constraints is provided in Lin et al. (2024).

k	p	$lb^c[k][p]$ by (4)	$lb^r[k][p]$ by Alg. 1
$k = 17$	(P, q_0)	0.814	0.988
	$(Base, q_0)$	0.359	0.798

Table 1: Lower bounds for the task $[H^1 P]^{[0,8]} \cdot [H^1 D_1]^{[0,8]}$

6. Conclusion

We propose a switching-based algorithm for learning policies to optimize a reward function while ensuring the satisfaction of the BTL constraint with a probability greater than a desired threshold throughout the learning process. Our approach uniquely combines a stationary policy for ensuring constraint satisfaction and an RL policy for reward maximization. Utilizing the Wilson score method, we effectively estimate the satisfaction rate's confidence interval, thereby adaptively adjusting the switching probability between the two policies. This method achieves a desired trade-off between constraint satisfaction and reward collection. Simulation results have demonstrated the algorithm's efficacy, showing improved performance over existing methods.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta. Q-learning for robust satisfaction of signal temporal logic specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 6565–6570. IEEE, 2016.

Derya Aksaray, Yasin Yazıcıoğlu, and Ahmet Semi Asarkaya. Probabilistically guaranteed satisfaction of temporal logic constraints during reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6531–6537, 2021.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Mingyu Cai and Cristian-Ioan Vasile. Safe-critical modular deep reinforcement learning with temporal logic through gaussian processes and control barrier functions. *arXiv preprint arXiv:2109.02791*, 2021.

Antonio Cau and Hussein Zedan. Refining interval temporal logic specifications. In *International AMAST Workshop on Aspects of Real-Time Systems and Concurrent and Distributed Software*, pages 79–94. Springer, 1997.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Mohammadhosein Hasanbeig, Alessandro Abate, and Daniel Kroening. *Cautious Reinforcement Learning with Logical Constraints*, page 483–491. International Foundation for Autonomous Agents and Multiagent Systems, 2020. ISBN 9781450375184.

Nils Jansen, Bettina Könighofer, Sebastian Junges, AC Serban, and Roderick Bloem. Safe reinforcement learning using probabilistic shields. 2020.

Xiao Li, Cristian-Ioan Vasile, and Calin Belta. Reinforcement learning with temporal logic rewards. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3834–3839, 2017.

Xiaoshan Lin, Abbasali Koochakzadeh, Yasin Yazıcıoğlu, and Derya Aksaray. Reinforcement learning under probabilistic spatio-temporal constraints with time windows. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8680–8686, 2023.

Xiaoshan Lin, Sadık Bera Yüksel, Yasin Yazıcıoğlu, and Derya Aksaray. Probabilistic satisfaction of temporal logic constraints in reinforcement learning via adaptive policy-switching. *arXiv preprint arXiv:2410.08022*, 2024.

Volodymyr Mnih. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Cristian-Ioan Vasile, Derya Aksaray, and Calin Belta. Time window temporal logic. *Theoretical Computer Science*, 691:27–54, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Paolo Zuliani, André Platzer, and Edmund M Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, pages 243–252, 2010.