Q-Learning for Robust Satisfaction of Signal Temporal Logic Specifications

Derya Aksaray, Austin Jones, Zhaodan Kong, Mac Schwager, and Calin Belta

***Abstract*— In this paper, we address the problem of learning optimal policies for satisfying signal temporal logic (STL) specifications by agents with unknown stochastic dynamics. The system is modeled as a Markov decision process, in which the states represent partitions of a continuous space and the transition probabilities are unknown. We formulate two synthesis problems where the desired STL specification is enforced by maximizing 1) the probability of satisfaction, and 2) the expected robustness degree, i.e., a measure quantifying the quality of satisfaction. We discuss that Q-learning is not directly applicable to these problems because, based on the quantitative semantics of STL, the probability of satisfaction and expected robustness degree are not in the standard objective form of Q-learning (i.e., the sum of instantaneous rewards). To resolve this issue, we propose an approximation of STL synthesis problems that can be solved via Q-learning, and we derive some performance bounds for the policies obtained by the approximate approach. Finally, we present simulation results to demonstrate the performance of the proposed method.**

I. Introduction

In this paper, we address the problem of controlling a system with unknown, stochastic dynamics to achieve a complex, time-sensitive task. An example is controlling a noisy aerial vehicle with partially known dynamics to visit a pre-specified set of regions in some desired order while avoiding hazardous areas. We consider tasks given in terms of temporal logic (TL) [2], an extension of first order Boolean logic that can be used to reason about how the state of a system evolves over time. When a stochastic dynamical model is known, there exist algorithms to find control policies for maximizing the probability of achieving a given TL specification (e.g., [21], [19], [17]) by planning over stochastic abstractions (e.g., [16], [1], [19]). However, only a handful of papers have considered the problem of enforcing TL specifications to a system with unknown dynamics. For example, reinforcement learning has been used to find a policy that maximizes the probability of satisfying a given linear temporal logic (LTL) formula in [3], [23], [11].

In contrast to existing works on reinforcement learning using propositional temporal logic, we consider signal temporal logic (STL), a rich predicate logic that can be used to describe tasks involving bounds on physical parameters and time intervals [8]. An example STL specification is "Within t_1 seconds, a region in which y is less than p_1 is reached, and regions in which y is larger than p_2 are avoided for t_2 seconds." STL is also endowed with a metric called *robustness degree* that quantifies how strongly a given trajectory satisfies an STL formula as a real number rather than just providing a *yes* or *no* answer [10], [8]. This measure enables the use of continuous optimization methods to solve inference (e.g., [14], [15], [18]) or formal synthesis problems (e.g., [22]) involving STL.

In this paper, we formulate two problems that enforce a desired STL specification by maximizing 1) the probability of satisfaction and 2) the expected robustness degree. One of the difficulties in solving these problems is the history-dependence of the satisfaction. For instance, if the specification requires visiting region A before region B, whether or not the system should move towards region B depends on whether or not it has previously visited region A. For LTL formulae with time-abstract semantics, this history-dependence can be broken by translating the formula to a deterministic Rabin automaton, i.e., a model that automatically takes care of the history-dependent "book-keeping", e.g., [23]. In the case of STL, such a construction is difficult due to the time-bounded semantics. We circumvent this problem by defining a fragment of STL such that the progress towards satisfaction is checked with a sufficient number of (i.e., τ) state measurements. We thus define a Markov decision process (MDP), called the τ-MDP, whose states correspond to the τ-step history of the system and the actions are from a finite set of motion primitives.

Even though the history dependence issue can be solved by defining a τ-MDP, a reinforcement learning strategy such as Q-learning [27] is still not applicable to maximize probability of satisfaction or expected robustness degree. In Q-learning, an agent tries an action, observes an immediate reward, and updates its policy to maximize the sum of (discounted) rewards. However, based on the quantitative semantics of STL, the objective functions such as probability of satisfaction or expected robustness degree are not in the standard form of Q-learning. Thus, we propose an approximation of these objective functions such that the new synthesis problems can be solved via Q-learning. Moreover, we provide some performance bounds for the approximate solutions, which can be sufficiently close to the actual solutions with a proper selection of the approximation parameter. Finally, we demonstrate the performance of the proposed approach through simulation case studies.

*This work was partially supported at Boston University by ONR grant number N00014-14-1-0554 and by the NSF grant numbers CMMI-1400167, NSF NRI-1426907. D. Aksaray and C. Belta are with the Department of Mechanical Engineering, Boston University, Boston, MA, USA. {daksaray,cbelta}@bu.edu. A. Jones is with the Departments of Mechanical Engineering and Electrical Engineering, Georgia Institute of Technology, Atlanta, GA, USA. austinjones@gatech.edu. Z. Kong is with the Department of Mechanical and Aerospace Engineering, University of California Davis, Davis, CA, USA. zdkong@ucdavis.edu. M. Schwager is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA. schwager@stanford.edu.

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

II. Preliminaries: Signal Temporal Logic

In this paper, the desired system behavior is described by a signal temporal logic (STL) fragment with the following *syntax* :

$$\varphi \quad := f(\mathbf{s}) < d \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, F_{[a,b]}\varphi \,|\, G_{[a,b]}\varphi, \qquad (1)$$

where $a, b \in \mathbb{R}_{\geq 0}$ are finite non-negative time bounds; φ is an STL formula; and $f(\mathbf{s}) < d$ is a predicate where $\mathbf{s} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is a signal, $f : \mathbb{R}^n \to \mathbb{R}$ is a function, and $d \in \mathbb{R}$ is a constant. The Boolean operators $\neg$, $\wedge$, and $\vee$ are negation, conjunction (i.e., *and*), and disjunction (i.e., *or*), respectively. The temporal operators F and G stand for *Finally* (i.e., eventually) and *Globally* (i.e., always), respectively.

For any signal $\mathbf{s}$, let s_t denote the value of $\mathbf{s}$ at time t and $(\mathbf{s},t) = s_t s_{t+1} s_{t+2} \ldots$ be the part of the signal that starts at t. Accordingly, the *Boolean semantics* of STL is recursively defined as follows:

$$\begin{array}{lcl}
(\mathbf{s},t) \models (f(\mathbf{s}) < d) & \Leftrightarrow & f(s_t) < d, \\
(\mathbf{s},t) \models \neg(f(\mathbf{s}) < d) & \Leftrightarrow & \neg\big((\mathbf{s},t) \models (f(\mathbf{s}) < d)\big), \\
(\mathbf{s},t) \models \phi_1 \wedge \phi_2 & \Leftrightarrow & (\mathbf{s},t) \models \phi_1 \text{ and } (\mathbf{s},t) \models \phi_2, \\
(\mathbf{s},t) \models \phi_1 \vee \phi_2 & \Leftrightarrow & (\mathbf{s},t) \models \phi_1 \text{ or } (\mathbf{s},t) \models \phi_2, \\
(\mathbf{s},t) \models G_{[a,b]}\phi & \Leftrightarrow & (\mathbf{s},t') \models \phi \quad \forall t' \in [t+a, t+b], \\
(\mathbf{s},t) \models F_{[a,b]}\phi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ s.t. } (\mathbf{s},t') \models \phi.
\end{array}$$

For a signal $(\mathbf{s},0)$, satisfying $F_{[a,b]}\phi$ means that "there exists a time instant within $[a,b]$ such that ϕ will eventually be true", and satisfying $G_{[a,b]}\phi$ means that "ϕ is true for all times between $[a,b]$".

STL is endowed with a metric called *robustness degree* [10], [8] (also called "degree of satisfaction") that quantifies how well a given signal $\mathbf{s}$ satisfies a given formula Φ. The robustness degree is calculated recursively according to the *quantitative semantics:*

$$\begin{array}{rcl}
r(\mathbf{s}, (f(\mathbf{s}) < d), t) & = & d - f(s_t), \\
r(\mathbf{s}, \neg(f(\mathbf{s}) < d), t) & = & -r(\mathbf{s}, (f(\mathbf{s}) < d), t), \\
r(\mathbf{s}, \phi_1 \wedge \phi_2, t) & = & \min\big(r(\mathbf{s}, \phi_1, t), r(\mathbf{s}, \phi_2, t)\big), \\
r(\mathbf{s}, \phi_1 \vee \phi_2, t) & = & \max\big(r(\mathbf{s}, \phi_1, t), r(\mathbf{s}, \phi_2, t)\big), \\
r(\mathbf{s}, G_{[a,b]}\phi, t) & = & \min\limits_{t' \in [t+a, t+b]} r(\mathbf{s}, \phi, t'), \\
r(\mathbf{s}, F_{[a,b]}\phi, t) & = & \max\limits_{t' \in [t+a, t+b]} r(\mathbf{s}, \phi, t').
\end{array}$$

As a short-hand notation, $r(\mathbf{s},\phi)$ refers to $r(\mathbf{s},\phi,0)$ throughout the paper. Let ε-perturbation be a sequence of disturbances such that any signal under ε-perturbation stays inside the ε-envelope. Note that $r(\mathbf{s},\phi) = \varepsilon \geq 0$ means that $\mathbf{s}$ satisfies ϕ. Moreover, the signal $\mathbf{s}$ under ε-perturbation still satisfies ϕ. Similarly, $r(\mathbf{s},\phi) = \varepsilon < 0$ means that $\mathbf{s}$ violates ϕ, and $\mathbf{s}$ under ε-perturbation still violates ϕ.

As in [7], let $hrz(\phi)$ denote the *horizon length* of an STL formula ϕ, which is the required number of samples to resolve any (future or past) requirements of ϕ. The horizon length can be computed recursively as

$$\begin{array}{rcl}
hrz(\psi) & = & 0, \\
hrz(\phi) & = & b \qquad \text{if } \phi = G_{[a,b]}\psi \text{ or } F_{[a,b]}\psi, \\
hrz(\phi) & = & b+d \text{ if } \phi = G_{[a,b]}F_{[c,d]}\psi \text{ or } F_{[a,b]}G_{[c,d]}\psi, \\
hrz(\neg\phi) & = & hrz(\phi), \\
hrz(\phi_1 \wedge \phi_2) & = & \max(hrz(\phi_1), hrz(\phi_2)), \\
hrz(\phi_1 \vee \phi_2) & = & \max(hrz(\phi_1), hrz(\phi_2)),
\end{array}$$

where $a,b,c,d \in \mathbb{R}_{\geq 0}$, ψ is a predicate, and ϕ, ϕ_1, ϕ_2 are STL formulae.

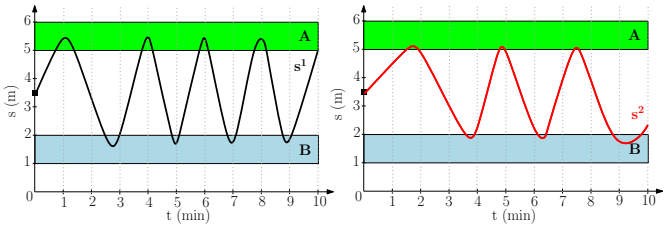


Fig. 1. The specification is "visit regions A and B every 3 minutes along a mission horizon of 10 minutes", i.e., $\Phi = G_{[0,7]}(F_{[0,3]}(s > 5 \wedge s < 6) \wedge F_{[0,3]}(s > 1 \wedge s < 2))$, which is satisfied by signal $\mathbf{s^1}$ and violated by signal $\mathbf{s^2}$.

Example 1: Consider the regions A and B illustrated in Figure 1 and a specification as "visit regions A and B every 3 minutes along a mission horizon of 10 minutes". Note that the desired specification can be formulated in STL as

$$\begin{array}{ll}
\Phi = & G_{[0,7]}\phi \\
\phi = & F_{[0,3]}(s > 5 \wedge s < 6) \wedge F_{[0,3]}(s > 1 \wedge s < 2).
\end{array} \qquad (2)$$

The horizon lengths of Φ and ϕ are $hrz(\Phi) = 10$ and $hrz(\phi) = 3$, respectively. Let $\psi_1 = (s > 5 \wedge s < 6)$ and $\psi_2 = (s > 1 \wedge s < 2)$. Then satisfying Φ implies satisfying $\bigwedge\limits_{t \in [0,7]} (F_{[t,t+3]}\psi_1 \wedge F_{[t,t+3]}\psi_2)$. Let $\mathbf{s^1}$ and $\mathbf{s^2}$ be two signals as illustrated in Figure 1. The signal $\mathbf{s^1}$ satisfies Φ because A and B are visited within $[t,t+3]$ for every $t \in [0,7]$. However, the signal $\mathbf{s^2}$ violates Φ because region B is not visited within $[0,3]$. Moreover, the robustness degree of $\mathbf{s}$ with respect to Φ can be computed via the quantitative semantics as follows:

$$\min_{t \in [0,7]} \min \left\{ \max_{t' \in [t,t+3]} r(\mathbf{s}, \psi_1, t'), \max_{t' \in [t,t+3]} r(\mathbf{s}, \psi_2, t') \right\} \qquad (3)$$

Based on (3), the robustness degrees of $\mathbf{s^1}$ and $\mathbf{s^2}$ with respect to Φ are computed as $r(\mathbf{s^1}, \Phi) = 0.35$ and $r(\mathbf{s^2}, \Phi) = -1$.

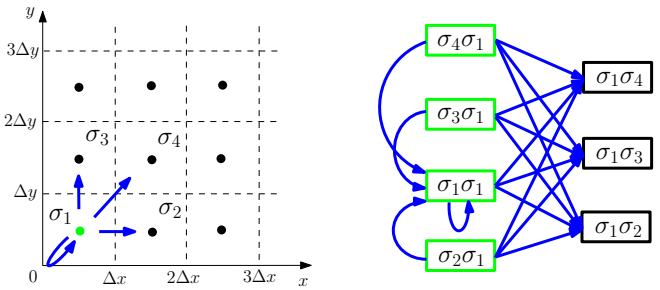
III. Problem Formulation

A. System Model

In this paper, we consider a system as a Markov decision process (MDP) $M = \langle \Sigma, A, P, R \rangle$, where Σ is the state-space of the system, A is a finite set of motion primitives, $P : \Sigma \times A \times \Sigma \to [0,1]$ is a probabilistic transition relation, and $R : \Sigma \to \mathbb{R}$ is a reward function. We assume that the state-space comprises a set of partitions and each $\sigma_i \in \Sigma$ corresponds to the centroid of a partition, e.g., $\sigma_1 = [\Delta x/2, \Delta y/2]$ in Figure 2(a). Moreover, each motion primitive $a \in A$ drives the system from the current state σ_i to an adjacent state σ_j. Let $s_t \in \Sigma$ denote the state of a system at time t, and let

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

$s_{t_1:t_2}$ denote the state trajectory of the system within $[t_1,t_2]$. Suppose that a system moves in an environment shown in Figure 2(a), and let its initial state be $s_0=\sigma_1$. If the system visits σ_3 and returns to σ_1, its state trajectory can be written as $s_{0:2\Delta t}=\sigma_1\sigma_3\sigma_1$ where $\Delta t>0$ is the discrete time step.



(a) (b)

Fig. 2. (a) Discretized state-space, (b) Representation of σ_1 over $2-MDP$.

In this paper, we assume that the MDP model M is already available. For more generic cases, abstractions of stochastic systems can be constructed via several methods (e.g., [16], [1], [19]). Moreover, if the system satisfies certain conditions, a discrete signal can be used to reason about whether or not the continuous signal satisfies a temporal logic formula (e.g., [9], [13], [12], [5]).

B. Problem Definition

In real-world applications, many systems (such as robotic systems) contain uncertainty in their dynamics due to mechanical, environmental, or sensing issues. In this respect, we consider an MDP M, for which the transition probability function P is unknown. This means that when the system at state s_t executes a motion primitive a, it is not certain where it will be in $t+1$, i.e., the probability distribution for s_{t+1} is unknown. Accordingly, the question becomes how to enforce an STL specification Φ to a system with unknown dynamics.

In this paper, we formulate two problems that have different objective functions to find a control policy π enforcing the desired specification Φ. In the first problem, we maximize the probability of satisfying Φ, which is a commonly used objective in formal synthesis problems (e.g., [17], [19], [6]). In the second problem, we maximize the expected robustness degree with respect to Φ, which has recently been used in model predictive control framework (e.g., [22], [24]).

Problem 1 (Maximizing Probability of Satisfaction): Let Φ be an STL specification with $hrz(\Phi)=T$. Given a stochastic model $M=\langle\Sigma,A,P,R\rangle$ with unknown P and an initial partial state trajectory $s_{0:\tau}$ for some $0\leq\tau<T$, find a control policy π such that

$$\pi_1^*=\arg\max_{\pi} Pr^{\pi}[s_{0:T}\models\Phi] \tag{4}$$

where $Pr^{\pi}[s_{0:T}\models\Phi]$ is the probability of $s_{0:T}$ satisfying Φ under policy π.

Problem 2 (Maximizing Expected Robustness Degree): Let Φ be an STL specification with $hrz(\Phi)=T$. Given a stochastic model $M=\langle\Sigma,A,P,R\rangle$ with unknown P and an initial partial state trajectory $s_{0:\tau}$ for some $0\leq\tau<T$, find a control policy π such that

$$\pi_2^*=\arg\max_{\pi} E^{\pi}[r(s_{0:T},\Phi)] \tag{5}$$

where $E^{\pi}[r(s_{0:T},\Phi)]$ is the expected robustness degree of $s_{0:T}$ with respect to Φ under policy π.

IV. Control Synthesis via Q-learning

For systems with unknown stochastic dynamics, reinforcement learning can be used to design optimal control policies (i.e., the system learns how to take actions by trial and error interactions with the environment [25]). In this paper, we use the Q-learning algorithm that is briefly presented in the following sub-section. Then, we discuss that Problems 1 and 2 are not in the standard form to apply the Q-learning algorithm. Finally, we present the main contribution of this paper, i.e., approximation of STL synthesis problems that can be solved via Q-learning.

A. Q-learning

Q-learning is a model-free reinforcement learning method [27], which can be used to find the optimal action selection policy for a finite MDP. In particular, the objective of an agent at state s_t is to maximize $V(s_t)$, its expected (discounted) reward in finite or infinite horizon, i.e.,

$$E\Big[\sum_{k=t+1}^{T} r(s_k)\Big] \quad \text{or} \quad E\Big[\sum_{k=0}^{\infty}\gamma^k r(s_{k+t+1})\Big], \tag{6}$$

where $r(s)$ is the reward obtained at state s, and $0<\gamma<1$ is the discount factor. Moreover, $V^*(s)=\max_a Q^*(s,a)$, where $Q^*(s,a)$ is the optimal Q-function for every pair of (s,a).

Starting from state s, the system chooses an action a, which takes it to state s' and results in a reward r. Then, the Q-learning rule is defined as follows:

$$Q(s,a):=(1-\alpha)Q(s,a)+\alpha[r+\gamma\max_{a^*\in A}Q(s',a^*)], \tag{7}$$

where $\gamma\in(0,1)$ is the discount factor and $\alpha\in(0,1]$ is the learning rate. Accordingly, if each action $a\in A$ is repetitively implemented in each state $s\in\Sigma$ for infinite number of times and α decays appropriately, then Q converges to Q^* with probability 1 (see Theorem 4.1). Thus, we can find the optimal policy $\pi^*:\Sigma\to A$ as $\pi^*=\arg\max_a Q^*(s,a)$. Algorithm 1 shows the steps of Q-learning.

Algorithm 1: Q-learning

Input: s - current state
Output: π- control policy maximizing the sum of (discounted) rewards

initialization: Arbitrary $Q(s,a)$ and π;
for $k=1:N_{episode}$
  Initialize s
  for $t=1:T$
    Select an action a (via ε-greedy or π);
    Take action a, observe r and s';
    $Q(s,a)\leftarrow(1-\alpha_k)Q(s,a)+\alpha_k\big[r+\gamma\max_{a'}Q(s',a')\big]$;
    $\pi(s)\leftarrow\arg\max_a Q(s,a)$;
    $s\leftarrow s'$;
  end for
end for

Theorem 4.1: [27], [26] Given a finite MDP, $M = \langle S, A, P, R \rangle$, let $Q^*(s,a)$ be the optimal Q-function for every pair of (s,a). Consider the Q-learning algorithm with the following update rule

$$Q_{k+1}(s,a) = (1-\alpha_k)Q_k(s,a) + \alpha_k[r + \gamma \max_{a^* \in A} Q_k(s',a^*)],$$

where $\gamma \in (0,1)$ is the discount factor and α_k satisfies

$$\sum_k \alpha_k = \infty \quad , \quad \sum_k \alpha_k^2 < \infty. \tag{8}$$

Then $Q_k(s,a)$ converges to $Q^*(s,a)$ with probability 1 as $k \to \infty$.

B. Q-learning and Formal Synthesis

There are several reasons why one cannot directly use Q-learning in Problems 1 and 2. First of all, the action selection at each time step cannot depend on only the current state as in Q-learning. For example, consider a specification $\Phi_1 = F_{[0,T]}\psi$ where $\psi = x > 3 \wedge x < 5 \wedge y > 3 \wedge y < 5$. Satisfying Φ_1 implies that visiting the desired region (i.e., $3 < x < 5$ and $3 < y < 5$) at least one time in $[0,T]$. Let the current state be $s_t := [x,y] = [2,2]$ and assume that the desired region is not visited before t. Note that if $t = T-1$, then the action selection via the optimal policy leads the agent to maximally approach the desired region. However, if $t = 0$, then the optimal policy may result in an action that drives the agent further away from the desired region (while ensuring to eventually satisfy ψ). Thus, the optimal policies may not necessarily be the same if the same state is occupied but the remaining mission horizons are different. Moreover, if Φ involves a nested temporal operator, the policy should also take into account a sufficient length of state history in addition to the current state and the remaining mission horizon. For example, consider $\Phi_2 = F_{[0,T]}G_{[0,\tau]}\psi$. Note that Φ_2 implies that the agent should eventually enter the desired region in $[0,T]$ and stay there for τ time steps. Similarly, let the current state be $s_t = [2,2]$. The action selection at t depends on the state history $s_{t-\tau:t}$ and the remaining mission horizon. Accordingly, the policies in Problems 1 and 2 should be defined as $\pi : \Sigma^\tau \times \mathbb{N}_{\geq 0} \to A$ where $\Sigma^\tau = \Sigma \times \cdots \times \Sigma$ for τ times. Hence, the state-space of the system needs to be redefined as $\Sigma^\tau \times \mathbb{N}_{\geq 0}$.

Secondly, although the state-space is redefined based on the previous discussion, one can still not directly apply Q-learning because an agent trying to optimize (4) or (5) does not have an immediate reward after taking an action. Consider a specification Φ such that $hrz(\Phi) = T$. Accordingly, both satisfaction and the robustness degree can be computed over a T-length trajectory (i.e., these measures are undefined for partial trajectories having a length smaller than T). For example, consider an agent trying to satisfy $\Phi_1 = F_{[0,T]}\psi$. Then, the objective function in Problem 2 can be written as

$$\max_{\pi} E^{\pi}\Big[\max\big(r(s_{0:T},\psi,0),\ldots,r(s_{0:T},\psi,T)\big)\Big]. \tag{9}$$

Hence, the objective functions in (4) or (5) are not in the standard form of Q-learning, i.e., the sum of (discounted) rewards as in (6).

C. Proposed Approach

In this paper, we approximate the synthesis problems in (4) and (5) such that one can use the Q-learning algorithm to find the optimal policy. The overview of the proposed method is: 1) for any STL formula (i.e., $G_{[0,T]}\phi$ or $F_{[0,T]}\phi$), redefine the state-space as Σ^τ where τ is a function of $hrz(\phi)$; 2) redefine the objective function such that an agent observes an immediate reward after taking each action and the remaining mission horizon can be eliminated in the policy design. After executing these steps, we will show that one can use the Q-learning algorithm to find the optimal policy $\pi^* : \Sigma^\tau \to A$.

Let Φ be $G_{[0,T]}\phi$ or $F_{[0,T]}\phi$, where Φ and ϕ are STL formulae with the syntax in (1). Let the horizon length of ϕ be $hrz(\phi) = \tau$. Then, we denote the τ-state of the agent at time t by s_t^τ, which is the τ-horizon trajectory involving the current state and the most recent $\tau - 1$ past states, i.e., $s_t^\tau = s_{t-\tau+1:t}$. By considering all τ-states of the agent, we remodel the agent as a τ-MDP.

Definition 1 (τ-MDP): Given an MDP $M = (\Sigma, A, P, R)$ and $\tau \in \mathbb{N}_{>0}$, a τ-MDP is a tuple $M^\tau = (\Sigma^\tau, A, P^\tau, R^\tau)$, where

- $\Sigma^\tau \subseteq (\Sigma \cup \varepsilon)^\tau$ is the set of finite states, where ε is the empty string. Each state $\sigma^\tau \in \Sigma^\tau$ corresponds to a $\tau-$horizon (or shorter) path on Σ. Shorter paths of length $n < \tau$ (representing the case in which the system has not yet evolved for τ time steps) have ε prepended $\tau - n$ times.
- $P^\tau : \Sigma^\tau \times A \times \Sigma^\tau \to [0,1]$ is a probabilistic transition relation. Let $\sigma_i^\tau = \sigma_a\sigma_b\ldots\sigma_c\sigma_d$ and $\sigma_j^\tau = \sigma_e\ldots\sigma_f\sigma_g$. $P^\tau(\sigma_i^\tau, a, \sigma_j^\tau) > 0$ if and only if $P(\sigma_d, a, \sigma_g) \in [0,1]$ and for $\tau > 1$ the first $\tau - 1$ elements of σ_j^τ are equal to the last $\tau - 1$ elements of σ_i^τ (i.e., $\sigma_e\ldots\sigma_f = \sigma_b\ldots\sigma_d$).
- $R^\tau : \Sigma^\tau \to \mathbb{R}$ is a reward function.

For instance, the highlighted state σ_1 in Figure 2(a) corresponds to four τ-states for $\tau = 2$ as illustrated in Figure 2(b).

For any $\Phi = F_{[0,T]}\phi$ or $\Phi = G_{[0,T]}\phi$, τ can be computed as follows:

$$\tau = \left\lceil \frac{hrz(\phi)}{\Delta t} \right\rceil + 1 \tag{10}$$

where Δt is the time step and $\lceil . \rceil$ is the ceiling function, i.e., $\lceil x \rceil$ is the smallest integer not less than $x \in \mathbb{R}$.

Remark 1: If Φ does not have nested temporal operators, then $hrz(\phi) = 0$ and $\tau = 1$ as a consequence. As such, $M^\tau = M$ for $\tau = 1$.

For any state trajectory $s_{0:T}$, we can write the corresponding τ-state trajectory as $s_{\tau-1:T}^\tau = s_{\tau-1}^\tau \ldots s_T^\tau$ where each $s_t^\tau = s_{t-\tau+1:t}$ for $\tau - 1 \leq t \leq T$. Moreover, for each τ-state s_t^τ, we can compute the corresponding robustness degree with respect to ϕ. Accordingly, the robustness degree of $s_{0:T}$ with respect to Φ can be written in terms of τ-states as

$$r(s_{0:T},\Phi) = \begin{cases} \max\big(r(s_{\tau-1}^\tau,\phi),\ldots,r(s_T^\tau,\phi)\big), & \text{if } \Phi = F_{[0,T]}\phi \\ \min\big(r(s_{\tau-1}^\tau,\phi),\ldots,r(s_T^\tau,\phi)\big), & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{11}$$

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

Note that plugging (11) into (5) makes the objective in Problem 2 as follows:

$$\max_{\pi} E^{\pi}[r(s_{0:T},\Phi)] = \begin{cases} \max_{\pi} E^{\pi}\big[\max_{\tau-1\le t\le T}\big(r(s_t^{\tau},\phi)\big)\big], & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\big[\min_{\tau-1\le t\le T}\big(r(s_t^{\tau},\phi)\big)\big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{12}$$

Since Q-learning cannot be used for cases like (12), we propose to use the *log-sum-exp* [4] approximation of the maximum function to represent the objective as a sum of rewards, i.e.,

$$\max(x_1,\ldots,x_n) \sim \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta x_i}, \tag{13}$$

where $\beta > 0$ is a constant and

$$\max(x_1,\ldots,x_n) \le \frac{1}{\beta}\log\sum_{i=1}^{n} e^{\beta x_i} \le \max(x_1,\ldots,x_n) + \frac{1}{\beta}\log n. \tag{14}$$

Based on (13), the equation in (12) can be approximated as

$$\max_{\pi} E^{\pi}[r(s_{0:T},\Phi)] \sim \begin{cases} \max_{\pi} E^{\pi}\Big[\frac{1}{\beta}\log\sum_{t=\tau-1}^{T} e^{\beta r(s_t^{\tau},\phi)}\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\Big[-\frac{1}{\beta}\log\sum_{t=\tau-1}^{T} e^{-\beta r(s_t^{\tau},\phi)}\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{15}$$

Similarly, maximizing the probability of satisfying Φ can be written as

$$\max_{\pi} Pr^{\pi}\big[s_{0:T} \models \Phi\big] = \max_{\pi} E^{\pi}\Big[I\big(r(s_{0:T},\Phi)\big)\Big] \tag{16}$$

where $I(.)$ is the indicator function defined as

$$I(x) = \begin{cases} 1, & \text{if } x \ge 0 \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Since $I\big(\max(x_1,\ldots,x_n)\big) = \max\big(I(x_1),\ldots,I(x_n)\big)$ (or $I\big(\min(x_1,\ldots,x_n)\big) = \min\big(I(x_1),\ldots,I(x_n)\big)$), plugging (11) into (16) makes the objective in Problem 1 as follows:

$$\max_{\pi} Pr^{\pi}\big[s_{0:T} \models \Phi\big] = \begin{cases} \max_{\pi} E^{\pi}\Big[\max_{\tau-1\le t\le T} I\big(r(s_t^{\tau},\phi)\big)\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\Big[\min_{\tau-1\le t\le T} I\big(r(s_t^{\tau},\phi)\big)\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{18}$$

Based on (13), the equation in (18) can be approximated as

$$\max_{\pi} Pr^{\pi}\big[s_{0:T} \models \Phi\big] \sim \begin{cases} \max_{\pi} E^{\pi}\Big[\frac{1}{\beta}\log\sum_{t=\tau-1}^{T} e^{\beta I\left(r(s_t^{\tau},\phi)\right)}\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \max_{\pi} E^{\pi}\Big[-\frac{1}{\beta}\log\sum_{t=\tau-1}^{T} e^{-\beta I\left(r(s_t^{\tau},\phi)\right)}\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{19}$$

Problem 1A (Max. Approx. Probability of Satisfaction): Let Φ and ϕ be STL formulae with the syntax in (1) such that $\Phi = F_{[0,.]}\phi$ or $\Phi = G_{[0,.]}\phi$. Let $hrz(\Phi) = T$. Given an unknown MDP M, let $M^{\tau} = \langle \Sigma^{\tau}, A, P^{\tau}, R^{\tau}\rangle$ be the τ-MDP where τ is computed as in (10). Assume that the initial τ-state $s_{\tau-1}^{\tau} = s_{0:\tau-1}$ is given and $\beta > 0$. Find a control policy $\pi : \Sigma^{\tau} \to A$ such that

$$\pi_{1A}^{*} = \begin{cases} \arg\max_{\pi} E^{\pi}\Big[\sum_{t=\tau-1}^{T} e^{\beta I(r(s_t^{\tau},\phi))}\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_{\pi} E^{\pi}\Big[-\sum_{t=\tau-1}^{T} e^{-\beta I(r(s_t^{\tau},\phi))}\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{20}$$

Problem 2A (Max. Expected Approx. Robustness Degree): Let Φ and ϕ be STL formulae with the syntax in (1) such that $\Phi = F_{[0,.]}\phi$ or $\Phi = G_{[0,.]}\phi$. Let $hrz(\Phi) = T$. Given an unknown MDP M, let $M^{\tau} = \langle \Sigma^{\tau}, A, P^{\tau}, R^{\tau}\rangle$ be the τ-MDP where τ is computed as in (10). Assume that the initial τ-state $s_{\tau-1}^{\tau} = s_{0:\tau-1}$ is given and $\beta > 0$. Find a control policy $\pi : \Sigma^{\tau} \to A$ such that

$$\pi_{2A}^{*} = \begin{cases} \arg\max_{\pi} E^{\pi}\Big[\sum_{t=\tau-1}^{T} e^{\beta r(s_t^{\tau},\phi)}\Big], & \text{if } \Phi = F_{[0,T]}\phi \\ \arg\max_{\pi} E^{\pi}\Big[-\sum_{t=\tau-1}^{T} e^{-\beta r(s_t^{\tau},\phi)}\Big], & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \tag{21}$$

In the following corollary, we show that Q-learning can be used in Problems 1A and 2A to obtain the optimal control policies.

Corollary 4.2: Let Φ and ϕ be STL specifications such that $\Phi = F_{[0,.]}\phi$ or $\Phi = G_{[0,.]}\phi$. Given a finite MDP M, let $M^{\tau} = \langle \Sigma^{\tau}, A, P^{\tau}, R^{\tau}\rangle$ be the τ-MDP where τ is computed as in (10) and $Q^{*}(s^{\tau},a)$ is the optimal Q-function for every pair of (s^{τ},a). Consider the Q-learning algorithm with the following update rule

$$Q_{k+1}(s_i^{\tau},a) = (1-\alpha_k)Q_k(s_i^{\tau},a) + \alpha_k[R + \gamma\max_{a^{*}\in A} Q_k(s_j^{\tau},a^{*})],$$

where s_j^{τ} is the resulting state by taking action a at s_i^{τ}, $\gamma \in (0,1)$, α_k satisfies

$$\sum_k \alpha_k = \infty \quad , \quad \sum_k \alpha_k^2 < \infty \tag{22}$$

and, for some $\beta > 0$, the immediate reward R obtained at s_j^{τ} is defined as

$$R = \begin{cases} e^{\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta I(r(s_j^{\tau},\phi))}, & \text{if Problem 1A with } \Phi = G_{[0,T]}\phi \\ e^{\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = F_{[0,T]}\phi \\ -e^{-\beta r(s_j^{\tau},\phi)}, & \text{if Problem 2A with } \Phi = G_{[0,T]}\phi \end{cases} \tag{23}$$

Then $Q_k(s^{\tau},a)$ converges to $Q^{*}(s^{\tau},a)$ with probability 1 as $k \to \infty$.

Proof: The proof follows from Theorem 4.1. ■

Now, we will show that the solutions of Problems 1A and 2A get closer to the solutions of Problems 1 and 2 as β increases.

Theorem 4.3: Let Φ and ϕ be STL formulae with the syntax in (1) such that $\Phi = F_{[0,.]}\phi$ or $\Phi = G_{[0,.]}\phi$. Let $hrz(\Phi) = T$. Assume that a partial state trajectory $s_{0:\tau-1}$ is initially given where τ is computed as in (10). For some

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

$\beta > 0$ and $\Delta t > 0$, let π_1^*, π_2^*, π_{1A}^*, π_{2A}^* be the optimal policies obtained by solving Problems 1, 2, 1A, 2A, respectively. Then,

$$Pr^{\pi_1^*}[s_{0:T} \models \Phi] - \frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2) \leq Pr^{\pi_{1A}^*}[s_{0:T} \models \Phi]$$
$$E^{\pi_2^*}[r(s_{0:T}, \Phi)] - \frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2) \leq E^{\pi_{2A}^*}[r(s_{0:T}, \Phi)]$$

Proof: First, we will show that solving (20) is equivalent to solving the right hand-side of (19). Let $\mathbf{s}^\tau = s^\tau_{\tau-1:T}$ and

$$g(\mathbf{s}^\tau) = \begin{cases} \sum_{t=\tau-1}^{T} e^{\beta I(r(s_t^\tau, \phi))}, & \text{if } \Phi = F_{[0,T]}\phi \\ -\sum_{t=\tau-1}^{T} e^{-\beta I(r(s_t^\tau, \phi))}, & \text{if } \Phi = G_{[0,T]}\phi \end{cases} \quad (24)$$

Since $\log(.)$ is a strictly monotonic function and $1/\beta$ is a constant,

$$\max_\pi E^\pi\Big[g(\mathbf{s}^\tau)\Big] \Leftrightarrow \max_\pi E^\pi\Big[\frac{1}{\beta}\log g(\mathbf{s}^\tau)\Big]. \quad (25)$$

In other words, π_{1A}^* is also the optimal policy for the right hand side of (19). Following the similar steps, we can show that solving (21) is equivalent to solving the right hand-side of (15), thus π_{2A}^* is also the optimal policy for the right hand side of (15).

Note that any τ-state trajectory $\mathbf{s}^\tau$ implies a state trajectory $\mathbf{s} = s_{0:T}$. Let Π be the set of policies. Starting from $s_{0:\tau-1}$ (i.e., initially given partial state trajectory), any $\pi \in \Pi$ induces a set of trajectories. Then, based on (14),

$$E^\pi[g(\mathbf{s}^\tau)] \leq Pr^\pi[\mathbf{s} \models \Phi] + \frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2) \quad (26)$$

where $\frac{T}{\Delta t} - \tau + 2$ is the total length of the τ-state trajectory (i.e., $\mathbf{s}^\tau = s^\tau_{\tau-1} s^\tau_{\tau-1+\Delta t} \ldots s^\tau_T$). The equation in (26) implies that the approximation function can over-evaluate the set of trajectories obtained by a policy π at most $\frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2)$. Hence, π_{1A}^* can result in a sub-optimal performance that is at most $\frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2)$ away from the performance obtained by π_1^*. Again, following the same steps, we can show that π_{2A}^* results in a sub-optimal performance that is at most $\frac{1}{\beta}\log(\frac{T}{\Delta t} - \tau + 2)$ away from the performance obtained by π_2^*. ■

Remark 2: Based on Theorem 4.3, arbitrarily large selection of β significantly reduces the performance gap between the solutions obtained via Problems 1 and 1A (or 2 and 2A). However, larger values of β would increase the maximum reward hence would reduce the convergence rate in Q-learning [20].

V. Simulation Results

In the following case studies, we consider a single agent moving in a discretized environment. The set of motion primitives at each state is $A = \{N, NW, W, SW, S, SE, E, NE, stay\}$. We model the motion uncertainty as in Figure 3 where, for any selected feasible action in A, the agent follows the corresponding blue arrow with probability 0.93 or a randomly selected red arrow with probability 0.07. Moreover, the resulting state after taking an infeasible action (i.e., the agent is next to a boundary and tries to move towards it) is the current state. All simulations were implemented in Matlab and performed on a PC with a 2.8 GHz processor and 8.0 GB RAM.

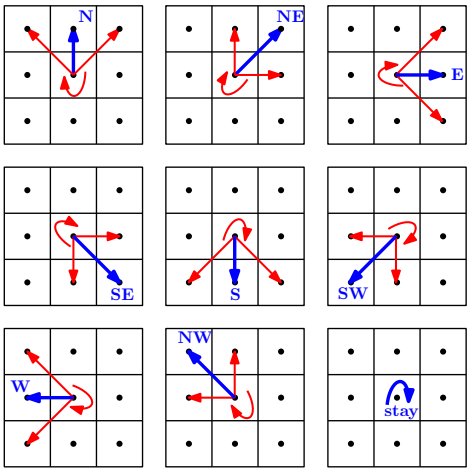


Fig. 3. The motion uncertainty (as red arrows) for a particular action (blue arrow).

A. Case Study 1: Reachability

In this case study, the initial state of the agent is $s_0 = [1.5, 1.5]$ as shown in Figure 4. We consider an STL formula defined over the environment as

$$\Phi_1 = F_{[0,7]}(x > 4 \wedge y > 4), \quad (27)$$

which expresses "eventually visit the desired region within $[0,7]$. Note that $\Phi_1 = F_{[0,8]}\phi$ where $\phi = (x > 4 \wedge y > 4)$ and $hrz(\phi) = 0$. Moreover, we choose $\Delta t = 1$, thus $\tau = 1$ from (10). The state-space of the system contains all (x,y) coordinates in the environment, i.e., $|S| = 36$. Since $\tau = 1$, the τ-state-space $S^\tau = S$.

To implement the Q-learning algorithm, the number of episodes is chosen as 1700 (i.e., $1 \leq k \leq 1700$), and we use the parameters $\beta = 50$, $\alpha_k = 0.95^k$, and $\gamma = 0.9999$. After 1700 trainings (episodes), which took approximately 1 minute for each problem, the resulting policies π_{1A}^* and π_{2A}^* are used to generate 1000 trajectories, which lead to

$$E^{\pi_{1A}^*}[r(s_{0:7}, \Phi_1)] = 0.523 \quad Pr^{\pi_{1A}^*}[s_{0:7} \models \Phi_1] = 0.999$$
$$E^{\pi_{2A}^*}[r(s_{0:7}, \Phi_1)] = 1.497 \quad Pr^{\pi_{2A}^*}[s_{0:7} \models \Phi_1] = 1.000$$

Sample trajectories generated by π_{1A}^* and π_{2A}^* are displayed in Figure 4 (a) and (b), respectively. While Φ_1 is satisfied with probability 1 in both cases, the trajectories via maximizing the expected robustness degree tend to reach the deepest state (i.e., having the maximum robustness degree with respect to ϕ) in the desired region.

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

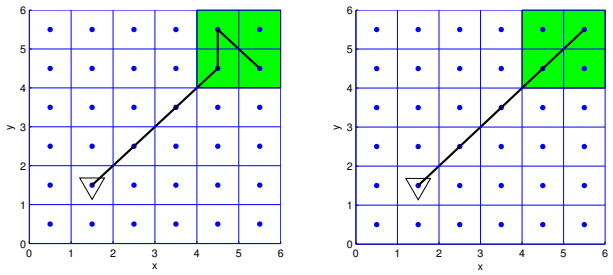


Fig. 4. The environment and the desired region in case study 1 for which a sample trajectory by (a) π^*_{1A} and (b) π^*_{2A}.

B. Case Study 2: Repeated Satisfiability

In the second case study, we consider an agent moving in an environment illustrated in Figure 5(a). We consider an STL formula defined over the environment as

$$\Phi_2 = G_{[0,12]}\left(F_{[0,2]}(\textit{region } A) \wedge F_{[0,2]}(\textit{region } B)\right) \tag{28}$$

where *region A* represents $x > 1 \wedge x < 2 \wedge y > 3 \wedge y < 4$ and *region B* represents $x > 2 \wedge x < 3 \wedge y > 2 \wedge y < 3$. Note that Φ_2 expresses the following: "for all $t \in [0,12]$, eventually visit *region A* every $[t, t+2]$ and eventually visit *region B* every $[t, t+2]$". Note that $\Phi_2 = G_{[0,12]}\phi$ where $\phi = F_{[0,2]}(\textit{region } A) \wedge F_{[0,2]}(\textit{region } B)$ and $hrz(\phi) = 2$. Assuming that $\Delta t = 1$, $\tau = 3$ based on (10).

In this case study, the sizes of the state-spaces are $|S| = 19$ and $|S^\tau| = 676$ for $\tau = 3$. To implement the Q-learning algorithm, the number of episodes is chosen as 2000 (i.e., $1 \leq k \leq 2000$), and we use the parameters $\beta = 50$, $\alpha_k = 0.95^k$, and $\gamma = 0.9999$. After 2000 trainings, which took approximately 6 minutes for each problem, the resulting policies π^*_{1A} and π^*_{2A} are used to generate 500 trajectories, which lead to

$$E^{\pi^*_{1A}}[r(s_{0:14}, \Phi_2)] = 0.084 \quad Pr^{\pi^*_{1A}}[s_{0:14} \models \Phi_2] = 0.732$$
$$E^{\pi^*_{2A}}[r(s_{0:14}, \Phi_2)] = 0.422 \quad Pr^{\pi^*_{2A}}[s_{0:14} \models \Phi_2] = 0.936$$

Sample trajectories generated by π^*_{1A} and π^*_{2A} are displayed in Figure 5 (b) and (c), respectively. In this case study, the performances obtained by maximizing probability of satisfaction and expected robustness degree are different from each other. The discrepancy between the two solutions can be explained by what happens when trajectories almost satisfy Φ_2. While solving Problem 1A, if a τ-state slightly violating or strongly violating ϕ (i.e., a partial trajectory almost oscillating or not oscillating between the regions A and B in two seconds) is encountered, the overall reward is observed as -1. On the other hand, while solving Problem 2A, the policy producing the slightly violating τ-state (i.e., almost oscillatory partial trajectory) will be reinforced much more strongly than an arbitrary policy as the resulting robustness degree is larger. Since the robustness degree gives "partial credit" for trajectories that are close to satisfaction, the Q-learning algorithm performs a directed search to find policies that satisfy the formula. Since probability maximization gives no partial credit, the Q-learning algorithm is essentially

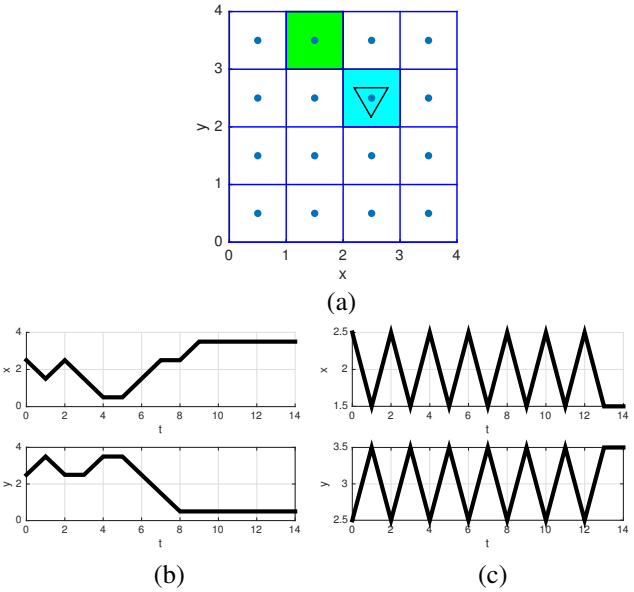


Fig. 5. (a) The initial state and the desired regions in case study 2 for which a sample trajectory by (b) π^*_{1A} and (c) π^*_{2A}.

performing a random search until it encounters a trajectory that satisfies the given formula. Therefore, if the family of policies that satisfy the formula with positive probability is small, it will on average take the Q-learning algorithm solving Problem 1A a longer time to converge to a solution that enforces formula satisfaction.

VI. Conclusions and Future Work

We considered a system that is modeled as a Markov decision process (MDP) with unknown transition probabilities and is required to satisfy a complex task given as a signal temporal logic (STL) formula. To find a control policy enforcing the desired STL formula, we addressed two problems that maximize 1) the probability of satisfaction, and 2) the expected robustness degree, i.e., a measure quantifying the quality of satisfaction. One way to learn optimal policies for unknown stochastic MDPs is via Q-learning, where an agent receives a reward after each action; the objective is maximizing the sum of rewards; and the action selection depends only on the current state. However, the problems maximizing the probability of satisfaction and the expected robustness degree do not have the aforementioned properties.

In this paper, we propose an approximation of STL synthesis problems that can be solved via Q-learning. The main ingredients of the proposed method are 1) remodeling the system as a τ-MDP where each state corresponds to a τ-length trajectory and τ is computed based on the given STL formula, 2) approximating the probability of satisfaction and expected robustness degree such that the new objective functions are in the form of sum of rewards. We also showed that the polices computed by the proposed method can be sufficiently close to the policies of the original problems when the approximation parameter is selected properly. Finally, we demonstrated the performance of the proposed method on some case studies, and we observed that

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.

after the same number of trainings, the converged policy by maximizing the expected robustness degree performs better than the converged policy by maximizing the probability of satisfaction. Future research includes incorporating complexity reduction techniques for faster convergence to optimal policies and extending this work for multi-agent systems.

REFERENCES

[1] A. Abate, A. D'Innocenzo, and M. Di Benedetto. Approximate abstractions of stochastic hybrid systems. *Automatic Control, IEEE Transactions on*, 56(11):2688–2694, Nov 2011.

[2] C. Baier and J.-P. Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[3] T. Brazdil, K. Chatterjee, M. Chmelik, M.k, V. Forejt, J. Kretinsky, M. Kwiatkowska, D. Parker, and M. Ujma. Verification of markov decision processes using learning algorithms. In F. Cassez and J.-F. Raskin, editors, *Automated Technology for Verification and Analysis*, volume 8837 of *Lecture Notes in Computer Science*, pages 98–114. Springer International Publishing, 2014.

[4] M. Chen and M. Chiang. Distributed optimization in networking: Recent advances in combinatorial and robust formulations. In *Modeling and Optimization: Theory and Applications*, pages 25–52. Springer, 2012.

[5] L. De Alfaro and Z. Manna. Verification in continuous time by discrete reasoning. In *Algebraic Methodology and Software Technology*, pages 292–306. Springer, 1995.

[6] X. C. Ding, S. L. Smith, C. Belta, and D. Rus. Optimal control of markov decision processes with linear temporal logic constraints. *IEEE Transactions on Automatic Control*, 59(5):1244–1257, 2014.

[7] A. Dokhanchi, B. Hoxha, and G. Fainekos. On-line monitoring for temporal logic robustness. In *Runtime Verification*, pages 231–246. Springer, 2014.

[8] A. Donzé and O. Maler. *Robust satisfaction of temporal logic over real-valued signals*. Springer, 2010.

[9] G. E. Fainekos and G. J. Pappas. *Robust sampling for MITL specifications*. Springer, 2007.

[10] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[11] J. Fu and U. Topcu. Probably approximately correct MDP learning and control with temporal logic constraints. *CoRR*, abs/1404.7073, 2014.

[12] C. A. Furia and M. Rossi. Integrating discrete-and continuous-time metric temporal logics through sampling. In *Formal Modeling and Analysis of Timed Systems*, pages 215–229. Springer, 2006.

[13] J. Huang, J. Voeten, and M. Geilen. Real-time property preservation in approximations of timed systems. In *Formal Methods and Models for Co-Design, 2003. MEMOCODE '03. Proceedings. First ACM and IEEE International Conference on*, pages 163–171, 2003.

[14] X. Jin, A. Donze, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid systems: computation and contro*, pages 43–52, 2013.

[15] A. Jones, Z. Kong, and C. Belta. Anomaly detection in cyber-physical systems: A formal methods approach. In *IEEE Conference on Decision and Control (CDC)*, pages 848–853, 2014.

[16] A. Julius and G. Pappas. Approximations of stochastic hybrid systems. *Automatic Control, IEEE Transactions on*, 54(6):1193–1203, June 2009.

[17] M. Kamgarpour, J. Ding, S. Summers, A. Abate, J. Lygeros, and C. Tomlin. Discrete time stochastic hybrid dynamic games: Verification and controller synthesis. In *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pages 6122–6127, 2011.

[18] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 273–282. ACM, 2014.

[19] M. Lahijanian, S. B. Andersson, and C. Belta. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Transactions on Automatic Control*, 6(8):2031–2045, 2015.

[20] S. H. Lim and G. DeJong. Towards finite-sample convergence of direct reinforcement learning. In *Machine Learning: ECML 2005*, pages 230–241. Springer, 2005.

[21] R. Luna, M. Lahijanian, M. Moll, and L. E. Kavraki. Asymptotically optimal stochastic motion planning with temporal goals. In *Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, 03/08/2014 2014.

[22] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. Model predictive control with signal temporal logic specifications. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 81–87, 2014.

[23] 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. *CoRR*, abs/1409.5486, 2014.

[24] S. Sadraddini and C. Belta. Robust temporal logic model predictive control. In *53nd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015.

[25] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[26] J. N. Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine Learning*, 16(3):185–202, 1994.

[27] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Preprint submitted to 55th IEEE Conference on Decision and Control.
Received March 15, 2016.