2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Statistically Model Checking PCTL Specifications on Markov Decision Processes via Reinforcement Learning

Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud

***Abstract*— Probabilistic Computation Tree Logic (PCTL) is frequently used to formally specify control objectives such as probabilistic reachability and safety. In this work, we focus on model checking PCTL specifications statistically on Markov Decision Processes (MDPs) by sampling, e.g., checking whether there exists a feasible policy such that the probability of reaching certain goal states is greater than a threshold. We use reinforcement learning to search for such a feasible policy for PCTL specifications, and then develop a statistical model checking (SMC) method with provable guarantees on its error. Specifically, we first use upper-confidence-bound (UCB) based Q-learning to design an SMC algorithm for bounded-time PCTL specifications, and then extend this algorithm to unbounded-time specifications by identifying a proper truncation time by checking the PCTL specification and its negation at the same time. Finally, we evaluate the proposed method by case studies.**

I. Introduction

Probabilistic Computation Tree Logic (PCTL) is frequently used to formally specify control objectives such as reachability and safety on probabilistic systems [1]. To check the correctness of PCTL specifications on these systems, *model checking* methods are required [2]. Although model checking PCTL by model-based analysis is theoretically possible [1], it is not preferable in practice when the system model is unknown or large, and model checking by sampling, i.e. statistical model checking (SMC), is needed [3].

The statistical model checking of PCTL specifications on Markov Decision Processes (MDPs) is frequently encountered in many decision problems – e.g., for a robot in a grid world under probabilistic disturbance, checking whether there exists a feasible control policy such that the probability of reaching certain goal states is greater than a probability threshold [4]–[8]. In these problems, the main challenge is to search for such a feasible policy for the PCTL specifications.

To search for feasible policies for temporal logics specifications, such as PCTL, on MDPs, one approach is model-based reinforcement learning [9]–[12] – i.e., first inferring the transition probabilities of the MDP by sampling over each state-action pair, and then searching for the feasible policy via model-based analysis. This approach is often inefficient, since not all transition probabilities are relevant to the PCTL specification of interest. Here instead, we adopt a model-free reinforcement learning approach [13].

Yu Wang is with Duke University, USA yu.wang094@duke.edu. Nima Roohi is with the University of California San Diego, USA nroohi@ucsd.edu. Matthew West, Mahesh Viswanathan, and Geir E. Dullerud are with the University of Illinois at Urbana-Champaign, USA {mwest, vmahesh, dullerud}@illinois.edu

Common model-free reinforcement learning techniques cannot directly handle temporal logic specifications. One solution is to find a surrogate reward function such that the policy learned for this surrogate reward function is the one needed for checking the temporal logic specification of interest. For certain temporal logics interpreted under special semantics (usually involving a metric), the surrogate reward can be found based on that semantics [14]–[16].

For temporal logics under the standard semantics [17], the surrogate reward functions can be derived via constructing the product MDP [8], [18], [19] of the initial MDP and the automaton realizing the temporal logic specification. However, the complexity of constructing the automaton from a general linear temporal logic (LTL) specification is *double exponential* [17], [20]. For a fraction of LTL, namely LTL/GU, the complexity is *exponential* [21], [22]. In addition, the size of the product MDP is usually much larger than the initial MDP, although the produce MDP may be constructed on-the-fly to reduce the extra computation cost, as it did in [19].

In this work, we propose a new statistical model checking method for PCTL specifications on MDPs. For a lucid discussion, we only consider non-nested PCTL specifications. PCTL formulas in general form with nested probabilistic operators can be handled in the standard manner using the approach proposed in [23]. Our method uses upper-confidence-bound (UCB) based Q-learning to directly learn the feasible policy of PCTL specifications, without constructing the product MDP. The effectiveness of UCB-based Q-learning has been proven for the K-bandit problem, and has been numerically demonstrated on many decision-learning problems on MDPs (see [24]).

For finite-horizon PCTL specifications, we treat the statistical model checking problem as a finite sequence of K-bandit problems and use the UCB-based Q-learning to learn the desirable decision at each time step. For infinite-horizon PCTL specifications, we look for a truncation time to reduce it to a finite-horizon problem by checking the PCTL specification and its negation at the same time. Our statistical model checking algorithm is *online*; it terminates with probability 1, and only when the statistical error of the learning result is smaller than a user-specified value.

The rest of the paper is organized as follows. The preliminaries on labeled MDPs and PCTL are given in Section II. In Section III, using the principle of optimism in the face of uncertainty, we design Q-learning algorithms to solve finite-time and infinite-time probabilistic satisfaction, and give finite sample probabilistic guarantees for the correctness

978-1-7281-7447-1/20/$31.00 ©2020 IEEE

of the algorithms. We implement and evaluate the proposed algorithms on several case studies in Section IV. Finally, we conclude this work in Section V.

II. Preliminaries and Problem Formulation

The set of integers and real numbers are denoted by $\mathbb{N}$ and $\mathbb{R}$, respectively. For $n \in \mathbb{N}$, let $[n] = \{1, \ldots, n\}$. The cardinality of a set is denoted by $|\cdot|$. The set of finite-length sequences taken from a finite set S is denoted by S^*.

A. Markov Decision Process

A Markov decision process (MDP) is a finite-state probabilistic system, where the transition probabilities between the states are determined by the control action taken from a given finite set. Each state of the MDP is labeled by a set of *atomic propositions* indicating the properties holding on it, e.g., whether the state is a safe/goal state. Formally, an MDP is a tuple $\mathcal{M} = (S, A, \mathbf{T}, \mathrm{AP}, L)$ where

- S is a finite set of *states*.
- A is a finite set of *actions*.
- $\mathbf{T} : S \times A \times S \to [0, 1]$ is a partial *transition probability* function. For any state $s \in S$ and any action $a \in A$,

$$\sum_{s' \in S} \mathbf{T}(s, a, s') = \begin{cases} 0, & \text{if } a \text{ is not allowed on } s \\ 1, & \text{otherwise.} \end{cases}$$

  With a slight abuse of notation, let $A(s)$ be the set of allowed actions on the state s.
- AP is a finite set of *labels*.
- $L : S \to 2^{\mathrm{AP}}$ is a *labeling function*.

A policy $\Pi : S^* \to A$ decides the action to take from the sequence of states visited so far. Given a policy Π and an initial state $s \in S$, the MDP $\mathcal{M}$ becomes purely probabilistic, denoted by $\mathcal{M}_{\Pi,s}$. The system $\mathcal{M}_{\Pi,s}$ is not necessarily Markovian.

B. Probabilistic Computation Tree Logic

The probabilistic computation tree logic (PCTL) is defined inductively from atomic propositions, temporal operators, and probability operators. It reasons about the probabilities of time-dependent properties. Let AP be a set of atomic propositions. A PCTL (state) formula is defined by

$$\begin{aligned} \phi ::= & \mathsf{a} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi) \mid \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi) \\ & \mid \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) \mid \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2) \\ & \mid \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) \mid \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2) \end{aligned}$$

where $\mathsf{a} \in \mathrm{AP}$, $\bowtie \in \{<, >, \leq, \geq\}$, $T \in \mathbb{N} \cup \{\infty\}$ is a (possibly infinite) time horizon, and $p \in [0, 1]$ is a threshold.[1] The operators $\mathbf{P}^{\min}_{\bowtie p}$ and $\mathbf{P}^{\max}_{\bowtie p}$ are called probability operators, and the "next", "until" and "release" operators $\mathbf{X}$, $\mathbf{U}_T$, $\mathbf{R}_T$ are called temporal operators. More temporal operators can be derived by composition: for example, "or" is $\phi_1 \vee \phi_2 ::= \neg(\neg\phi_1 \wedge \neg\phi_2)$; "true" is $\mathtt{True} = \mathsf{a} \vee (\neg\mathsf{a})$; "finally" is $\mathbf{F}_T\phi ::= \mathtt{True}\mathbf{U}_T\phi$; and "always" is $\mathbf{G}_T\phi ::= \mathtt{False}\mathbf{R}_T\phi$. For simplicity, we write $\mathbf{U}_\infty$, $\mathbf{R}_\infty$, $\mathbf{F}_\infty$ and $\mathbf{G}_\infty$ as $\mathbf{U}$, $\mathbf{R}$, $\mathbf{F}$ and $\mathbf{G}$, respectively.

[1]This logic is a fraction of PCTL* from [17].

For an MDP $\mathcal{M} = (S, A, \mathbf{T}, s_{\mathrm{init}}, \mathrm{AP}, L)$, the satisfaction relation $\models$ is defined by for a state $s \in S$ or an (infinite) path $\sigma : \mathbb{N} \to S$ by

$s \models \mathsf{a}$ iff $\mathsf{a} \in L(s)$,
$s \models \neg\phi$ iff $s \not\models \phi$,
$s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$,
$s \models \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi)$ iff $\min_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \mathbf{X}\phi\right] \bowtie p$,
$s \models \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi)$ iff $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \mathbf{X}\phi\right] \bowtie p$,
$s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$ iff $\min_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{U}_T \phi_2\right] \bowtie p$,
$s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$ iff $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{U}_T \phi_2\right] \bowtie p$,
$s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2)$ iff $\min_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{R}_T \phi_2\right] \bowtie p$,
$s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{R}_T \phi_2)$ iff $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{R}_T \phi_2\right] \bowtie p$,
$\sigma \models \mathbf{X}\phi$ iff $\sigma(1) \models \phi$,
$\sigma \models \phi_1 \mathbf{U}_T \phi_2$ iff $\exists i \leq T.\ \sigma(i) \models \phi_2 \wedge \left(\forall j < i.\ \sigma(i) \models \phi_1\right)$,
$\sigma \models \phi_1 \mathbf{R}_T \phi_2$ iff $\sigma \not\models \neg\phi_1 \mathbf{U}_T \neg\phi_2$

where $\bowtie \in \{<, >, \leq, \geq\}$. And $\sigma \sim \mathcal{M}_{\Pi,s}$ means the path σ is drawn from the MDP $\mathcal{M}$ under the policy Π, starting from the state s from. For example, the PCTL formulas $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathbf{X}\phi)$ (or $s \models \mathbf{P}^{\min}_{\bowtie p}(\mathbf{X}\phi)$) mean that the maximal (or minimal) satisfaction probability of "next" ϕ is $\bowtie p$. The formulas $s \models \mathbf{P}^{\max}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$ (or $s \models \mathbf{P}^{\min}_{\bowtie p}(\phi_1 \mathbf{U}_T \phi_2)$) mean that the maximal (or minimal) satisfaction probability that ϕ_1 holds "until" ϕ_2 holds is $\bowtie p$.

III. Checking Non-Nested PCTL Formulas

In this work, we consider the statistical model checking of non-nested PCTL formulas using an upper-confidence-bound based Q-learning (nested ones can be handled following [23]). For simplicity, we focus on $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ and $\mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$, where a_1 and a_2 are atomic propositions; other cases can be handled in the same way. Following previous works [3], we make the following assumption.

Assumption 1: For $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ and $s \models \mathbf{P}^{\max}_{\bowtie p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$ with $T \in \mathbb{N} \cup \{\infty\}$ and $\bowtie \in \{<, >, \leq, \geq\}$, we assume that $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{U}_T \phi_2\right] \neq p$ and $\max_{\Pi} \mathbb{P}_{\sigma \sim \mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{R}_T \phi_2\right] \neq p$, respectively.

From Assumption 1, as the number of samples increases, the samples will concentrate on one side of the threshold p by the central limit theorem, so any significance level (¿0) is achievable. Compared to previous works based on sequential probability ratio tests (SPRT) [6], [7], [25], we require no assumption on the indifference region. Finally, by Assumption 1, we have the additional semantic equivalence between the PCTL formulas: $\mathbf{P}^{\max}_{<p}\psi \equiv \mathbf{P}^{\max}_{\leq p}\psi$ and $\mathbf{P}^{\max}_{>p}\psi \equiv \mathbf{P}^{\max}_{\geq p}\psi$; we will not distinguish between them.

For further discussion, we first identify a few trivial cases. For $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$, let

$$\begin{aligned} S_0 &= \{s \in S \mid \mathsf{a}_1 \notin L(s), \mathsf{a}_2 \notin L(s)\} \\ S_1 &= \{s \in S \mid \mathsf{a}_2 \in L(s)\}. \end{aligned} \tag{1}$$

Then for any policy Π, $\mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{U}_T \phi_2\right] = 0$ if $s \in S_0$; and $\mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}\left[\sigma \models \phi_1 \mathbf{U}_T \phi_2\right] = 1$ if $s \in S_1$. The same holds for $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$ by defining S_1 to be the union of end components of the MDP $\mathcal{M}$ labeled by a_2 (this only requires knowing the topology of $\mathcal{M}$) [17]. In the rest of this section, we focus on handling the nontrivial case $s \in S\backslash(S_0 \cup S_1)$ for different values of T.

A. Single Time Horizon

When $T = 1$, for any $s \in S\backslash(S_0 \cup S_1)$, the PCTL formula $\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2$ (or $\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2$) holds on a random path $\sigma : \mathbb{N} \to S$ starting from the state s if and only if $\sigma(1) \in S_1$, where S_0 and S_1 are from (1). So it suffices to learn from samples if

$$\max_{a\in A(s)} Q_1(s,a) > p, \tag{2}$$

where $Q_1(s,a) = \mathbb{P}_{\substack{\sigma(1)\sim T(s,a,\cdot)\\ \sigma(0)=s}}\left[\sigma \models \phi_1 \mathbf{U}_1 \phi_2\right]$ and $\sigma(1) \sim T(s,a,\cdot)$ means $\sigma(1)$ is drawn from the transition probability $T(s,a,\cdot)$ for state s and action a. This is an $|A(s)|$-arm bandit problem; we solve this problem by upper-confidence-bound strategies [26]. Specifically, for the iteration k, let $N^{(k)}(s,a,s')$ be the number samples for the one-step path (s,a,s'). With a slight abuse of notation, let

$$N^{(k)}(s,a) = \sum_{s'\in S} N^{(k)}(s,a,s'). \tag{3}$$

The unknown transition probability function $\mathbf{T}(s,a,s')$ is estimated by the empirical transition probability function

$$\hat{\mathbf{T}}^{(k)}(s,a,s') = \begin{cases} \frac{N^{(k)}(s,a,s')}{N^{(k)}(s,a)}, & \text{if } N^{(k)}(s,a) > 0, \\ \frac{1}{|S|}, & \text{if } N^{(k)}(s,a) = 0. \end{cases} \tag{4}$$

The estimation of $Q_1(s,a)$ from the existing k samples is

$$\hat{Q}_1^{(k)}(s,a) = \sum_{s'\in S_1} \hat{\mathbf{T}}^{(k)}(s,a,s'). \tag{5}$$

Since the value of the Q-function $Q_1(s,a) \in [0,1]$ is bounded, we can construct a confidence interval for the estimate $\hat{Q}_1^{(k)}$ with statistical error at most δ using Hoeffding's inequality by

$$\begin{aligned} \underline{Q}_1^{(k)}(s,a) &= \max\left\{\hat{Q}_1^{(k)}(s,a) - \sqrt{\frac{|\ln(\delta/2)|}{2N^{(k)}(s,a)}}, 0\right\}, \\ \overline{Q}_1^{(k)}(s,a) &= \min\left\{\hat{Q}_1^{(k)}(s,a) + \sqrt{\frac{|\ln(\delta/2)|}{2N^{(k)}(s,a)}}, 1\right\}, \end{aligned} \tag{6}$$

where we set the value of the division to be ∞ for $N^{(k)}(s,a) = 0$.

Remark 1: We use Hoeffding's bounds to yield hard guarantees on the statistical error of the model checking algorithms. Tighter bounds like Bernstein's bounds [27] and Massart's bounds [28], [29] can also be used, but they only yield asymptotic guarantees on the statistical error.

The sample efficiency for learning for the bandit problem (2) depends on the choice of sampling policy, decided from the existing samples. We use the Q-learning from [26]. Specifically, an upper confidence bound (UCB) is constructed for each state-action pair using the number of samples and the observed reward, and the best action is chosen with the highest possible reward, namely the UCB. The sampling policy is chosen by maximizing the possible reward greedily:

$$\pi_1^{(k)}(s) = \text{argmax}_{a\in A(s)} \overline{Q}_1^{(k)}(s,a). \tag{7}$$

The action is chosen arbitrarily when there are multiple candidates. The choice of $\pi_1^{(k)}$ in (7) ensures that the policy giving the upper bound of the value function gets most frequently sampled in the long run.

To initialize the iteration, the Q-function is set to

$$\overline{Q}_1^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \notin S_0, \\ 0, & \text{otherwise,} \end{cases} \quad \underline{Q}_1^{(0)}(s,a) = \begin{cases} 1, & \text{if } s \in S_1, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

to ensure that every state-action is sampled at least once. The termination condition of the above algorithm is

$$\begin{cases} \text{true}, & \text{if } \max_{a\in A(s)} \underline{Q}_1^{(k)}(s) > p, \\ \text{false}, & \text{if } \max_{a\in A(s)} \overline{Q}_1^{(k)}(s) < p, \\ \text{continue}, & \text{otherwise,} \end{cases} \tag{9}$$

where p is from (2).

Remark 2: To handle $s \models \mathbf{P}^{\max}_{<p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{<p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$, it suffices to change the termination condition (9) by returning true if $\overline{Q}_1^{(k)}(s) < p$, and returning false if $\underline{Q}_1^{(k)}(s) > p$. The same statements hold for general PCTL formulas, as discussed in Sections III-B and III-C

Algorithm 1 SMC of $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_1 \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_1 \mathsf{a}_2)$

Require: MDP $\mathcal{M}$, parameter δ.

1: Initialize the Q-function, and the policy by (8)(7).
2: Obtain S_0 and S_1 by (1).
3: **while** True **do**
4: Sample from $\mathcal{M}$, and update the transition probability function by (3)(4).
5: Update the bounds on the Q-function and the policies by (6)(7).
6: Check termination condition (9).
7: **end while**

B. Finite Time Horizon

When $T \in \mathbb{N}$, for any $s \in S\backslash(S_0 \cup S_1)$, let

$$\begin{aligned} V_h(s) &= \max_{\Pi} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}(\sigma \models \mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2), \\ Q_h(s,a) &= \max_{\Pi(s)=a} \mathbb{P}_{\sigma\sim\mathcal{M}_{\Pi,s}}(\sigma \models \mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2), \quad h \in [T], \end{aligned} \tag{10}$$

i.e., $V_h(s)$ and $Q_h(s,a)$ are the maximal satisfaction probability of $\mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2$ for a random path starting from s for any policy and any policy with first action being a, respectively.

Then $V_h(s)$ and $Q_h(s,a)$ satisfy the Bellman equation

$$\begin{aligned} V_h(s) &= \max_{a\in A} Q_h(s,a), \\ Q_{h+1}(s,a) &= \sum_{s'\in S} \mathbf{T}(s,a,s')V_h(s') \\ &= \sum_{s\in S\backslash(S_0\cup S_1)} \mathbf{T}(s,a,s')V_h(s') + \sum_{s'\in S_1} \mathbf{T}(s,a,s'). \end{aligned} \tag{11}$$

The second equality of the second equation is derived from

$$V_h(s) = \begin{cases} 0, & \text{if } s \in S_0, \\ 1, & \text{if } s \in S_1. \end{cases}$$

From (11), we check $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_h \mathsf{a}_2)$ by induction on the time horizon T. For $h \in T$, the lower and upper bounds for $Q_h(s,a)$ can be derived using the bounds on the value function for the previous step: for $h = 1$ from (6) and for $h > 0$ by the following.

$$\begin{aligned} \underline{Q}^{(k)}_{h+1}(s,a) = \max\Bigg\{0, &\sum_{s\in S\backslash(S_0\cup S_1)} \hat{\mathbf{T}}^{(k)}(s,a,s')\underline{V}_h(s') + \\ &\sum_{s'\in S_1} \hat{\mathbf{T}}^{(k)}(s,a,s') - \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}}\Bigg\}, \\ \overline{Q}^{(k)}_{h+1}(s,a) = \max\Bigg\{1, &\sum_{s\in S\backslash(S_0\cup S_1)} \hat{\mathbf{T}}^{(k)}(s,a,s')\underline{V}_h(s') + \\ &\sum_{s'\in S_1} \hat{\mathbf{T}}^{(k)}(s,a,s') + \sqrt{\frac{|\ln(\delta_h/2)|}{2N^{(k)}(s,a)}}\Bigg\}, \end{aligned} \tag{12}$$

and

$$\overline{V}^{(k)}_h(s) = \max_{a\in A(s)} \overline{Q}^{(k)}_h(s,a), \quad \underline{V}^{(k)}_h(s) = \max_{a\in A(s)} \underline{Q}^{(k)}_h(s,a), \tag{13}$$

where δ_h is a parameter such that $Q_h(s,a) \in [\underline{Q}^{(k)}_h(s,a), \overline{Q}^{(k)}_h(s,a)]$ with probability at least $1-\delta_h$. The bounds in (12) are derived from (11) by applying Hoeffding's inequality, using the fact that $\mathbb{E}[\hat{\mathbf{T}}^{(k)}(s,a,s')] = \mathbf{T}(s,a,s')$ and the Q-functions are bounded within $[0,1]$.

By the boundedness of $Q_h(s,a) \in [0,1]$, we note that this confidence interval encompasses the statistical error in both the estimated transition probability function $\hat{\mathbf{T}}^{(k)}(s,a,s')$ and the bounds $\overline{V}^{(k)}_h(s,a)$ and $\underline{V}^{(k)}_h(s,a)$ of the value function. So the policy $\pi^{(k)}_h$ chosen at the h step is

$$\pi^{(k)}_h(s) = \text{argmax}_{a\in A(s)} \overline{Q}^{(k)}_h(s,a), \tag{14}$$

with an optimal action chosen arbitrarily when there are multiple candidates, to ensure that the policy giving the upper bound of the value function is sampled the most in the long run. To initialize the iteration, the Q-function is set to

$$\overline{Q}^{(0)}_h(s,a) = \begin{cases} 1, & \text{if } s \notin S_0 \\ 0, & \text{otherwise,} \end{cases} \quad \underline{Q}^{(0)}_h(s,a) = \begin{cases} 1, & \text{if } s \in S_1 \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

Algorithm 2 SMC of $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U}_T \mathsf{a}_2)$ or $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{R}_T \mathsf{a}_2)$

Require: MDP $\mathcal{M}$, parameters δ_h for $h \in [T]$.

1: Initialize the Q-function and the policy by (15)(14).
2: Obtain S_0 and S_1 by (1).
3: **while** true **do**
4: Sample by (16), and update the transition probability function by (3)(4).
5: Update the bounds by (12)(13) and the policy by (14).
6: Check the termination condition (17).
7: **end while**

for all $h \in [T]$, to ensure that every state-action is sampled at least once.

Sampling by the updated policy $\pi^{(k)}_h(s)$ can be performed in either episodic or non-episodic ways [24]. The only requirement is that the state-action pair $(s, \pi^{(k)}_h(s))$ should be performed frequently for each $h \in [T]$ and for each state s satisfying $s \in S\backslash(S_0 \cup S_1)$. In addition, batch samples may be drawn, namely sampling over the state-action pairs multiple times before updating the policy. In this work, for simplicity, we use a non-episodic, non-batch sampling method, by drawing

$$s' \sim \mathbf{T}(s, \pi^{(k)}_h(s), \cdot), \tag{16}$$

for all $h \in [T]$ and state s such that $\mathsf{a}_1 \in L(s), \mathsf{a}_2 \notin L(s)$. The Q-function and the value function are set and initialized by (13) and (15). The termination condition is give by

$$\mathbf{P}^{\max}_{>p}\phi : \begin{cases} \text{false}, & \text{if } \overline{V}^{(k)}_H(s_0) < p, \\ \text{true}, & \text{if } \underline{V}^{(k)}_H(s_0) > p, \\ \text{continue}, & \text{otherwise,} \end{cases} \tag{17}$$

where p is the probability threshold in the non-nested PCTL formula. The above discussion is summarized by Algorithm 2 and Theorem 1.

Theorem 1: Algorithm 2 terminates with probability 1 and its return value is correct with probability at least $1 - N|A|\sum_{h\in[T]} \delta_h$, where $N = |S\backslash(S_0 \cup S_1)|$.

By Theorem 1, the desired overall statistical error splits into the statistical errors for each state-action pair through the time horizon. For implementation, we can split it equally by $\delta_1 = \cdots = \delta_H$. The formula $\mathbf{P}^{\min}_{\bowtie p}(\phi)$ can be handled by replacing argmax with argmin in (14), and $\max$ with $\min$ in (13). The termination condition is the same as (17).

Remark 3: From the semantics of PCTL, running Algorithm 2 proving $\mathbf{P}^{\max}_{>p}(\phi)$ or disproving $\mathbf{P}^{\max}_{<p}(\phi)$ is easier than disproving $\mathbf{P}^{\max}_{>p}(\phi)$ or proving $\mathbf{P}^{\max}_{<p}(\phi)$; and the difference increases with the number of actions $|A|$ and the time horizon T. This is because proving $\mathbf{P}^{\max}_{>p}(\phi)$ or disproving $\mathbf{P}^{\max}_{<p}(\phi)$ requires only finding and evaluating some policy Π with $\mathbb{P}_{\mathcal{M}_\Pi}[s \models \phi] > p$, while disproving it requires evaluating all possible policies with sufficient accuracy. This is illustrated by the simulation results presented in Section IV.

Algorithm 3 SMC of $s \models \mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$

Require: MDP $\mathcal{M}$, parameters δ_h for $h \in \mathbb{N}$.

1: Initialize two sets of Q-function and the policy by (15)(14) for **(i)** $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$ and **(ii)** $\mathbf{P}^{\min}_{>1-p}(\neg \mathsf{a}_1 \mathbf{R} \neg \mathsf{a}_2)$, respectively.
2: Obtain S_0 and S_1 for **(i)** and **(ii)** respectively by (1).
3: **while** True **do**
4: Sample by (16), and update $\hat{\mathbf{T}}^{(k)}(s,a,s')$ by (3)(4).
5: Update the bounds on the Q-function, the policies, the value function, and the time horizon by (12)(14)(13)(20) respectively for **(i)** and **(ii)**.
6: Check the termination condition (19).
7: **end while**

C. Infinite Time Horizon

Infinite-step satisfaction probability can be estimated from finite-step satisfaction probabilities, using the monotone convergence of the value function in the time step H,

$$V_0(s) \leq \ldots \leq V_H(s) \leq \ldots \leq V(s) = \lim_{H \to \infty} V_H(s). \quad (18)$$

Therefore, if the satisfaction probability is larger than p for some step H, then the statistical model checking algorithm should terminate, namely,

$$\begin{cases} \text{false}, & \text{if } \underline{V}^{(k)}_H(s_0) > p, \\ \text{continue}, & \text{otherwise.} \end{cases} \quad (19)$$

If we check both $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$ and its negation $\mathbf{P}^{\min}_{>1-p}(\neg \mathsf{a}_1 \mathbf{R} \neg \mathsf{a}_2)$ simultaneously, then one of them should terminate in finite time. The termination in finite time is guaranteed, if the time horizon for both computations increases.

We use the convergence of the best policy as the criterion for increasing the time step H for checking the formula or its negation. For each H, besides finding the optimal policy $\pi^{(k)}_H(s)$ for the upper confidence bounds of the Q-functions $\overline{Q}^{(k)}_H(s,a)$ by (14), we also consider the optimal policy for the lower confidence bounds of the Q-functions $\underline{Q}^{(k)}_H(s,a)$. Obviously, when $\pi^{(k)}_H(s) \in \operatorname{argmax}_{a \in A} \underline{Q}^{(k)}_H(s,a)$, we know that the policy $\pi^{(k)}_H(s)$ is optimal for all possible Q-functions within $[\underline{Q}^{(k)}_H, \overline{Q}^{(k)}_H]$. This implies that these bounds are fine enough for estimating Q_H; thus, if the algorithm does not terminate by the condition (19), we let

$$H \leftarrow \begin{cases} 1, & \text{initially}, \\ H+1, & \text{if } \pi^{(k)}_H(s) \in \operatorname{argmax}_{a \in A} \underline{Q}^{(k)}_H(s,a) \\ & \quad \text{for all } s \in S, \\ \text{Continue}, & \text{otherwise.} \end{cases} \quad (20)$$

Combining the above procedure, we derive Algorithm 3 and Theorem 2 below for statistically model checking PCTL formula $\mathbf{P}^{\max}_{>p}(\mathsf{a}_1 \mathbf{U} \mathsf{a}_2)$.

Theorem 2: Algorithm 3 terminates with probability 1 and its return value is correct with probability $1 - |A| \max\{N_1, N_2\} \sum_{h \in [T]} \delta_h$, where H is the largest time horizon when the algorithm stops, $N_1 = |S \backslash (S^{\phi}_0 \cup S^{\phi}_1)|$ and $N_2 = |S \backslash (S^{\psi}_0 \cup S^{\psi}_1)|$ with $S^{\phi}_0 \cup S^{\phi}_1$ and $S^{\psi}_0 \cup S^{\psi}_1$ derived from (1) for $\phi = \mathsf{a}_1 \mathbf{U} \mathsf{a}_2$ and $\psi = \mathsf{a}_1 \mathbf{R} \mathsf{a}_2$, respectively.

Remark 4: By Theorem 2, given the desired overall confidence level δ, we can split it geometrically by $\delta_h = (1-\lambda)\lambda^{h-1}\delta$, where $\lambda \in (0,1)$.

Remark 5: Similar to Section III-B, checking $\mathbf{P}^{\min}_{\sim p}(\phi)$ for $\sim \in \{<, >, \leq, \geq\}$ is derived by replacing argmax with argmin in (14), and $\max$ with $\min$ in (13). The termination condition is the same as (19).

Remark 6: Finally, we note that the exact savings of sample costs for Algorithms 2 and 3 depend on the structure of the MDP. Specifically, the proposed method is more efficient than [10], [11], when the satisfaction probabilities differ significantly among actions, as it can quickly detect sub-optimal actions without over-sampling on them. On the other hand, if all the state-action pairs yield the same Q-value, then an equal number of samples will be spent on each of them — in this case, the sample cost of Algorithms 2 and 3 is the same as [10], [11].

IV. Simulation

We implement Algorithms 2 and 3 in Scala on Ubuntu 18.04 with i7-8700 CPU 3.2GHz and 16GB memory. They are evaluated on two sets of MDPs with known topology but unknown transition probabilities (different from [10]) on 10 randomly generated MDPs with different sizes, where we check a finite-horizon formula $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$ and an infinite-horizon formula $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U} \alpha_2)$ for some given α_1 and α_2. For each MDP, we test two different thresholds p with the (nominal) significance level $\delta = 0.05$. To estimate the average running time and number of iterations for, we repeated each model checking task for 100 times. For each task, the repeated runs return the same answers, showing that the actual significance level of the proposed algorithms is much smaller than 0.05; this is because of the conservativeness of the Hoeffding's bounds used in the algorithms All the checking results agree with the satisfaction probabilities from PRISM [30], which requires the transition probabilities of the MDP (so we do not compare the running time).

Table I show the results for finite horizons. In all examples, returning "False" is 3 to 100 times faster than returning "True". We believe this is because, to return "False", the algorithm only needs to find a falsifying policy; but to return "True", it needs to check for all policies (see Remark 3). Table II show the results for infinite horizons. Since Algorithm 3 checks both the formula and its negation, returning "False" is not faster than returning "True". For larger MDPs, the time steps H_1 and H_2 needed to model check the formula and its negation (updated by (20)) are very small on average, showing that the algorithm can quickly decide that there is no need to increase the time horizon to solve the problem.

V. Conclusion

We proposed a statistical model checking method for Probabilistic Computation Tree Logic on Markov decision processes using reinforcement learning. We first checked

$\|S\|$	$\|A\|$	H	p	PRISM	Ans.	Iter.	Time (s)
3	3	4	0.25	0.35	False	208.5	0.02
		4	0.45		True	3528.7	0.19
4	2	4	0.09	0.19	False	171.8	0.01
		4	0.29		True	3671.7	0.22
5	2	4	0.05	0.11	False	441.7	0.04
		4	0.21		True	4945.5	0.42
10	2	4	0.04	0.09	False	544.7	0.14
		4	0.19		True	5193.3	1.45
15	2	3	0.04	0.09	False	873.1	0.28
		3	0.19		True	4216.7	1.28
20	4	5	0.12	0.22	False	337.6	0.99
		5	0.32		True	9353.3	28.06
25	5	10	0.09	0.19	False	270.5	7.57
		10	0.29		True	25709.8	728.49
30	5	10	0.08	0.17	False	355.6	14.35
		10	0.27		True	27161.7	1085.77
35	5	10	0.09	0.18	False	328.9	18.82
		10	0.28		True	27369.6	1529.84
40	5	10	0.08	0.16	False	390.0	26.79
		10	0.26		True	30948.8	2122.24

TABLE I

CHECKING $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U}_H \alpha_2)$. "ANS." IS RETURN OF ALGORITHM 2. "ITER." IS AVERAGE NUMBER OF ITERATIONS. "PRISM" IS PRISM'S ESTIMATION OF SATISFACTION PROBABILITY.

$\|S\|$	$\|A\|$	p	PRISM	Ans.	H_1	H_2	Iter.	Time (s)
3	3	0.25	0.35	False	14.7	15.8	126.0	0.03
		0.45		True	12.9	13.0	111.3	0.01
4	2	0.09	0.19	False	13.0	27.2	103.7	0.01
		0.29		True	9.6	19.2	73.0	0.01
5	2	0.05	0.11	False	12.4	59.3	239.9	0.06
		0.21		True	6.9	24.3	92.7	0.00
10	2	0.04	0.09	False	3.2	225.6	891.4	0.24
		0.19		True	1.1	21.5	79.6	0.00
15	2	0.04	0.09	False	1.3	117.8	1862.0	0.61
		0.19		True	1.0	11.0	161.3	0.01
20	4	0.12	0.22	False	1.0	1.0	1336.2	0.27
		0.32		True	1.0	1.8	16843.8	3.02
25	5	0.09	0.19	False	1.0	1.0	1619.2	0.64
		0.29		True	1.0	2.0	154621.6	56.56
30	5	0.08	0.17	False	1.0	1.0	2246.8	1.05
		0.27		True	1.0	1.3	86617.0	42.53
35	5	0.09	0.18	False	1.0	1.0	1925.1	0.82
		0.28		True	1.0	1.0	13219.0	5.79
40	5	0.08	0.16	False	1.0	1.0	2401.5	1.35
		0.26		True	1.0	1.0	12158.6	6.66

TABLE II

CHECKING $\mathbf{P}^{\max}_{<p}(\alpha_1 \mathbf{U} \alpha_2)$. H_1 AND H_2 ARE THE TIME STEPS NEEDED TO MODEL CHECK THE FORMULA AND ITS NEGATION, UPDATED BY (20).

PCTL formulas with bounded time horizon, using upper-confidence-bounds based Q-learning, and then extended the technique to unbounded-time specifications by finding a proper truncation time by checking the specification of interest and its negation at the same time. Finally, we demonstrated the efficiency of our method by case studies.

REFERENCES

[1] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen, "Model-checking algorithms for continuous-time Markov chains," *IEEE Transactions on Software Engineering*, vol. 29, no. 6, pp. 524–541, 2003.

[2] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*. Springer, 2018.

[3] G. Agha and K. Palmskog, "A Survey of Statistical Model Checking," *ACM Trans. Model. Comput. Simul.*, vol. 28, no. 1, pp. 6:1–6:39, 2018.

[4] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[5] H. Kress-Gazit, M. Lahijanian, and V. Raman, "Synthesis for robots: Guarantees and feedback for robot behavior," *Annu. Rev. of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 211–236, 2018.

[6] Y. Wang, N. Roohi, M. West, M. Viswanathan, and G. E. Dullerud, "Statistical verification of PCTL using stratified samples," in *IFAC Conference on Analysis and Design of Hybrid Systems*, 2018, pp. 85–90.

[7] Y. Wang, N. Roohi, M. West, M. Viswanathan, and G. E. Dullerud, "Statistical verification of PCTL using antithetic and stratified samples," *Formal Methods in System Design*, vol. 45, no. 2, pp. 145–163, 2019.

[8] A. K. Bozkurt, Y. Wang, M. Zavlanos, and M. Pajic, "Control synthesis from linear temporal logic specifications using model-free reinforcement learning," in *IEEE Int. Conf. on Robotics and Automation*, 2020.

[9] D. Henriques, J. G. Martins, P. Zuliani, A. Platzer, and E. M. Clarke, "Statistical Model Checking for Markov Decision Processes," in *Quantitative Evaluation of Systems*, 2012, pp. 84–93.

[10] T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínský, M. Kwiatkowska, D. Parker, and M. Ujma, "Verification of Markov Decision Processes Using Learning Algorithms," in *Automated Technology for Verification and Analysis*, 2014, pp. 98–114.

[11] J. Fu and U. Topcu, "Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints," *arXiv:1404.7073 [cs]*, 2014.

[12] P. Ashok, J. Křetínský, and M. Weininger, "PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games," *arXiv:1905.04403 [cs]*, 2019.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT Press, 2018.

[14] X. Li, C.-I. Vasile, and C. Belta, "Reinforcement learning with temporal logic rewards," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 3834–3839.

[15] A. Jones, D. Aksaray, Z. Kong, M. Schwager, and C. Belta, "Robust satisfaction of temporal logic specifications via reinforcement learning," *arXiv:1510.06460 [cs]*, 2015.

[16] M. L. Littman, U. Topcu, J. Fu, C. Isbell, M. Wen, and J. MacGlashan, "Environment-independent task specifications via GLTL," *arXiv:1704.04341 [cs]*, 2017.

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

[18] E. M. Hahn, M. Perez, S. Schewe, F. Somenzi, A. Trivedi, and D. Wojtczak, "Omega-regular objectives in model-free reinforcement learning," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2019, pp. 395–412.

[19] 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 *IEE Conf. on Decision and Control*, 2019, pp. 5338–5343.

[20] E. M. Hahn, G. Li, S. Schewe, A. Turrini, and L. Zhang, "Lazy probabilistic model checking without determinisation," in *Int. Conf. on Concurrency Theory*, 2015, p. 354.

[21] D. Kini and M. Viswanathan, "Complexity of model checking MDPs against LTL specifications," in *Foundations of Software Technology and Theoretical Computer Science*, 2017, pp. 35:1–35:13.

[22] ——, "Limit deterministic and probabilistic automata for LTL\GU," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2015, pp. 628–642.

[23] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *Computer Aided Verification*, 2005, pp. 266–280.

[24] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.

[25] P. Zuliani, "Statistical model checking for biological applications," *Software Tools for Tech. Transfer*, vol. 17, no. 4, pp. 527–536, 2015.

[26] S. Bubeck, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[27] V. Mnih, C. Szepesvári, and J.-Y. Audibert, "Empirical Bernstein stopping," in *Int. Conf. on Machine Learning*, 2008, pp. 672–679.

[28] C. Jegourel, J. Sun, and J. S. Dong, "On the sequential Massart algorithm for statistical model checking," in *Int. Sym. on Leveraging Applications of Formal Methods*, 2018, pp. 287–304.

[29] L. Cardelli, M. Kwiatkowska, L. Laurenti, N. Paoletti, A. Patane, and M. Wicker, "Statistical guarantees for the robustness of Bayesian neural networks," in *Int. Joint Conf. on Artificial Intelligence*, 2019, pp. 5693–5700.

[30] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification*, 2011, pp. 585–591.