2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Temporal Logic Motion Control using Actor-Critic Methods *

Xu Chu Ding†, Jing Wang‡, Morteza Lahijanian‡, Ioannis Ch. Paschalidis‡, and Calin A. Belta‡

Abstract— **In this paper, we consider the problem of deploying a robot from a specification given as a temporal logic statement about some properties satisfied by the regions of a large, partitioned environment. We assume that the robot has noisy sensors and actuators and model its motion through the regions of the environment as a Markov Decision Process (MDP). The robot control problem becomes finding the control policy maximizing the probability of satisfying the temporal logic task on the MDP. For a large environment, obtaining transition probabilities for each state-action pair, as well as solving the necessary optimization problem for the optimal policy are usually not computationally feasible. To address these issues, we propose an approximate dynamic programming framework based on a least-square temporal difference learning method of the actor-critic type. This framework operates on sample paths of the robot and optimizes a randomized control policy with respect to a small set of parameters. The transition probabilities are obtained only when needed. Hardware-in-the-loop simulations confirm that convergence of the parameters translates to an approximately optimal policy.**

I. Introduction

One major goal in robot motion planning and control is to specify a mission task in an expressive and high-level language and convert the task automatically to a control strategy for the robot. The robot is subject to mechanical constraints, actuation and measurement noise, and limited communication and sensing capabilities. The challenge in this area is the development of a computationally efficient framework accommodating both the robot constraints and the uncertainty of the environment, while allowing for a large spectrum of task specifications. Temporal logics such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) have been promoted as formal task specification languages for robotic applications [1]–[4], due to their high expressivity and closeness to human language.

In this paper, we assume that the robot model in the environment is described by a (finite) Markov Decision Process (MDP). In this model, the robot can precisely determine its current state, and by applying an action (corresponding to a motion primitive) enabled at each state, it triggers a transition to an adjacent state with a fixed probability. We are interested in controlling the MDP robot model such that it maximizes the probability of satisfying a temporal logic formula over a set of properties satisfied at the states of the MDP. By adapting existing probabilistic model checking [5]–[7] and synthesis [8], [9] algorithms, we recently developed such computational frameworks for formulas of LTL [10] and a fragment of probabilistic CTL [11].

These approaches assumed that the transition probabilities are known for each state-action pair of the MDP, which can be computed by using a Monte-Carlo method and repeated forward simulations. However, this is often not feasible for realistic robotic applications, even if an accurate model or a simulator of the robot in the environment is available. The problem is even more challenging when considering temporal logic specifications, due to the size of the automata corresponding to these specifications.

In this paper, we show that approximate dynamic programming can be effectively used to address the above limitations. For large dynamic programming problems, an approximately optimal solution can be provided using actor-critic algorithms [12]. In particular, actor-critic algorithms with Least Squares Temporal Difference (LSTD) learning have been shown recently to be a powerful tool to solve large-sized problems [13], [14]. This paper extends from [15], in which we proposed an actor-critic method for maximal reachability (MRP) problems, *i.e.,* maximizing the probability of reaching a set of states, to a computational framework that finds a control policy such that the probability of its paths satisfying an arbitrary LTL formula is locally optimal over a set of parameters. This set of parameters is designed to tailor to this class of approximate dynamical programming problems.

Our method requires transition probabilities to be generated only along sample paths, and is therefore particularly suitable for robotic applications. To the best of our knowledge, this is the first attempt to combine temporal logic formal synthesis with actor-critic type methods. We illustrate the algorithms with hardware-in-the-loop simulations using an accurate simulator of our Robotic InDoor Environment (RIDE) platform (see Fig. 1). We omit proofs in this paper due to space constraints. Full proofs and extra details of this paper can be found in the technical report [16].

Notation: We use bold letters to denote sequences and vectors. Vectors are assumed to be column vectors. Transpose of a vector $\mathbf{x}$ is denoted by $\mathbf{x}^\intercal$. $\|\cdot\|$ stands for the Euclidean norm. $|S|$ denotes the cardinality of a set S.

II. Problem Formulation and Approach

We consider a robot moving in an environment partitioned into regions such as the Robotic Indoor Environment (RIDE) (Fig. 1). Each region in the environment is associated with a set of observations. Observations can be **Un** for unsafe regions, or **Up** for a region where the robot can upload

* Research partially supported by the NSF under grants CNS-1035588 and CMMI-0928776, by the DOE under grant DE-FG52-06NA27490, by the ONR MURI under grants N00014-10-1-0952 and N00014-09-1051, by the AFOSR under grant FA9550-09-1-0209, and by the ARO under grants W911NF-09-1-0088 and W911NF-11-1-0227.

† Xu Chu Ding is with Embedded Systems and Networks group, United Technologies Research Center, East Hartford, CT 06108 (dingx@utrc.utc.com).

Jing Wang, Morteza Lahijanian, Ioannis Ch. Paschalidis, and Calin A. Belta are with the Division of System Eng., Dept. of Mechanical Eng., Dept. of Electrical & Computer Eng., and Dept. of Mechanical Eng., Boston University, Boston, MA 02215 ({wangjing,morteza, yannisp, cbelta}@bu.edu), respectively.

978-1-4673-1405-3/12/$31.00 ©2012 IEEE

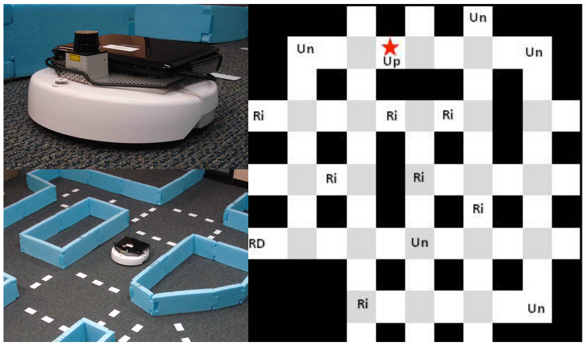


Fig. 1. Robotic InDoor Environment (RIDE) platform. Left: An iCreate mobile platform moving autonomously through the corridors and intersections of an indoor-like environment. Right: The partial schematics of the environment. The black blocks represent walls, and the grey and white regions are intersection and corridors, respectively. The labels inside a region represents observations associated with regions, such as **Un** (unsafe regions) and **Ri** (risky regions).

data. We assume that the robot can detect its current region. Moreover, the robot is programmed with a set of motion primitives allowing it to move from a region to an adjacent region. To capture noise in actuation and sensing, we make the natural assumption that, at a given region, a motion primitive designed to take the robot to a specific adjacent region may take the robot to a different adjacent region.

Such a robot model naturally leads to a labeled Markov Decision Process (MDP), which is defined below.

Definition II.1 (Labeled Markov Decision Process)**.** *A labeled Markov decision process (MDP) is a tuple* $\mathcal{M} = (Q, q_0, U, A, P, \Pi, h)$, *where*

(i) $Q = \{1, \ldots, n\}$ *is a finite set of states;*
(ii) $q_0 \in Q$ *is the initial state;*
(iii) U *is a finite set of actions;*
(iv) $A : Q \to U$ *maps a state* $q \in Q$ *to actions enabled at* q*;*
(v) $P : Q \times U \times Q \to [0, 1]$ *is the transition probability function such that for all* $q \in Q$, $\sum_{q' \in Q} P(q, u, q') = 1$ *if* $u \in A(q)$, *and* $P(q, u, q') = 0$ *for all* $q' \in Q$ *if* $u \notin A(q)$*;*
(vi) Π *is a set of observations;*
(vii) $h : Q \to 2^{\Pi}$ *is the observation map.*

Each state of the MDP $\mathcal{M}$ modeling the robot in the environment corresponds to an ordered set of regions in the environment, while the actions label the motion primitives that can be applied at a region. For example, a state of $\mathcal{M}$ may be labeled as I_1-C_1, which means that the robot is currently at region C_1, coming from region I_1. Each ordered set of regions corresponds to a recent history of the robot trajectory, and is needed to ensure the Markov property (more details on such MDP abstraction of the robot in the environment can be found in, *e.g.,* [11]; for our hardware-in-the-loop set-up, see Sec. IV). The transition probability function P can be obtained through extensive simulations of the robot in the environment. We assume that there exists an accurate simulator that is capable of generating (computing) the transition probability $P(q, u, \cdot)$ for each state-action pair $q \in Q$ and $u \in A(q)$.

If the exact transition probabilities are not known, $\mathcal{M}$ can be seen as a labeled non-deterministic transition system (NTS) $\mathcal{M}^{\mathcal{N}} = (Q, q_0, U, A, P^{\mathcal{N}}, \Pi, h)$, where P in $\mathcal{M}$ is replaced by $P^{\mathcal{N}} : Q \times U \times Q \to \{0, 1\}$, and $P^{\mathcal{N}}(q, u, q') = 1$ indicates a possible transition from q to q' applying an enabled action $u \in A(q)$; if $P^{\mathcal{N}}(q, u, q') = 0$, then the transition from q to q' is not possible under u.

A path on $\mathcal{M}$ is a sequence of states $\mathbf{q} = q_0 q_1 \ldots$ such that for all $k \geq 0$, there exists $u_k \in A(q_k)$ such that $P(q_k, u_k, q_{k+1}) > 0$. Along a path $\mathbf{q} = q_0 q_1 \ldots$, q_k is said to be the state at time k. The trajectory of the robot in the environment is represented by a path $\mathbf{q}$ on $\mathcal{M}$ (which corresponds to a sequence of regions in the environment). A path $\mathbf{q} = q_1 q_2 \ldots$ generates a sequence of observations $\mathbf{h}(\mathbf{q}) := o_1 o_2 \ldots$, where $o_k = h(q_k)$ for all $k \geq 0$. We call $\mathbf{o} = \mathbf{h}(\mathbf{q})$ the word generated by $\mathbf{q}$.

Definition II.2 (Policy)**.** *A control policy for an MDP* $\mathcal{M}$ *is an infinite sequence* $M = \mu_0 \mu_1 \ldots$, *where* $\mu_k : Q \times U \to [0, 1]$ *is such that* $\sum_{u \in A(q)} \mu_k(q, u) = 1$, *for all* $k \geq 0$.

Namely, at time k, $\mu_k(q, \cdot)$ is a discrete probability distribution over $A(q)$. If $\mu = \mu_k$ for all $k \geq 0$, then $M = \mu\mu \ldots$ is called a *stationary* policy. If for all $k \geq 0$, $\mu_k(q, u) = 1$ for some u, then M is *deterministic*; otherwise, M is *randomized*. Given a policy M, we can then generate a set of paths on $\mathcal{M}$, by applying u_k with probability $\mu_k(q_k, u_k)$ at state q_k for all time k.

We require the trajectory of the robot in the environment to satisfy a rich task specification given as a Linear Temporal Logic (LTL) (see, *e.g.,* [5]) formula over a set of observations Π. An LTL formula over Π is evaluated over an infinite sequence $\mathbf{o} = o_0 o_1 \ldots$ (*e.g.,* a word generated by a path on $\mathcal{M}$), where $o_k \subseteq \Pi$ for all $k \geq 0$. We denote $\mathbf{o} \vDash \phi$ if word $\mathbf{o}$ satisfies the LTL formula ϕ, and we say $\mathbf{q}$ satisfies ϕ if $\mathbf{h}(\mathbf{q}) \vDash \phi$. Roughly, ϕ can be constructed from a set of observations Π, Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\longrightarrow$ (implication), and temporal operators X (next), U (until), F (eventually), G (always). A variety of complex robotic tasks can be easily translated to LTL formulas. An example of a complex mission task specified as an LTL specification is given in Sec. IV.

We can then formulate the following problem: Given a labeled MDP $\mathcal{M} = (Q, q_0, U, A, P, \Pi, h)$ modeling the motion of a robot in a partitioned environment and a task specified as an LTL formula ϕ over Π; Find a control policy that maximizes the probability of its paths satisfying ϕ.

In [10], we proposed a computational framework to solve this problem by adapting methods from the area of probabilistic model checking [5]–[7]. However, as mentioned in Sec. I, this framework relies upon the fact that the transition probabilities are known for all state-action pairs. Even if the transition probabilities are obtained for each state-action pair, computing the exact solution still requires solving a linear program on the product of the MDP and the automata representing the formula, which might be very large (thousands or even millions of states). An approximate method might be more desirable in such cases. For these reasons, we focus on the following problem in this paper.

Problem II.3. *Given a labeled NTS $\mathcal{M}^{\mathcal{N}} = (Q, q_0, U, A, P^{\mathcal{N}}, \Pi, h)$ modeling a robot in a partitioned environment, a mission task specified as an LTL formula ϕ over Π, and an accurate simulator to compute transition probabilities $P(q, u, \cdot)$ given a state-action pair (q, u); Find a control policy that approximately maximizes the probability of its path satisfying ϕ.*

In many robotic applications, the NTS model can be quickly constructed for the robot in the environment. Our approach to Prob. II.3 can be summarized as follows: First, we proceed to translate the problem to a maximal reachability probability (MRP) problem using $\mathcal{M}^{\mathcal{N}}$ and ϕ (Sec. III-A). Given a randomized policy as a function of a set of parameters, we use an actor critic framework to find a locally optimal policy (Sec. III-B). The design of this randomized policy suitable for Prob. II.3 is detailed in Sec. III-C. The algorithmic framework presented in this paper is finally summarized in Sec. III-D.

III. Control Synthesis

A. Formulation of the MRP Problem

The formulation of the MRP problem is based on [5]–[7], [10] with modification if needed when using the NTS $\mathcal{M}_{\mathcal{N}}$ instead of $\mathcal{M}$. We start by converting the LTL formula ϕ over Π to a so-called deterministic *Rabin automaton*.

Definition III.1 (Deterministic Rabin Automaton)**.** *A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (S, s_0, \Sigma, \delta, F)$, where*

(i) S is a finite set of states;
(ii) $s_0 \in S$ is the initial state;
(iii) Σ is a set of inputs (alphabet);
(iv) $\delta : S \times \Sigma \to S$ is the transition function;
(v) $F = \{(L(1), K(1)), \ldots, (L(M), K(M))\}$ is a set of pairs of sets of states such that $L(i), K(i) \subseteq S$ for all $i = 1, \ldots, M$.

A run of a Rabin automaton $\mathcal{R}$, denoted by $\mathbf{r} = s_0 s_1 \ldots$, is an infinite sequence of states in $\mathcal{R}$ such that for each $k \geq 0$, $s_{k+1} \in \delta(s_k, \alpha)$ for some $\alpha \in \Sigma$. A run $\mathbf{r}$ is *accepting* if there exists a pair $(L, K) \in F$ such that $\mathbf{r}$ intersects with L finitely many times and K infinitely many times. For any LTL formula ϕ over Π, one can construct a DRA (for which we denote by $\mathcal{R}_\phi$) with input alphabet $\Sigma = 2^\Pi$ accepting all and only words over Π that satisfy ϕ (see [5]).

We then obtain an MDP as the product of $\mathcal{M}$ and $\mathcal{R}_\phi$, which captures all paths of $\mathcal{M}$ satisfying ϕ.

Definition III.2 (Product MDP)**.** *The product MDP $\mathcal{M} \times \mathcal{R}_\phi$ between a labeled MDP $\mathcal{M} = (Q, q_0, U, A, P, \Pi, h)$ and a DRA $\mathcal{R}_\phi = (S, s_0, 2^\Pi, \delta, F)$ is an MDP $\mathcal{P} = (S_{\mathcal{P}}, s_{\mathcal{P}0}, U_{\mathcal{P}}, A_{\mathcal{P}}, P_{\mathcal{P}}, \Pi, h_{\mathcal{P}})$, where*

(i) $S_{\mathcal{P}} = Q \times S$ is a set of states;
(ii) $s_{\mathcal{P}0} = (q_0, s_0)$ is the initial state;
(iii) $U_{\mathcal{P}} = U$ is a set of actions inherited from $\mathcal{M}$;
(iv) $A_{\mathcal{P}}$ is also inherited from $\mathcal{M}$ and $A_{\mathcal{P}}((q, s)) := A(q)$;
(v) $P_{\mathcal{P}}$ gives the transition probabilities:

$$P_{\mathcal{P}}((q, s), u, (q', s')) = \begin{cases} P(q, u, q') & \textit{if } q' = \delta(s, h(q)) \\ 0 & \text{otherwise}; \end{cases}$$

Note that $h_{\mathcal{P}}$ is not used in the product MDP. Moreover, $\mathcal{P}$ is associated with pairs of accepting states (similar to a DRA) $F_{\mathcal{P}} := \{(L_{\mathcal{P}}(1), K_{\mathcal{P}}(1)), \ldots, (L_{\mathcal{P}}(M), K_{\mathcal{P}}(M))\}$ where $L_{\mathcal{P}}(i) = Q \times L(i)$, $K_{\mathcal{P}}(i) = Q \times K(i)$, for $i = 1, \ldots, M$.

The product MDP is constructed in a ways such that, given a path $(s_0, q_0)(s_1, q_1) \ldots$, the corresponding path $s_0 s_1 \ldots$ on $\mathcal{M}$ satisfies ϕ if and only if there exists a pair $(L_{\mathcal{P}}, K_{\mathcal{P}}) \in F_{\mathcal{P}}$ satisfying the Rabin acceptance condition.

We can make a very similar product between a labeled NTS $\mathcal{M}^{\mathcal{N}} = (Q, q_0, U, A, P^{\mathcal{N}}, \Pi, h)$ and $\mathcal{R}_\phi$. This product is also an NTS, which we denote by $\mathcal{P}^{\mathcal{N}} = (S_{\mathcal{P}}, s_{\mathcal{P}0}, U_{\mathcal{P}}, A_{\mathcal{P}}, P_{\mathcal{P}}^{\mathcal{N}}, \Pi, h_{\mathcal{P}}) := \mathcal{M}^{\mathcal{N}} \times \mathcal{R}_\phi$, associated with accepting sets $F_{\mathcal{P}}$. The definition (and the accepting condition) of $\mathcal{P}^{\mathcal{N}}$ is exactly the same as for the product MDP $\mathcal{P}$. The only difference between $\mathcal{P}^{\mathcal{N}}$ and $\mathcal{P}$ is in $P_{\mathcal{P}}^{\mathcal{N}}$, which is either 0 or 1 for every state-action-state tuple.

From $\mathcal{P}$ or equivalently $\mathcal{P}^{\mathcal{N}}$, we can proceed to construct the MRP problem. To do so, it is necessary to produce the so-called *accepting maximum end components* (AMECs). An end component is a subset of an MDP (consisting of a subset of states and a subset of enabled actions at each state) such that for each pair of states (i, j) in $\mathcal{P}$, there is a sequence of actions such that i can be reached from j with positive probability, and states outside the component cannot be reached. An AMEC of $\mathcal{P}$ is the largest end component containing at least one state in $K_{\mathcal{P}}$ and no state in $L_{\mathcal{P}}$, for a pair $(K_{\mathcal{P}}, L_{\mathcal{P}}) \in F_{\mathcal{P}}$.

A procedure to obtain all AMECs of an MDP is outlined in [5]. This procedure is intended to be used for the product MDP $\mathcal{P}$, but it can be used without modification to find all AMECs associated with $\mathcal{P}$ when $\mathcal{P}^{\mathcal{N}}$ is used instead of $\mathcal{P}$. This is because the information needed to construct the AMECs is the set of none-zero probability transitions at each state, and this information is already contained in $\mathcal{P}^{\mathcal{N}}$.

If we denote $S_{\mathcal{P}}^\star$ as the union of all states in all AMECs associated with $\mathcal{P}$, it has been shown in probabilistic model checking (see *e.g.,* [5]) that the probability of satisfying the LTL formula is given by the maximal probability of reaching the set $S_{\mathcal{P}}^\star$ from the initial state $S_{\mathcal{P}0}$ (an MRP problem). The desired optimal policy can then be obtained as the policy maximizing this probability. If transition probabilities are available for each state-action pair, then the solution to this MRP problem can be solved as by a linear program (see [5]). The resultant optimal policy is a stationary policy $M = \mu\mu\ldots$ defined on the product MDP $\mathcal{P}$. To implement this policy on $\mathcal{M}$, it is necessary to use the DRA as a feedback automaton to keep track of the current state $s_{\mathcal{P}}$ on $\mathcal{P}$, and apply the action u where $\mu(s_{\mathcal{P}}, u) = 1$ (since μ is deterministic).

B. LSTD Actor-Critic Method

We now describe how relevant results in [15] can be applied to solve Prob. II.3. An approximate dynamic programming algorithm of the actor-critic type was presented in [15], which obtains a stationary randomized policy (RSP) (see Def. II.2) $M = \mu_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}\ldots$, where $\mu_{\boldsymbol{\theta}}(q, u)$ is a function of the state-action pair (q, u) and $\boldsymbol{\theta} \in \mathbb{R}^n$, which is a vector of parameters. For the convenience of notations, we denote an

RSP $\mu_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}\ldots$ simply by $\mu_{\boldsymbol{\theta}}$. In this sub-section we assume that the RSP $\mu_{\boldsymbol{\theta}}(q,u)$ to be given, and we will describe in Sec. III-C on how to design a suitable RSP.

Given an RSP $\mu_{\boldsymbol{\theta}}$, actor-Critic algorithms can be applied to optimize the parameter vector $\boldsymbol{\theta}$ by policy gradient estimations. The basic idea is to use stochastic learning techniques to find $\boldsymbol{\theta}$ that locally optimizes a cost function. In particular, the algorithm presented in [15] is targeted at Stochastic Shortest Path (SSP) problems commonly studied in literature (see *e.g.,* [17]). Given an MDP $\mathcal{M}=(Q,q_0,U,A,P,\Pi,h)$, a termination state $q^\star\in Q$ and a function $g(q,u)$ defining the one-step cost of applying action u at state q, the *expected total cost* is defined as:

$$\bar{\alpha}(\boldsymbol{\theta})=\lim_{N\to\infty}E\left\{\sum_{k=0}^{N-1}g(q_k,u_k)\right\}, \tag{1}$$

where (q_k,u_k) is the state-action pair at time k along a path under RSP $\mu_{\boldsymbol{\theta}}$.

The SSP problem is formulated as the problem of finding $\boldsymbol{\theta}^\star$ minimizing (1). Note that, in general, we assume $q^\star$ to be cost-free and absorbing, *i.e.,* $g(q^\star,u)=0$ and $P(q^\star,u,q^\star)=1$ for all $u\in A(q^\star)$. Under these conditions, the expected total cost (1) is finite. Then, an MRP problem as described in Sec. III-A can be immediately converted to an SSP problem.

Definition III.3 (Conversion from MRP to SSP)**.** *Given the product MDP $\mathcal{P}=(S_\mathcal{P},s_{\mathcal{P}0},U_\mathcal{P},A_\mathcal{P},P_\mathcal{P},F_\mathcal{P})$ and a set of states $S^\star_\mathcal{P}\subseteq S_\mathcal{P}$, the problem of maximizing the probability of reaching $S^\star_\mathcal{P}$ can be converted to an SSP problem by defining a new MDP $\widetilde{\mathcal{P}}=(\widetilde{S}_\mathcal{P},\tilde{s}_{\mathcal{P}0},\widetilde{U}_\mathcal{P},\widetilde{A}_\mathcal{P},\widetilde{P}_\mathcal{P},g_\mathcal{P})$, where $\widetilde{S}_\mathcal{P}=(S_\mathcal{P}\setminus S^\star_\mathcal{P})\cup\{s^\star_\mathcal{P}\}$, and $s^\star_\mathcal{P}$ is a "dummy" terminal state; $\tilde{s}_{\mathcal{P}0}=s_{\mathcal{P}0}$ (without the loss of generality, we exclude the trivial case where $s_{\mathcal{P}0}\in S^\star_\mathcal{P}$); $\widetilde{U}_\mathcal{P}=U_\mathcal{P}$; $\widetilde{A}_\mathcal{P}(s_\mathcal{P})=A_\mathcal{P}(s_\mathcal{P})$ for all $s_\mathcal{P}\in S_\mathcal{P}$, and for the dummy state we set $\widetilde{A}_\mathcal{P}(s^\star_\mathcal{P})=\widetilde{U}_\mathcal{P}$; The transition probability is redefined as follows. We first define $\bar{S}^\star_\mathcal{P}$ as the set of states on $\mathcal{P}$ that cannot reach $S^\star_\mathcal{P}$ under any policy. We then define:*

$$\begin{aligned}&\widetilde{P}_\mathcal{P}(s_\mathcal{P},u,s'_\mathcal{P})\\ =\;&\begin{cases}\displaystyle\sum_{s''_\mathcal{P}\in S^\star_\mathcal{P}}P_\mathcal{P}(s_\mathcal{P},u,s''_\mathcal{P}), & \text{if } s'_\mathcal{P}=s^\star_\mathcal{P}\\ P_\mathcal{P}(s_\mathcal{P},u,s'_\mathcal{P}), & \text{if } s'_\mathcal{P}\in S_\mathcal{P}\setminus S^\star_\mathcal{P}\end{cases}\end{aligned}$$

for all $s_\mathcal{P}\in S_\mathcal{P}\setminus(S^\star_\mathcal{P}\cup\bar{S}^\star_\mathcal{P})$ and $u\in\widetilde{A}_\mathcal{P}(s_\mathcal{P})$. Moreover, for all $s_\mathcal{P}\in\bar{S}^\star_\mathcal{P}$ and $u\in\widetilde{A}_\mathcal{P}(s_\mathcal{P})$, we set $\widetilde{P}_\mathcal{P}(s^\star_\mathcal{P},u,s^\star_\mathcal{P})=1$ and $\widetilde{P}_\mathcal{P}(s_\mathcal{P},u,s_{\mathcal{P}0})=1$; For all $s_\mathcal{P}\in\widetilde{S}_\mathcal{P}$ and $u\in\widetilde{A}_\mathcal{P}(s_\mathcal{P})$, we define the one-step cost $g_\mathcal{P}(s_\mathcal{P},u)=1$ if $s_\mathcal{P}\in\bar{S}^\star_\mathcal{P}$, and $g(s_\mathcal{P},u)=0$ otherwise.

We showed in [15] that the policy minimizing (1) for the SSP problem with MDP $\widetilde{\mathcal{P}}$ and the termination state $s^\star_\mathcal{P}$ is the solution to the MRP problem on $\mathcal{P}$ for the set $S^\star_\mathcal{P}$.

The SSP problem can also be constructed from $\mathcal{P}^\mathcal{N}$. In this case we obtain an NTS $\widetilde{\mathcal{P}}^\mathcal{N}(\widetilde{S}_\mathcal{P},\tilde{s}_{\mathcal{P}0},\widetilde{U}_\mathcal{P},\widetilde{A}_\mathcal{P},\widetilde{P}^\mathcal{N}_\mathcal{P},g_\mathcal{P})$, using the same construction as Def. III.3, except $\widetilde{P}^\mathcal{N}_\mathcal{P}$ is defined as: $\widetilde{P}^\mathcal{N}_\mathcal{P}(s_\mathcal{P},u,s'_\mathcal{P})$

$$=\begin{cases}\displaystyle\max_{s''_\mathcal{P}\in S^\star_\mathcal{P}}P^\mathcal{N}_\mathcal{P}(s_\mathcal{P},u,s''_\mathcal{P}), & \text{if } s'_\mathcal{P}=s^\star_\mathcal{P}\\ P^\mathcal{N}_\mathcal{P}(s_\mathcal{P},u,s'_\mathcal{P}), & \text{if } s'_\mathcal{P}\in S_\mathcal{P}\setminus S^\star_\mathcal{P}\end{cases}$$

for all $s_\mathcal{P}\in S_\mathcal{P}\setminus(S^\star_\mathcal{P}\cup\bar{S}^\star_\mathcal{P})$ and $u\in\widetilde{A}_\mathcal{P}(s_\mathcal{P})$. Moreover, for all $s_\mathcal{P}\in\bar{S}^\star_\mathcal{P}$ and $u\in\widetilde{A}_\mathcal{P}(s_\mathcal{P})$, we set $\widetilde{P}^\mathcal{N}_\mathcal{P}(s^\star_\mathcal{P},u,s^\star_\mathcal{P})=1$ and $\widetilde{P}^\mathcal{N}_\mathcal{P}(s_\mathcal{P},u,s_{\mathcal{P}0})=1$.

Once the SSP problem is constructed, the algorithm presented in [15] is an iterative procedure that obtains a policy that locally minimizes the cost function (1) by simulating sample paths on $\widetilde{\mathcal{P}}$. Each sample paths on $\widetilde{\mathcal{P}}$ starts at $s_{\mathcal{P}0}$ and ends when the termination state $s^\star_\mathcal{P}$ is reached. Since the probabilities is needed only along the sample path, we do not require $\widetilde{\mathcal{P}}$, but only $\widetilde{\mathcal{P}}^\mathcal{N}$.

An actor-critic algorithm operates in the following way: the critic observes state and one-step cost from MDP and uses observed information to update the critic parameters, then the critic parameters are used to update the policy; the actor generates the action based on the policy and applies the action to the MDP. The algorithm stops when the gradient of $\bar{\alpha}(\boldsymbol{\theta})$ is small enough (*i.e.,* $\boldsymbol{\theta}$ is locally optimal).

We omit the detail of the actor-critic algorithm, and only note that it does not depend on the form of RSP $\mu_{\boldsymbol{\theta}}$, and it is of the LSTD type, which has shown to be superior to other approximate dynamic programming methods in terms of the convergence rate [14]. The detail of this algorithm can be found in the technical report [16].

C. RSP Design

In this section we describe a suitable randomized policy for MRP problems that can be obtained from $\mathcal{P}^\mathcal{N}$ and do not require the transition probabilities. We propose a family of RSPs that perform a "t steps look-ahead". This class of policies consider all possible sequences of actions in t steps and obtain a probability for each action sequence.

To simplify notation, for a pair of states $i,j\in\widetilde{S}_\mathcal{P}$, we denote $i\xrightarrow{t}j$ if there is a positive probability of reaching j from i in t step. This can be quickly verified given $\widetilde{P}^\mathcal{N}_\mathcal{P}$. At state $i\in\widetilde{S}_\mathcal{P}$, we denote an action sequence from i with t steps look-ahead as $e=u_1u_2\ldots u_t$, where $u_k\in\widetilde{A}_\mathcal{P}(j)$ for some j such that $i\xrightarrow{k}j$, for all $k=1,\ldots t$. We denote the set of all action sequences from state i as $E(i)$. Given $e\in E(i)$, we denote (with a slight abuse of notation) $\widetilde{P}^\mathcal{N}_\mathcal{P}(i,e,j)=1$ if there is a positive probability of reaching j from i with the action sequence e. This can also be recursively obtained given $\widetilde{P}^\mathcal{N}_\mathcal{P}(i,u,\cdot)$.

For each pair of states $i,j\in\widetilde{S}_\mathcal{P}$, we define $d(i,j)$ as the minimum number of steps from i to reach j under any policy (this can be obtained quickly from $\widetilde{P}^\mathcal{N}_\mathcal{P}$ with a simple graph search). We denote $j\in N(i)$ if and only if $d(i,j)\leq r_N$, where r_N is a fixed integer given apriori. If $j\in N(i)$, then we say i is in the neighborhood of j, and r_N represents the radius of the neighborhood around each state.

For each state $i\in\widetilde{S}_\mathcal{P}$, We define the *safety score* $\mathtt{safe}(i)$ as the ratio of the neighboring states not in $\bar{S}^\star_\mathcal{P}$ over all neighboring states of i, where $\bar{S}^\star_\mathcal{P}$ is the set of states with 0 probability of reaching $S^\star_\mathcal{P}$ under any policy. More specifically:

$$\mathtt{safe}(i):=\frac{\sum_{j\in N(i)}I(j)}{|N(i)|}, \tag{2}$$

where $I(i)$ is an indicator function such that $I(i) = 1$ if and only if $i \in \widetilde{S}_{\mathcal{P}} \backslash \bar{S}^{\star}_{\mathcal{P}}$ and $I(i) = 0$ if otherwise. A higher safety score for the current state implies that it is less likely to reach $\bar{S}^{\star}_{\mathcal{P}}$ in the near future. Furthermore, we define the *progress score* of a state $i \in \widetilde{S}_{\mathcal{P}}$ as $\texttt{prog}(i) := \min_{j \in S^{\star}_{\mathcal{P}}} d(i,j)$, which is the minimum number of steps from i to any state in $\mathcal{S}^{\star}_{\mathcal{P}}$.

We can now present the definition of our RSP. Let $\boldsymbol{\theta} := [\theta_1, \theta_2]^{\mathrm{T}}$. We define:

$$a(\boldsymbol{\theta}, i, e) = \texttt{exp}\Big(\theta_1 \sum_{j \in N(i)} \texttt{safe}(j) \widetilde{P}^{\mathcal{N}}_{\mathcal{P}}(i,e,j) + \theta_2 \sum_{j \in N(i)} \left(\texttt{prog}(j) - \texttt{prog}(i)\right) \widetilde{P}^{\mathcal{N}}_{\mathcal{P}}(i,e,j)\Big), \quad (3)$$

where exp is the exponential function. Note that $a(\boldsymbol{\theta}, i, e)$ is the combination of the expected safety score of the next state applying the action sequence e, and the expected improved progress score from the current state applying e, weighted by θ_1 and θ_2. We assign the probability of choosing the action sequence e at i proportional to the combined score $a(\boldsymbol{\theta}, i, e)$. Hence, the probability of choosing action sequence e at state i is:

$$\tilde{\mu}_{\boldsymbol{\theta}}(i,e) = \frac{a(\boldsymbol{\theta}, i, e)}{\sum_{e \in E(i)} a(\boldsymbol{\theta}, i, e)}. \quad (4)$$

Note that, for any chosen action sequence e, only the first action is applied. Hence, at stat i, the probability of choosing action $u \in \widetilde{A}_{\mathcal{P}}(i)$ can be derived from (4):

$$\mu_{\boldsymbol{\theta}}(i,u) = \sum_{\{e \in E(i) \,|\, e = u u_2 \ldots u_t\}} \tilde{\mu}_{\boldsymbol{\theta}}(i,e), \quad (5)$$

which completes the definition of the RSP.

D. Overall Algorithm

We now connect all the pieces together and present the overall algorithm giving a solution to Prob. II.3.

Algorithm 1 Overall algorithm

Input: A labeled NTS $\mathcal{M}^{\mathcal{N}} = (Q, q_0, U, A, P^{\mathcal{N}}, \Pi, h)$ modeling a robot in a partitioned environment, LTL formula ϕ over Π, and a simulator to compute $P(q,u,\cdot)$ given any state-action pair (q,u)

1: Translate the LTL formula ϕ to a DRA $\mathcal{R}_{\phi}$
2: Generate the product NTS $\mathcal{P}^{\mathcal{N}} = \mathcal{M}^{\mathcal{N}} \times \mathcal{R}_{\phi}$
3: Find the union of all AMECs $S^{\star}_{\mathcal{P}}$ associated with $\mathcal{P}^{\mathcal{N}}$
4: Convert from an MRP to an SSP and generate $\widetilde{\mathcal{P}}^{\mathcal{N}}$
5: Obtained the RSP $\mu_{\boldsymbol{\theta}}$ from $\mathcal{P}^{\mathcal{N}}$
6: Execute the actor-critic algorithm with $\widetilde{\mathcal{P}}^{\mathcal{N}}$ until gradient $||\nabla \bar{\alpha}(\boldsymbol{\theta}^{\star})|| \leq \epsilon$ for a $\boldsymbol{\theta}^{\star}$ and a given ϵ

Output: RSP μ_{θ} and $\theta^{\star}$ locally maximizing the probability of satisfying ϕ with respect to $\boldsymbol{\theta}$ up to a threshold ϵ

Proposition III.4. *Alg. 1 returns in finite time with $\boldsymbol{\theta}^{\star}$ locally maximizing the probability of the RSP $\mu_{\boldsymbol{\theta}}$ satisfying the LTL formula ϕ.*

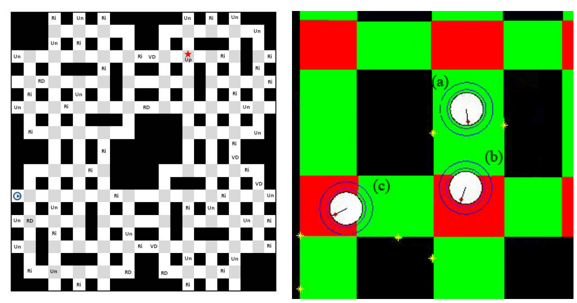
Proof. See [16]. ∎

IV. Hardware-in-the-loop simulation

In this section, we test the algorithms proposed in this paper through hardware-in-the-loop simulation for the RIDE environment (see www.hyness.bu.edu/ride for more information). The transition probabilities are computed by an accurate simulator of RIDE as needed. We then compare the exact solution with the approximate solution obtained by the proposed approach.

A. Environment

We consider an environment whose topology is shown in Fig. 2a. This environment is made of square blocks forming 164 corridors and 84 intersections. The corridors $(C_1, C_2, \ldots, C_{164})$ shown as white regions in Fig. 2a are of three different lengths, one-, two-, and three-unit lengths. The three-unit corridors are used to build corners in the environment. The intersections $(I_1, I_2, \ldots, I_{84})$ are of two types, three-way and four-way, and are shown as grey blocks in Fig. 2a. The black regions in this figure represent the walls of the environment. Note that there is always a corridor between two intersections.



(a) (b)

Fig. 2. Fig. 3a:Schematic representation of the environment with 84 intersections and 164 corridors. The black blocks represent walls, and the grey and white regions are intersection and corridors, respectively. There are five properties of interest in the regions $\{\mathbf{VD}, \mathbf{RD}, \mathbf{Up}, \mathbf{Ri}, \mathbf{Un}\}$. Fig. 3b: Simulation snapshots. The white disk represents the robot and the different circles around it indicate different regions. (a) The robot centers itself on a stretch of corridor by using FollowRoad; (b) The robot applies GoRight in an intersection; (c) The robot applies GoLeft.

There are five properties of interest (observations) associated with the regions of the environment. These properties are: **VD** = *ValuableData* (regions containing valuable data to be collected), **RD** = *RegularData* (regions containing regular data to be collected), **Up** = *Upload* (regions where data can be uploaded), **Ri** = *Risky* (regions that could pose a threat to the robot), and **Un** = *Unsafe* (unsafe regions).

B. Construction of the MDP model

The robot is equipped with a set of feedback control primitives (actions) - FollowRoad, GoRight, GoLeft, and GoStraight. The controller FollowRoad is only available (enabled) at the corridors. At four-way intersections, controllers are GoRight, GoLeft, and GoStraight. At three-way intersections, depending on the shape of the intersection, two of the four controllers are available. The resulting motion may be different than intended due to the presence of noise in the actuators and sensors, leading to probabilistic transitions.

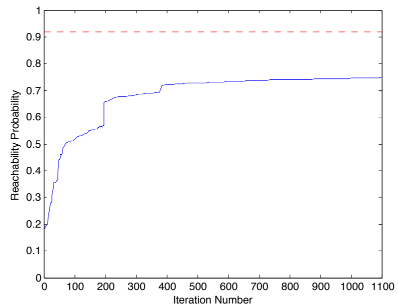


Fig. 3. The optimal solution (the maximal probability of satisfying ϕ) is shown with the dashed line, and the solid line represents the satisfying probability for the RSP as a function of $\boldsymbol{\theta}_k$ at each iteration k.

To create an MDP model of the robot in RIDE, we define each state of the MDP as a collection of two adjacent regions (a corridor and an intersection). For instance the pairs C_1-I_2 and I_3-C_4 are two states of the MDP. Through this pairing of regions, it was shown that the Markov property (*i.e.,* the result of an action at a state depends only on the current state) can be achieved [11]. The resulting MDP has 608 states. The set of actions available at a state is the set of controllers available at the last region corresponding to the state. For example, when in state C_1-I_2 only those actions from region I_2 are allowed. Each state of the MDP whose second region satisfies an observation in Π is mapped to that observation.

To obtain transition probabilities, we use an accurate simulator (see Fig. 2b) incorporating the motion and sensing of an iRobot Create platform with a Hokoyu URG-04LX laser range finder, APSX RW-210 RFID reader, and an MSI Wind U100-420US netbook. Specifically, it emulates experimentally measured response times, sensing and control errors, noise levels and distributions in the laser scanner readings. We perform a total of 1000 simulations to obtain transition probabilities for each action-state pair of the MDP.

C. Task specification and results

We consider the following mission task: *Reach a location with* ValuableData ***(VD)*** *or* RegularData ***(RD)****, and then reach* Upload ***(Up)****. Do not reach* Risky ***(Ri)*** *regions unless eventually reach a location with* ValuableData ***(VD)****. Always avoid* Unsafe ***(Un)*** *regions until* Upload ***(Up)*** *is reached (and mission completed)*. This task specification can be translated to the LTL formula:

$$\begin{aligned} \phi \quad := \quad & \mathsf{F}\,\mathbf{Up} \wedge (\neg \mathbf{Un}\,\mathsf{U}\,\mathbf{Up}) \wedge \mathsf{G}\,(\mathbf{Ri} \longrightarrow \mathsf{F}\,\mathbf{VD}) \\ & \wedge \mathsf{G}\,(\mathbf{VD} \vee \mathbf{RD} \longrightarrow \mathsf{X}\,\mathsf{F}\,\mathbf{Up}) \end{aligned} \tag{6}$$

The initial position of the robot is shown as a blue circle in Fig. 2a. The size of the DRA is 17 which results in the product MDP with 10336 states. By applying both methods of linear programming (exact solution) and Alg. 1 (approximate solution), we found the maximal probability of satisfying the specification were 92% and 75%, respectively. The graph of the convergence of the actor-critic solution is shown in Fig. 3. The parameters for this examples are: $\lambda = 0.9$, and the initial $\boldsymbol{\theta}_0 = [5, -0.5]^{\mathsf{T}}$. The look-ahead window t for the RSP is 2. Since the transition probabilities are computed only along the sample path. When Alg. 1 is completed (at iteration 1100), at most 1100 transition probabilities were computed. In comparison, in order to solve the probability exactly, around 30000 transition probabilities of state-action pairs must be computed.

V. Conclusions

We presented a framework that brings together an approximate dynamic programming computational method of the actor critic type, with formal control synthesis for robots modeled as Markov Decision Processes (MDPs) from temporal logic specifications. We show that this approach is particularly suitable for problems where the transition probabilities are difficult or computationally expensive to compute. We show that this approach effectively finds an approximate optimal policy within a class of randomized stationary polices. Future directions include extending this result to multi-robot teams, providing an insight on how to choose an appropriate look-ahead window when designing the RSP, and applying the result to more realistic test-cases.

References

[1] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, “Where’s Waldo? Sensor-based temporal logic motion planning,” in *Proc ICRA*, Rome, Italy, 2007, pp. 3116–3121.

[2] S. Karaman and E. Frazzoli, “Sampling-based motion planning with deterministic μ-calculus specifications,” in *Proc CDC*, Shanghai, China, 2009, pp. 2222 – 2229.

[3] T. Wongpiromsarn, U. Topcu, and R. M. Murray, “Receding horizon temporal logic planning for dynamical systems,” in *Proc CDC*, Shanghai, China, 2009.

[4] A. Bhatia, L. Kavraki, and M. Vardi, “Sampling-based motion planning with temporal goals,” in *Proc ICRA*. IEEE, 2010, pp. 2689–2696.

[5] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[6] L. De Alfaro, “Formal verification of probabilistic systems,” Ph.D. dissertation, Stanford University, 1997.

[7] M. Vardi, “Probabilistic linear-time model checking: An overview of the automata-theoretic approach,” *Formal Methods for Real-Time and Probabilistic Systems*, pp. 265–276, 1999.

[8] C. Courcoubetis and M. Yannakakis, “Markov decision processes and regular events,” *IEEE TAC*, vol. 43, no. 10, pp. 1399–1418, 1998.

[9] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, “Controller synthesis for probabilistic systems,” in *Proc IFIP TCS*, 2004.

[10] X. C. Ding, S. L. Smith, C. Belta, and D. Rus, “LTL planning for groups of robots control in uncertain environments with probabilistic satisfaction guarantees,” in *18th IFAC World Congress*, 2011.

[11] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, “Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees,” in *Proc ICRA*, Anchorage, AK, 2010, pp. 3227 – 3232.

[12] A. Barto, R. Sutton, and C. Anderson, “Neuronlike adaptive elements that can solve difficult learning control problems.” *IEEE Transactions on Systems, Man, & Cybernetics*, 1983.

[13] I. Paschalidis, K. Li, and R. Estanjini, “An actor-critic method using least squares temporal difference learning,” in *Proc CDC*, pp. 2564–2569.

[14] V. Konda and J. Tsitsiklis, “On actor-critic algorithms,” *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2004.

[15] R. Estanjini, X. Ding, M. Lahijanian, J. Wang, C. Belta, and I. Paschalidis, “Least squares temporal difference actor-critic methods with applications to robot motion control,” in *Proc CDC*, December 2011.

[16] X. C. Ding, J. Wang, M. Lahijanian, I. C. Paschalidis, and C. Belta, “Temporal logic motion control using actor-critic methods,” Boston University, CISE Technical Report, Tracking Number 2012-IR-0003, (available at http://www.bu.edu/systems/publications/).

[17] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.