2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Least Squares Temporal Difference Actor-Critic Methods with Applications to Robot Motion Control *

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

***Abstract*— We consider the problem of finding a control policy for a Markov Decision Process (MDP) to maximize the probability of reaching some states while avoiding some other states. This problem is motivated by applications in robotics, where such problems naturally arise when probabilistic models of robot motion are required to satisfy temporal logic task specifications. We transform this problem into a Stochastic Shortest Path (SSP) problem and develop a new approximate dynamic programming algorithm to solve it. This algorithm is of the actor-critic type and uses a least-square temporal difference learning method. It operates on sample paths of the system and optimizes the policy within a pre-specified class parameterized by a parsimonious set of parameters. We show its convergence to a policy corresponding to a stationary point in the parameters' space. Simulation results confirm the effectiveness of the proposed solution.**

***Index Terms*— Markov Decision Processes, dynamic programming, actor-critic methods, robot motion control, robotics.**

I. Introduction

Markov Decision Processes (MDPs) have been widely used in a variety of application domains. In particular, they have been increasingly used to model and control autonomous agents subject to noises in their sensing and actuation, or uncertainty in the environment they operate. Examples include: unmanned aircraft [1], ground robots [2], and steering of medical needles [3]. In these studies, the underlying motion of the system cannot be predicted with certainty, but they can be obtained from the sensing and the actuation model through a simulator or empirical trials, providing transition probabilities.

Recently, the problem of controlling an MDP from a temporal logic specification has received a lot of attention. Temporal logics such as Linear Temporal Logic (LTL) and Computational Tree Logic (CTL) are appealing as they provide formal, high level languages in which the behavior of the system can be specified (see [4]). In the context of MDPs, providing probabilistic guarantees means finding optimal policies that maximize the probabilities of satisfying these specifications. In [2], [5], it has been shown that, the problem of finding an optimal policy that maximizes the probability of satisfying a temporal logic formula can be naturally translated to one of maximizing the probability of reaching a set of states in the MDP. Such problems are referred to as Maximal Reachability Probability (MRP) problems. It has been known [3] that they are equivalent to Stochastic Shortest Path (SSP) problems, which belong to a standard class of infinite horizon problems in dynamic programming.

However, as suggested in [2], [5], these problems usually involve MDPs with large state spaces. For example, in order to synthesize an optimal policy for an MDP satisfying an LTL formula, one needs to solve an MRP problem on a much larger MDP, which is the product of the original MDP and an automaton representing the formula. Thus, computing the exact solution can be computationally prohibitive for realistically-sized settings. Moreover, in some cases, the system of interest is so complex that it is not feasible to determine transition probabilities for all actions and states explicitly.

Motivated by these limitations, in this paper we develop a new approximate dynamic programming algorithm to solve SSP MDPs and we establish its convergence. The algorithm is of the actor-critic type and uses a *Least Square Temporal Difference* (LSTD) learning method. Our proposed algorithm is based on sample paths, and thus only requires transition probabilities along the sampled paths and not over the entire state space.

Actor-critic algorithms are typically used to optimize some *Randomized Stationary Policy* (RSP) using policy gradient estimation. RSPs are parameterized by a parsimonious set of parameters and the objective is to optimize the policy with respect to these parameters. To this end, one needs to estimate appropriate policy gradients, which can be done using learning methods that are much more efficient than computing a cost-to-go function over the entire state-action space. Many different versions of actor-critic algorithms have been proposed and have been shown to be effective for various applications (e.g., in robotics [6] and navigation [7], power management of wireless transmitters [8], biology [9], and optimal bidding for electricity generation [10]).

A particularly attractive design of the actor-critic architecture was proposed in [11], where the *critic* estimates the policy gradient using sequential observations from a

* Research partially supported by the NSF under grant EFRI-0735974, by the DOE under grant DE-FG52-06NA27490, by the ARO under grant W911NF-11-1-0227, by the ODDR&E MURI10 program under grant N00014-10-1-0952, and by ONR MURI under grant N00014-09-1051.

† Reza Moazzez Estanjini and Jing Wang are with the Division of Systems Eng., Boston University, 8 St. Mary's St., Boston, MA 02215, email: {reza,wangjing}@bu.edu.

‡ Xu Chu Ding, Morteza Lahijanian, and Calin A. Belta are with the Dept. of Mechanical Eng., Boston University, 15 St. Mary's St., Boston, MA 02215, email: {xcding,morteza,cbelta}@bu.edu.

§ Ioannis Ch. Paschalidis is with the Dept. of Electrical & Computer Eng., and the Division of Systems Eng., Boston University, 8 St. Mary's St., Boston, MA 02215, email: yannisp@bu.edu.

§ Corresponding author.

sample path while the *actor* updates the policy at the same time, although at a slower time-scale. It was proved that the estimate of the critic tracks the slowly varying policy asymptotically under suitable conditions. A center piece of these conditions is a relationship between the actor step-size and the critic step-size, which will be discussed later.

The critic of [11] uses first-order variants of the *Temporal Difference* (TD) algorithm (TD(1) and TD(λ)). However, it has been shown that the least squares methods – LSTD (Least Squares TD) and LSPE (Least Squares Policy Evaluation) – are superior in terms of convergence rate (see [12], [13]). LSTD and LSPE were first proposed for discounted cost problems in [12] and [14], respectively. Later, [13] showed that the convergence rate of LSTD is optimal. Their results clearly demonstrated that LSTD converges much faster and more reliably than TD(1) and TD(λ).

Motivated by these findings, in this paper we propose an actor-critic algorithm that adopts LSTD learning methods tailored to SSP problems, while at the same time maintains the concurrent update architecture of the actor and the critic. (Note that [15] also used LSTD in an actor-critic method, but the actor had to wait for the critic to converge before making each policy update.) To illustrate salient features of the approach, we present a case study where a robot in a large environment is required to satisfy a task specification of "go to a set of goal states while avoiding a set of unsafe states." We note that more complex task specifications can be directly converted to MRP problems as shown in [2], [5].

Notation: We use bold letters to denote vectors and matrices; typically vectors are lower case and matrices upper case. Vectors are assumed to be column vectors unless explicitly stated otherwise. Transpose is denoted by prime. For any $m \times n$ matrix $\mathbf{A}$, with rows $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{R}^n$, $\mathbf{v}(\mathbf{A})$ denotes the column vector $(\mathbf{a}_1, \ldots, \mathbf{a}_m)$. $\|\cdot\|$ stands for the Euclidean norm and $\|\cdot\|_{\boldsymbol{\theta}}$ is a special norm in the MDP state-action space that we will define later. $\mathbf{0}$ denotes a vector or matrix with all components set to zero and $\mathbf{I}$ is the identity matrix. $|S|$ denotes the cardinality of a set S.

II. Problem Formulation

Consider an SSP MDP with finite state and action spaces. Let k denote time, $\mathbb{X}$ denote the state space with cardinality $|\mathbb{X}|$, and $\mathbb{U}$ denote the action space with cardinality $|\mathbb{U}|$. Let $\mathbf{x}_k \in \mathbb{X}$ and $u_k \in \mathbb{U}$ be the state of the system and the action taken at time k, respectively. Let $g(\mathbf{x}_k, u_k)$ be the one-step cost of applying action u_k while the system is at state $\mathbf{x}_k$. Let $\mathbf{x}_0$ and $\mathbf{x}^*$ denote the initial state and the special cost-free termination state, respectively. Let $p(\mathbf{j}|\mathbf{x}_k, u_k)$ denote the state transition probabilities (which are typically not explicitly known); that is, $p(\mathbf{j}|\mathbf{x}_k, u_k)$ is the probability of transition from state $\mathbf{x}_k$ to state $\mathbf{j}$ given that action u_k is taken while the system is at state $\mathbf{x}_k$. A policy $\boldsymbol{\mu}$ is said to be *proper* if, when using this policy, there is a positive probability that $\mathbf{x}^*$ will be reached after at most $|\mathbb{X}|$ transitions, regardless of the initial state $\mathbf{x}_0$. We make the following assumption.

Assumption A
There exist a proper stationary policy.

The policy candidates are assumed to belong to a parameterized family of *Randomized Stationary Policies* (RSPs) $\{\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \mid \boldsymbol{\theta} \in \mathbb{R}^n\}$. That is, given a state $\mathbf{x} \in \mathbb{X}$ and a parameter $\boldsymbol{\theta}$, the policy applies action $u \in \mathbb{U}$ with probability $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$. Define the *expected total cost* $\bar{\alpha}(\boldsymbol{\theta})$ to be $\lim_{t\to\infty} E\{\sum_{k=0}^{t-1} g(\mathbf{x}_k, u_k)|\mathbf{x}_0\}$ where u_k is generated according to RSP $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$. The goal is to optimize the expected total cost $\bar{\alpha}(\boldsymbol{\theta})$ over the n-dimensional vector $\boldsymbol{\theta}$.

With no explicit model of the state transitions but only a sample path denoted by $\{\mathbf{x}_k, u_k\}$, the actor-critic algorithms typically optimize $\boldsymbol{\theta}$ locally in the following way: first, the critic estimates the policy gradient $\nabla\bar{\alpha}(\boldsymbol{\theta})$ using a *Temporal Difference (TD)* algorithm; then the actor modifies the policy parameter along the gradient direction. Let the operator $P_{\boldsymbol{\theta}}$ denote taking expectation after one transition. More precisely, for a function $f(\mathbf{x}, u)$, $(P_{\boldsymbol{\theta}} f)(\mathbf{x}, u) = \sum_{\mathbf{j}\in\mathbb{X}, \nu\in\mathbb{U}} \mu_{\boldsymbol{\theta}}(\nu|\mathbf{j}) p(\mathbf{j}|\mathbf{x}, u) f(\mathbf{j}, \nu)$. Define the $Q_{\boldsymbol{\theta}}$-value function to be any function satisfying the Poisson equation

$$Q_{\boldsymbol{\theta}}(\mathbf{x}, u) = g(\mathbf{x}, u) + (P_{\boldsymbol{\theta}} Q_{\boldsymbol{\theta}})(\mathbf{x}, u),$$

where $Q_{\boldsymbol{\theta}}(\mathbf{x}, u)$ can be interpreted as the expected future cost we incur if we start at state $\mathbf{x}$, apply control u, and then apply RSP $\mu_{\boldsymbol{\theta}}$. We note that in general, the Poisson equation need not hold for SSP, however, it holds if the policy corresponding to RSP $\mu_{\boldsymbol{\theta}}$ is a proper policy [16]. We make the following assumption.

Assumption B
For any $\boldsymbol{\theta}$, and for any $\mathbf{x} \in \mathbb{X}$, $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) > 0$ if action u is feasible at state $\mathbf{x}$, and $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0$ otherwise.

We note that one possible RSP for which Assumption B holds is the "Boltzmann" policy (see [17]), that is

$$\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) = \frac{\exp(h_{\boldsymbol{\theta}}^{(u)}(\mathbf{x}))}{\sum_{a\in\mathbb{U}} \exp(h_{\boldsymbol{\theta}}^{(a)}(\mathbf{x}))}, \tag{1}$$

where $h_{\boldsymbol{\theta}}^{(u)}(\mathbf{x})$ is a function that corresponds to action u and is parameterized by $\boldsymbol{\theta}$. The Boltzmann policy is simple to use and is the policy that will be used in the case study in Sec. V.

Lemma II.1 *If Assumptions A and B hold, then for any $\boldsymbol{\theta}$ the policy corresponding to RSP $\mu_{\boldsymbol{\theta}}$ is proper.*

Proof: The proof follows from the definition of a proper policy. ∎

Under suitable ergodicity conditions, $\{\mathbf{x}_k\}$ and $\{\mathbf{x}_k, u_k\}$ are Markov chains with stationary distributions under a fixed policy. These stationary distributions are denoted by $\pi_{\boldsymbol{\theta}}(\mathbf{x})$ and $\eta_{\boldsymbol{\theta}}(\mathbf{x}, u)$, respectively. We will not elaborate on the ergodicity conditions, except to note that it suffices that the process $\{\mathbf{x}_k\}$ is irreducible and aperiodic given any

$\boldsymbol{\theta}$, and Assumption B holds. Denote by $\mathbf{Q}_{\boldsymbol{\theta}}$ the $(|\mathbb{X}||\mathbb{U}|)$-dimensional vector $\mathbf{Q}_{\boldsymbol{\theta}} = (Q_{\boldsymbol{\theta}}(\mathbf{x}, u);\ \forall \mathbf{x} \in \mathbb{X},\ u \in \mathbb{U})$. Let now

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \nabla_{\boldsymbol{\theta}} \ln \mu_{\boldsymbol{\theta}}(u|\mathbf{x}),$$

where $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \mathbf{0}$ when $\mathbf{x}, u$ are such that $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0$ for all $\boldsymbol{\theta}$. It is assumed that $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u)$ is bounded and continuously differentiable. We write $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = (\psi^1_{\boldsymbol{\theta}}(\mathbf{x}, u), \ldots, \psi^n_{\boldsymbol{\theta}}(\mathbf{x}, u))$ where n is the dimensionality of $\boldsymbol{\theta}$. As we did in defining $\mathbf{Q}_{\boldsymbol{\theta}}$ we will denote by $\boldsymbol{\psi}^i_{\boldsymbol{\theta}}$ the $(|\mathbb{X}||\mathbb{U}|)$-dimensional vector $\boldsymbol{\psi}^i_{\boldsymbol{\theta}} = (\psi^i_{\boldsymbol{\theta}}(\mathbf{x}, u);\ \forall \mathbf{x} \in \mathbb{X},\ u \in \mathbb{U})$.

A key fact underlying the actor-critic algorithm is that the policy gradient can be expressed as (Theorem 2.15 in [13])

$$\frac{\partial \bar{\alpha}(\boldsymbol{\theta})}{\partial \theta_i} = \langle \mathbf{Q}_{\boldsymbol{\theta}}, \boldsymbol{\psi}^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}},\ i = 1, \ldots, n,$$

where for any two functions f_1 and f_2 of $\mathbf{x}$ and u, expressed as $(|\mathbb{X}||\mathbb{U}|)$-dimensional vectors $\mathbf{f}_1$ and $\mathbf{f}_2$, we define

$$\langle \mathbf{f}_1, \mathbf{f}_2 \rangle_{\boldsymbol{\theta}} \stackrel{\triangle}{=} \sum_{\mathbf{x}, u} \eta_{\boldsymbol{\theta}}(\mathbf{x}, u) f_1(\mathbf{x}, u) f_2(\mathbf{x}, u). \tag{2}$$

Let $\|\cdot\|_{\boldsymbol{\theta}}$ denote the norm induced by the inner product (2), *i.e.*, $\|\mathbf{f}\|^2_{\boldsymbol{\theta}} = \langle \mathbf{f}, \mathbf{f} \rangle_{\boldsymbol{\theta}}$. Let also $\mathscr{S}_{\boldsymbol{\theta}}$ be the subspace of $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$ spanned by the vectors $\boldsymbol{\psi}^i_{\boldsymbol{\theta}}$, $i = 1, \ldots, n$ and denote by $\Pi_{\boldsymbol{\theta}}$ the projection with respect to the norm $\|\cdot\|_{\boldsymbol{\theta}}$ onto $\mathscr{S}_{\boldsymbol{\theta}}$, *i.e.*, for any $\mathbf{f} \in \mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$, $\Pi_{\boldsymbol{\theta}}\mathbf{f}$ is the unique vector in $\mathscr{S}_{\boldsymbol{\theta}}$ that minimizes $\|\mathbf{f} - \hat{\mathbf{f}}\|_{\boldsymbol{\theta}}$ over all $\hat{\mathbf{f}} \in \mathscr{S}_{\boldsymbol{\theta}}$. Since for all i

$$\langle \mathbf{Q}_{\boldsymbol{\theta}}, \boldsymbol{\psi}^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}} = \langle \Pi_{\boldsymbol{\theta}} \mathbf{Q}_{\boldsymbol{\theta}}, \boldsymbol{\psi}^i_{\boldsymbol{\theta}} \rangle_{\boldsymbol{\theta}},$$

it is sufficient to know the projection of $\mathbf{Q}_{\boldsymbol{\theta}}$ onto $\mathscr{S}_{\boldsymbol{\theta}}$ in order to compute $\nabla \bar{\alpha}(\boldsymbol{\theta})$. One possibility is to approximate $\mathbf{Q}_{\boldsymbol{\theta}}$ with a parametric linear architecture of the following form (see [11]):

$$Q^{\mathbf{r}}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \boldsymbol{\psi}'_{\boldsymbol{\theta}}(\mathbf{x}, u)\mathbf{r}^*, \qquad \mathbf{r}^* \in \mathbb{R}^n. \tag{3}$$

This dramatically reduces the complexity of learning from the space $\mathbb{R}^{|\mathbb{X}||\mathbb{U}|}$ to the space $\mathbb{R}^n$. Furthermore, the temporal difference algorithms can be used to learn such an $\mathbf{r}^*$ effectively. The elements of $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u)$ are understood as features associated with an $(\mathbf{x}, u)$ state-action pair in the sense of basis functions used to develop an approximation of the $Q_{\boldsymbol{\theta}}$-value function.

III. Actor-Critic Algorithm Using LSTD

The critic in [11] used either TD(λ) or TD(1). The algorithm we propose uses least squares TD methods (LSTD in particular) instead as they have been shown to provide far superior performance. In the sequel, we first describe the LSTD actor-critic algorithm and then we prove its convergence.

A. The Algorithm

The algorithm uses a sequence of simulated trajectories, each of which starting at a given $\mathbf{x}_0$ and ending as soon as $\mathbf{x}^*$ is visited for the first time in the sequence. Once a trajectory is completed, the state of the system is reset to the initial state $\mathbf{x}_0$ and the process is repeated.

Let $\mathbf{x}_k$ denote the state of the system at time k. Let $\mathbf{r}_k$, the iterate for $\mathbf{r}^*$ in (3), be the parameter vector of the critic at time k, $\boldsymbol{\theta}_k$ be the parameter vector of the actor at time k, and $\mathbf{x}_{k+1}$ be the new state, obtained after action u_k is applied when the state is $\mathbf{x}_k$. A new action u_{k+1} is generated according to the RSP corresponding to the actor parameter $\boldsymbol{\theta}_k$ (see [11]). The critic and the actor carry out the following updates, where $\mathbf{z}_k \in \mathbb{R}^n$ represents Sutton's eligibility trace [17], $\mathbf{b}_k \in \mathbb{R}^n$ refers to a statistical estimate of the single period reward, and $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ is a sample estimate of the matrix formed by $\mathbf{z}_k(\boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}) - \boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_k, u_k))$, which can be viewed as a sample observation of the scaled difference of the observation of the state incidence vector for iterations k and $k+1$, scaled to the feature space by the basis functions.

LSTD Actor-Critic for SSP

Initialization:

Set all entries in $\mathbf{z}_0, \mathbf{A}_0, \mathbf{b}_0$ and $\mathbf{r}_0$ to zeros. Let $\boldsymbol{\theta}_0$ take some initial value, potentially corresponding to a heuristic policy.

Critic:

$$\begin{aligned} \mathbf{z}_{k+1} &= \lambda \mathbf{z}_k + \boldsymbol{\psi}_{\boldsymbol{\theta}_k}(\mathbf{x}_k, u_k), \\ \mathbf{b}_{k+1} &= \mathbf{b}_k + \gamma_k \left[g(\mathbf{x}_k, u_k)\mathbf{z}_k - \mathbf{b}_k \right], \\ \mathbf{A}_{k+1} &= \mathbf{A}_k + \gamma_k [\mathbf{z}_k(\boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}) - \boldsymbol{\psi}'_{\theta_k}(\mathbf{x}_k, u_k)) \\ &\qquad - \mathbf{A}_k], \end{aligned} \tag{4}$$

where $\lambda \in [0, 1)$, $\gamma_k \stackrel{\triangle}{=} \dfrac{1}{k}$, and finally

$$\mathbf{r}_{k+1} = -\mathbf{A}_k^{-1}\mathbf{b}_k. \tag{5}$$

Actor:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta_k \Gamma(\mathbf{r}_k)\mathbf{r}'_k \boldsymbol{\psi}_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1})\boldsymbol{\psi}_{\theta_k}(\mathbf{x}_{k+1}, u_{k+1}). \tag{6}$$

In the above, $\{\gamma_k\}$ controls the critic step-size, while $\{\beta_k\}$ and $\Gamma(\mathbf{r})$ control the actor step-size together. An implementation of this algorithm needs to make these choices. The role of $\Gamma(\mathbf{r})$ is mainly to keep the actor updates bounded, and we can for instance use

$$\Gamma(\mathbf{r}) = \begin{cases} \dfrac{D}{||\mathbf{r}||}, & \text{if } ||\mathbf{r}|| > D, \\ 1, & \text{otherwise,} \end{cases}$$

for some $D > 0$. $\{\beta_k\}$ is a deterministic and non-increasing sequence for which we need to have

$$\sum_k \beta_k = \infty, \quad \sum_k \beta_k^2 < \infty, \quad \lim_{k \to \infty} \frac{\beta_k}{\gamma_k} = 0. \tag{7}$$

An example of $\{\beta_k\}$ satisfying Eq. (7) is

$$\beta_k = \frac{c}{k \ln k}, \quad k > 1, \tag{8}$$

where $c > 0$ is a constant parameter. Also, $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u)$ is defined as

$$\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \nabla_{\boldsymbol{\theta}} \ln \mu_{\boldsymbol{\theta}}(u|\mathbf{x}),$$

where $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = \mathbf{0}$ when $\mathbf{x}, u$ are such that $\mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \equiv 0$ for all $\boldsymbol{\theta}$. It is assumed that $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u)$ is bounded and continuously differentiable. Note that $\boldsymbol{\psi}_{\boldsymbol{\theta}}(\mathbf{x}, u) = (\psi^1_{\boldsymbol{\theta}}(\mathbf{x}, u), \ldots, \psi^n_{\boldsymbol{\theta}}(\mathbf{x}, u))$ where n is the dimensionality of $\boldsymbol{\theta}$. The convergence of the algorithm is stated in the following Theorem (see the technical report [18] for the proof).

Theorem III.1 *[Actor Convergence] For the LSTD actor-critic with some step-size sequence $\{\beta_k\}$ satisfying (7), for any $\epsilon > 0$, there exists some λ sufficiently close to 1, such that $\liminf_k ||\nabla \bar{\alpha}(\boldsymbol{\theta}_k)|| < \epsilon$ w.p.1. That is, $\boldsymbol{\theta}_k$ visits an arbitrary neighborhood of a stationary point infinitely often.*

IV. The MRP and its conversion into an SSP problem

In the MRP problem, we assume that there is a set of *unsafe* states which are set to be absorbing on the MDP (*i.e.*, there is only one control at each state, corresponding to a self-transition with probability 1). Let $\mathbb{X}_G$ and $\mathbb{X}_U$ denote the set of goal states and unsafe states, respectively. A *safe* state is a state that is not unsafe. It is assumed that if the system is at a safe state, then there is at least one sequence of actions that can reach one of the states in $\mathbb{X}_G$ with positive probability. Note that this implies that Assumption A holds. In the MRP, the goal is to find the optimal policy that maximizes the probability of reaching a state in $\mathbb{X}_G$ from a given initial state. Note that since the unsafe states are absorbing, to satisfy this specification the system must not visit the unsafe states.

We now convert the MRP problem into an SSP problem, which requires us to change the original MDP (now denoted as $\text{MDP}_{\mathbb{M}}$) into a SSP MDP (denoted as $\text{MDP}_{\mathbb{S}}$). Note that [3] established the equivalence between an MRP problem and an SSP problem where the expected reward is maximized. Here we present a different transformation where an MRP problem is converted to a more standard SSP problem where the expected cost is minimized.

To begin, we denote the state space of $\text{MDP}_{\mathbb{M}}$ by $\mathbb{X}_{\mathbb{M}}$, and define $\mathbb{X}_{\mathbb{S}}$, the state space of $\text{MDP}_{\mathbb{S}}$, to be

$$\mathbb{X}_{\mathbb{S}} = (\mathbb{X}_{\mathbb{M}} \setminus \mathbb{X}_G) \cup \{\mathbf{x}^*\},$$

where $\mathbf{x}^*$ denotes a special termination state. Let $\mathbf{x}_0$ denote the initial state, and $\mathbb{U}$ denote the action space of $\text{MDP}_{\mathbb{M}}$. We define the action space of $\text{MDP}_{\mathbb{S}}$ to be $\mathbb{U}$, *i.e.*, the same as for $\text{MDP}_{\mathbb{M}}$.

Let $p_{\mathbb{M}}(\mathbf{j}|\mathbf{x}, u)$ denote the probability of transition to state $\mathbf{j} \in \mathbb{X}_{\mathbb{M}}$ if action u is taken at state $\mathbf{x} \in \mathbb{X}_{\mathbb{M}}$. We now define the transition probability $p_{\mathbb{S}}(\mathbf{j}|\mathbf{x}, u)$ for all states $\mathbf{x}, \mathbf{j} \in \mathbb{X}_{\mathbb{S}}$ as:

$$p_{\mathbb{S}}(\mathbf{j}|\mathbf{x}, u) = \begin{cases} \sum_{\mathbf{i} \in \mathbb{X}_G} p_{\mathbb{M}}(\mathbf{i}|\mathbf{x}, u), & \text{if } \mathbf{j} = \mathbf{x}^*, \\ p_{\mathbb{M}}(\mathbf{j}|\mathbf{x}, u), & \text{if } \mathbf{j} \in \mathbb{X}_{\mathbb{M}} \setminus \mathbb{X}_G, \end{cases} \tag{9}$$

for all $\mathbf{x} \in \mathbb{X}_{\mathbb{M}} \setminus (\mathbb{X}_G \cup \mathbb{X}_U)$ and all $u \in \mathbb{U}$. Furthermore, we set $p_{\mathbb{S}}(\mathbf{x}^*|\mathbf{x}^*, u) = 1$ and $p_{\mathbb{S}}(\mathbf{x}_0|\mathbf{x}, u) = 1$ if $\mathbf{x} \in \mathbb{X}_U$, for all $u \in \mathbb{U}$. The transition probability of $\text{MDP}_{\mathbb{S}}$ is defined to be the same as for $\text{MDP}_{\mathbb{M}}$, except that the probability of visiting the goal states in $\text{MDP}_{\mathbb{M}}$ is changed into the probability of visiting the termination state; and the unsafe states transit to the initial state with probability 1.

For all $\mathbf{x} \in \mathbb{X}_S$, we define the cost $g(\mathbf{x}, u) = 1$ if $\mathbf{x} \in \mathbb{X}_U$, and $g(\mathbf{x}, u) = 0$ otherwise. Define the expected total cost of a policy μ to be $\bar{\alpha}^{\mathbb{S}}_{\mu} = \lim_{t \to \infty} E\{\sum_{k=0}^{t-1} g(\mathbf{x}_k, u_k)|\mathbf{x}_0\}$ where actions u_k are obtained according to policy μ in $\text{MDP}_{\mathbb{S}}$. Moreover, for each policy μ on $\text{MDP}_{\mathbb{S}}$, we can define a policy on $\text{MDP}_{\mathbb{M}}$ to be the same as μ for all states $\mathbf{x} \in \mathbb{X}_{\mathbb{M}} \setminus (\mathbb{X}_G \cup \mathbb{X}_U)$. Since actions are irrelevant at the goal and unsafe states in both MDPs, with slight abuse of notation we denote both policies to be μ. Finally, we define the *Reachability Probability* $R^{\mathbb{M}}_{\mu}$ as the probability of reaching one of the goal states from $\mathbf{x}_0$ under policy μ on $\text{MDP}_{\mathbb{M}}$. The Lemma below relates $R^{\mathbb{M}}_{\mu}$ and $\bar{\alpha}^{\mathbb{S}}_{\mu}$:

Lemma IV.1 *For any RSP μ, we have $R^{\mathbb{M}}_{\mu} = \frac{1}{\bar{\alpha}^{\mathbb{S}}_{\mu}+1}$.*

Proof: See [18]. ∎

The above lemma means that μ as a solution to the SSP problem on $\text{MDP}_{\mathbb{S}}$ (minimizing $\bar{\alpha}^{\mathbb{S}}_{\mu}$) corresponds to a solution for the MRP problem on $\text{MDP}_{\mathbb{M}}$ (maximizing $R^{\mathbb{M}}_{\mu}$). Note that the algorithm uses a sequence of simulated trajectories, each of which starting at $\mathbf{x}_0$ and ending as soon as $\mathbf{x}^*$ is visited for the first time in the sequence. Once a trajectory is completed, the state of the system is reset to the initial state $\mathbf{x}_0$ and the process is repeated. Thus, the actor-critic algorithm is applied to a modified version of the $\text{MDP}_{\mathbb{S}}$ where transition to a goal state is always followed by a transition to the initial state.

V. Case study

In this section we apply our algorithm to control a robot moving in a square-shaped mission environment, which is partitioned into 2500 smaller square regions (a 50×50 grid) as shown in Fig. 1. We model the motion of the robot in the environment as an MDP: each region corresponds to a state of the MDP, and in each region (state), the robot can take the following control primitives (actions): "North", "East", "South", "West", which represent the directions in which the robot intends to move (depending on the location of a region, some of these actions may not be enabled, for example, in the lower-left corner, only actions "North" and "East" are enabled). These control primitives are not reliable and are subject to noise in actuation and possible surface roughness in the environment. Thus, for each motion primitive at a region, there is a probability that the robot enters an adjacent region.

We label the region in the south-west corner as the initial state. We marked the regions located at the other three corners as the set of *goal* states as shown in Fig. 1. We assume that there is a set of *unsafe* states $\mathbb{X}_U$ in the environment (shown in black in Fig. 1). Our goal is to find the optimal policy that maximizes the probability of reaching a state in $\mathbb{X}_G$ (set of goal states) from the initial state (an instance of an MRP problem).

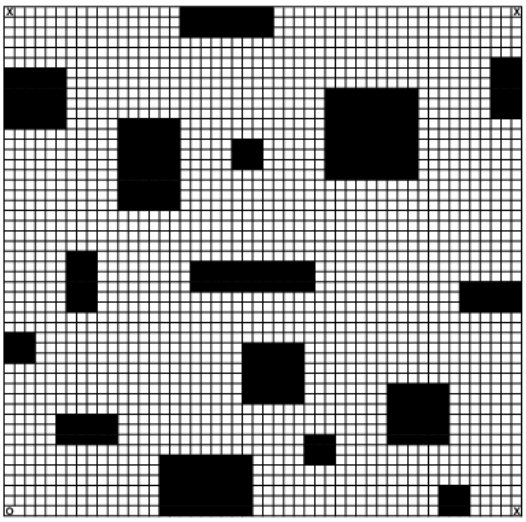

Fig. 1. View of the mission environment. The initial region is marked by o, the goal regions by x, and the unsafe regions are shown in black.

A. Designing an RSP

To apply the LSTD Actor-Critic algorithm, the key step is to design an RSP $\mu_{\boldsymbol{\theta}}(u|\mathbf{x})$. In this case study, we define the RSP to be an exponential function of two scalar parameters θ_1 and θ_2, respectively. These parameters are used to provide a balance between *safety* and *progress* from applying the control policy.

For each pair of states $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{X}$, we define $d(\mathbf{x}_i, \mathbf{x}_j)$ as the minimum number of transitions from $\mathbf{x}_i$ and $\mathbf{x}_j$. We denote $\mathbf{x}_j \in \mathscr{N}(\mathbf{x}_i)$ if and only if $d(\mathbf{x}_i, \mathbf{x}_j) \leq r$, where r is a fixed integer given apriori. If $\mathbf{x}_j \in \mathscr{N}(\mathbf{x}_i)$, then we say $\mathbf{x}_j$ is in the neighborhood of $\mathbf{x}_i$, and r represents the radius of the neighborhood around each state.

For each state $\mathbf{x} \in \mathbb{X}$, the safety score $s(\mathbf{x})$ is defined as the ratio of the safe neighboring states over all neighboring states of $\mathbf{x}$, namely,

$$s(\mathbf{x}) = \frac{\sum_{\mathbf{y} \in \mathscr{N}(\mathbf{x})} \mathbf{1}\{\mathbf{y} \in \mathbb{X} \setminus \mathbb{X}_U\}}{|\mathscr{N}(\mathbf{x})|} \tag{10}$$

where $\mathbf{1}\{\cdot\}$ denotes the indicator function. A higher safety score for the current state means it is less likely for the robot to reach an unsafe region in the future.

We define the progress score of a state $\mathbf{x} \in \mathbb{X}$ as $d_g(\mathbf{x}) := \min_{\mathbf{y} \in \mathbb{X}_G} d(\mathbf{x}, \mathbf{y})$, which is the minimum number of transitions from $\mathbf{x}$ to any goal region. We can now propose the RSP policy, which is a Boltzmann policy as defined in (1). Note that $\mathbb{U} = \{u_1, u_2, u_3, u_4\}$, which corresponds to "North", "East", "South", and "West", respectively. We first define

$$a_i(\mathbf{x}, \boldsymbol{\theta}) = F_i(\mathbf{x}) e^{\theta_1 E\{s(f(\mathbf{x}, u_i))\} + \theta_2 E\{d_g(f(\mathbf{x}, u_i)) - d_g(\mathbf{x})\}}, \tag{11}$$

where $\boldsymbol{\theta} := (\theta_1, \theta_2)$, $f(\mathbf{x}, u_i)$ denotes the next state from $\mathbf{x}$ when using control u_i, and $F_i(\mathbf{x})$ is an indicator function such that $F_i(\mathbf{x}) = 1$ if u_i is available at $\mathbf{x}$ and $F_i(\mathbf{x}) = 0$ if otherwise. Note that the availability of control actions at a state is limited for the states at the boundary. For example, at the initial state, which is at the lower-left corner, the set of available actions is $\{u_1, u_2\}$, corresponding to "North" and "East", respectively. If an action u_i is not available at state $\mathbf{x}$, we set $a_i(\mathbf{x}, \boldsymbol{\theta}) = 0$, which means that $\mu_{\boldsymbol{\theta}}(u_i|\mathbf{x}) = 0$.

Note that $a_i(\mathbf{x}, \boldsymbol{\theta})$ corresponds to applying control u_i and is defined to be the combination of the expected safety score of the next state, and the expected improvement in the progress score due to the transition from the current state to the next, weighted by θ_1 and θ_2, respectively. The RSP is then given by

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(u_i|\mathbf{x}) = \frac{a_i(\mathbf{x}, \boldsymbol{\theta})}{\sum_{i=1}^{4} a_i(\mathbf{x}, \boldsymbol{\theta})}. \tag{12}$$

We note that Assumption B holds for the proposed RSP. Moreover, Assumption A also holds, therefore Theorem II.1 holds for this RSP.

B. Generating transition probabilities

To implement the LSTD Actor-Critic algorithm, we first constructed the MDP. As mentioned above, this MDP represents the motion of the robot in the environment where each state corresponds to a cell in the environment (Fig. 1). To capture the transition probabilities of the robot from a cell to its adjacent one under an action, we built a simulator.

The simulator uses a unicycle model (see, *e.g.*, [19]) for the dynamics of the robot with noisy sensors and actuators. In this model, the motion of the robot is determined by specifying a forward and an angular velocity. At a given region, the robot implements one of the following four controllers (motion primitives) - "East", "North", "West", "South". Each of these controllers operates by obtaining the difference between the current heading angle and the desired heading angle. Then, it is translated into a proportional feedback control law for angular velocity. The desired heading angles for the "East", "North", "West", and "South" controllers are $0°$, $90°$, $180°$, and $270°$, respectively. Each controller also uses a constant forward velocity. The environment in the simulator is a 50 by 50 square grid as shown in Fig. 1. To each cell of the environment, we randomly assigned a surface roughness which affects the motion of the robot in that cell. The perimeter of the environment is made of walls, and when the robot runs to them, it bounces with the mirror-angle.

To find the transition probabilities, we performed a total of 5000 simulations for each controller and state of the MDP. In each trial, the robot was initialized at the center of the cell, and then an action was applied. The robot moved in that cell according to its dynamics and surface roughness of the region. As soon as the robot exited the cell, a transition was encountered. Then, a reliable center-converging controller was automatically applied to steer the robot to the center of the new cell. We assumed that the center-converging controller is reliable enough that always drives the robot to the center of the new cell before exiting it. Thus, the robot always started from the center of a cell. This makes the process Markov (the probability of the current transition depends only the control and the current state, and not on

the history up to the current state). We also assumed perfect observation at the boundaries of the cells.

It should be noted that, in general, it is not required to have all the transition probabilities of the model in order to apply the LSTD Actor-Critic algorithm, but rather, we only need transition probabilities along the trajectories of the system simulated while running the algorithm. This becomes an important advantage in the case where the environment is large and obtaining all transition probabilities becomes infeasible.

C. Results

We first obtained the exact optimal policy for this problem using the methods described in [2], [5]. The maximal reachability probability is 99.9988%. We then used our LSTD actor-critic algorithm to optimize with respect to $\boldsymbol{\theta}$ as outlined in Sec. III and IV.

Given $\boldsymbol{\theta}$, we can compute the exact probability of reaching $\mathbb{X}_G$ from any state $\mathbf{x} \in \mathbb{X}$ applying the RSP $\mu_{\boldsymbol{\theta}}$ by solving the following set of linear equations

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \sum_{u \in \mathbb{U}} \mu_{\boldsymbol{\theta}}(u|\mathbf{x}) \sum_{\mathbf{y} \in \mathbb{X}} p(\mathbf{y}|\mathbf{x}, u) p_{\boldsymbol{\theta}}(\mathbf{y}), \quad \text{for all } x \in \mathbb{X} \setminus (\mathbb{X}_U \cup \mathbb{X}_G) \qquad (13)$$

such that $p_{\boldsymbol{\theta}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathbb{X}_U$ and $p_{\boldsymbol{\theta}}(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathbb{X}_G$. Note that the system of equations in (13) contains exactly $|\mathbb{X}| - |\mathbb{X}_U| - |\mathbb{X}_G|$ number of equations and unknowns.

We plotted in Fig. 2 the reachability probability of the RSP from the initial state (i.e., $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$) against the number of iterations in the actor-critical algorithm each time $\boldsymbol{\theta}$ is updated. As $\boldsymbol{\theta}$ converges, the reachability probability converges to 90.3%. The parameters for this examples are: $r = 2$, $\lambda = 0.9$, $D = 5$ and the initial $\boldsymbol{\theta}$ is $(30, -10)$. We use (8) for β_k with $c = 1$.

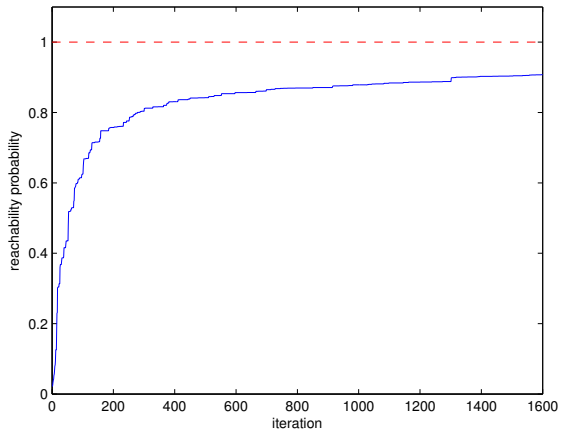

Fig. 2. The dashed line represents the optimal solution (the maximal reachability probability) and the solid line represents the exact reachability probability for the RSP as a function of the number of iterations applying the proposed algorithm.

VI. Conclusion

We considered the problem of finding a control policy for a Markov Decision Process (MDP) to maximize the probability of reaching some states of the MDP while avoiding some other states. We presented a transformation of the problem into a Stochastic Shortest Path (SSP) MDP and developed a new approximate dynamic programming algorithm to solve this class of problems. The algorithm operates on a sample-path of the system and optimizes the policy within a pre-specified class parameterized by a parsimonious set of parameters. Simulation results confirm the effectiveness of the proposed solution in robot motion planning applications.

References

[1] S. Temizer, M. Kochenderfer, L. Kaelbling, T. Lozano-Pérez, and J. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes."

[2] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, "Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees," in *IEEE Int. Conf. on Robotics and Automation*, Anchorage, AK, 2010, pp. 3227 – 3232.

[3] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*. Citeseer, 2007.

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

[5] X. Ding, S. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *IFAC*, 2011.

[6] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[7] K. Samejima and T. Omori, "Adaptive internal state space construction method for reinforcement learning of a real-world agent," *Neural Networks*, vol. 12, pp. 1143–1155, 1999.

[8] H. Berenji and D. Vengerov, "A convergent Actor-Critic-based FRL algorithm with application to power management of wireless transmitters," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 478–485, 2003.

[9] "Actor-critic models of reinforcement learning in the basal ganglia: From natural to artificial rats," *Adaptive Behavior*, vol. 13, no. 2, pp. 131–148, 2005.

[10] G. Gajjar, S. Khaparde, P. Nagaraju, and S. Soman, "Application of actor-critic learning algorithm for optimal bidding problem of a GenCo," *IEEE Transactions on Power Engineering Review*, vol. 18, no. 1, pp. 11–18, 2003.

[11] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[12] S. Bradtke and A. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 2, pp. 33–57, 1996.

[13] V. R. Konda, "Actor-critic algorithms," Ph.D. dissertation, MIT, Cambridge, MA, 2002.

[14] D. Bertsekas and S. Ioffe, "Temporal differences-based policy iteration and applications in neuro-dynamic programming," LIDS REPORT, Tech. Rep. 2349, 1996, mIT.

[15] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, pp. 1180–1190, 2008.

[16] D. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[17] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[18] R. M. Estanjini, X. C. Ding, M. Lahijanian, J. Wang, C. A. Belta, and I. C. Paschalidis, "Least squares temporal difference actor-critic methods with applications to robot motion control," 2011, available at http://arxiv.org/abs/1108.4698.

[19] S. LaValle, *Planning algorithms*. Cambridge University Press, 2006.