2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Sampling-based approximate optimal temporal logic planning

Lening Li[1] and Jie Fu[2]

Abstract— **In this paper, we propose a sampling-based policy iteration for optimal planning under temporal logic constraints. The method integrates approximate optimal control, importance sampling, and formal methods. For a subclass of linear temporal logic, the planning problem is transformed to an optimal control problem for a hybrid system where discrete transitions are triggered by linear time events in temporal logic. Instead of solving the Hamilton-Jacobi-Bellman equation, we use policy function approximation to reduce the problem into a search of an optimal weight vector that parametrizes the near-optimal policy for given bases. Then, we incorporate Model Reference Adaptive Search — an importance sampling-based optimization algorithm to perform a sample-efficient search within the parameter space of policy function approximations. Facing the discontinuity in cost function introduced by temporal logic constraints and system dynamics, we introduce 1) a rank function in formal logic specifications to enable sample-efficient search; 2) specification-guided basis selection. Under mild technical assumptions, the proposed algorithm converges, with probability one, to a global approximate optimal policy that ensures the satisfaction of temporal logic constraints. The correctness and efficiency of the method are demonstrated through numerical experiments including temporal logic planning for a linear system and a nonlinear mobile robot.**

I. Introduction

In this work, we propose a novel sampling-based planning algorithm for nonlinear systems subject to a subclass of temporal logic constraints, i.e. co-safe linear temporal logic (LTL) [19]. LTL is an expressive language that allows one to specify system specifications, including traditional reaching-a-goal, stability, obstacle avoidance, sequentially visiting interesting regions, periodic surveillance, and conditional reactive behaviors. Given a specification in temporal logic, control and planning for continuous and nonlinear systems are generally performed by first computing a discrete transition system that abstracts the system dynamics, and then performs temporal logic planning in the discrete state space computed from a product of the transition system and the automaton representing the LTL constraints. The abstraction-based synthesis methods have been studied extensively for continuous linear and nonlinear systems [15], [7], [8], [21], [1], [26], [30].

The class of abstraction-based methods suffer from the issue of scalability and the procedure of abstraction can be computationally expensive. Temporal logic specifications introduce additional complexity because planning is performed in the product of the system states and the automaton states, which correspond to subformulas that characterize different stages in satisfying the temporal logic specifications. Comparing to discretiziation-based methods, such as A* [9], D* [27], and their variants, sampling-based methods are promising in solving high-dimensional planning. Algorithms in this class include Rapidly-exploring random tree(RRT) [20] and Probabilistic roadmaps(PRM) [14] with their variants, including CBiRRT [4] and RRT* [13]. These methods have been extended to solve temporal logic planning problems [12], [29]. Sampling-based methods are probabilistically complete in the sense that they calculate a solution given infinite samples in the configuration space. Moreover, RRT* ensures the asymptotic convergence to the optimal solution. However, the assumption on Lipschitz continuous cost function of RRT* is hard to satisfy under sensitive and stringent temporal logic constraints, for that two trajectories that are close to each other may satisfy different temporal logic specifications.

In this paper, a sampling-based algorithm is presented to perform sampling not in the state or configuration space but in a space of weight vectors that parametrize the policy function approximation. The main motivation is to remove, or at least weaken, the dependency of planning complexity on the dimensionality of the system. In recent work [25], the authors exploit the idea that continuous time systems constrained by LTL can be formulated as a hybrid dynamical system by augmenting the continuous state space with the discrete states. For the class of linear quadratic systems subject to co-safe LTL formulas, the optimal control problem is soluble through solving a sum-of-square program.

Here, we consider policy function approximations instead of value function approximation for temporal logic planning for nonlinear systems and propose a novel sampling-based policy iteration algorithm. The key idea is to regard planning as a problem to infer, from sampled trajectories, a weight vector that parametrizes the optimal feedback policy function approximation for given bases. Under this consideration, we integrate Model Reference Adaptive Search (MRAS) [11], an importance sampling-based global optimization algorithm, to search the optimal weight vector through simulated runs. Previously, the Cross entropy (CE) method has been used for trajectory planning [16], [22], where the objective is to find a sequence of motion primitives or a sequence of states for interpolation-based planning. In stochastic policy optimization [17], CE is used for computing linear feedback policies.

This paper is the first to systematically integrate importance sampling, temporal logic planning, and approximate optimal control. The main contributions include the follow-

[1]Lening Li is with Robotics Engineering Program, Worcester Polytechnic Institute, 01609, Worcester, MA, US. `lli4@wpi.edu`

[2]Jie Fu is with Robotics Engineering Program, Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, 01609, Worcester, MA, US. `jfu2@wpi.edu`

ing: 1) We develop a sampling-based planning algorithm for nonlinear systems subject to co-safe LTL constraints. The algorithm generates provably correct policies quickly and continues to improve the optimality through iterative sampling while adapting the sample distribution. 2) We employ rank functions in the specification to guide sample-efficient planning. We introduce specifications dependent bases selection to reduce the complexity of the policy function approximation. Based on the experimental evaluations, we discuss the advantages and limitations of the proposed method and propose the future directions.

II. Preliminaries

Notations: For a finite alphabet Σ, Σ^* is the set of finite words generated using symbols in Σ. The product between two vectors u, v are given by $u^\intercal v$ or $\langle u, v\rangle$. Given a symmetric matrix R, $\|x\|_R = x^\intercal R x$ is the weighted norm. For a real number r, ceiling of r denotes as $\lceil r \rceil$, where $r \in \mathbb{R}$.

A. System model

We consider continuous-time nonlinear systems of the form

$$\begin{aligned}\Sigma: \quad & \dot{x}(t) = f(x(t), u(t)), \\ & x(t) \in X, u(t) \in U.\end{aligned} \tag{1}$$

where $x \in X$ is the state, $u \in U$ is the control input, $x_0 \in X$ is the initial state, and $f(x, u)$ is a vector field. We assume that X and U are compact. Given a finite time interval $[0, T]$, a controller $u : [0, T] \rightarrow U$ is a continuous function that maps time t to control input in $u(t) \in U$ for $t \in [0, T]$. $\mathbf{Traj}(x_0, [0, T], u)$ refers to the trajectory of the continuous system with initial condition x_0 under the control input $u(t)$ over the time interval $[0, T]$.

B. System specification: Temporal logic

The system (1) is constrained to satisfy a specification on the discrete behavior obtained from its continuous trajectory.

First, let $\mathcal{AP}$ be a set of atomic propositions evaluated over the state space X. Given a proposition $p \in \mathcal{AP}$ and a state $x \in X$, the value of p at x, denoted $\mathsf{val}(p, x)$, is a Boolean indicating whether p is evaluated true (1) or false (0) at x. Then, we define a labeling function $L : X \rightarrow 2^{\mathcal{AP}}$, which maps a continuous state $x \in X$ to a finite set in $2^{\mathcal{AP}}$ of atomic propositions that evaluate to be true at x. This function partitions the continuous space X into regions that share the same truth values in $\mathcal{AP}$. The labeling function also links the continuous system with its *discrete behavior*.

Definition 1: Let $t_0, t_1, \ldots, t_N$ be *times of discrete transitions*, that satisfy

- $0 = t_0 < t_1 < \cdots < t_N = T$,
- $L(x(t)) = L(x(t_k))$, $t_k \leq t < t_{k+1}$, $k = 0, \ldots, N$,
- $L(x(t_k^-)) \neq L(x(t_k^+))$, $k = 0, \ldots, N$.

A *discrete behavior*, denoted as $\mathbb{B}(\mathbf{Traj}(x_0, [0, T], u))$, is defined to be a discrete word $\sigma_0\sigma_1 \ldots \sigma_N \in (2^{\mathcal{AP}})^*$, where $\sigma_k = L(x(t_k))$.

A specification on a discrete behavior can be written as a co-safe LTL formula over the finite set of atomic propositions (for a comprehensive description of the syntax and semantics of LTL, readers can refer to [23], [2]). A co-safe LTL formula is an LTL formula where every satisfying word has a finite good prefix [1] [18]. We restrict to such formulas to take advantages of the expressiveness of temporal logic for specifying optimal control problems without imposing infinite Büchi acceptance conditions. Essentially, co-safe LTL formula describes tasks that can be completed in a finite time. It allows us to specify tasks such as reaching a goal, safety until reaching the goal, coverage task with or without an sequential ordering of regions to be visited, etc. In the example section, we provide several examples of co-safe LTL specifications.

Given a co-safe LTL specification φ over the set of atomic propositions $\mathcal{AP}$, there exists a corresponding deterministic finite-state automaton (DFA) $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, \delta, q_0, F\rangle$, where Q is a finite set of states (modes), $2^{\mathcal{AP}}$ is a finite alphabet, $\delta : Q \times 2^{\mathcal{AP}} \rightarrow Q$ is a *deterministic* transition function such that when the symbol $\sigma \in 2^{\mathcal{AP}}$ is read at state q, the automaton makes a deterministic transition: $\delta(q, \sigma) = q'$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of final, or *accepting* states. The transition function is extended to a sequence of symbols, or a *word* $w = \sigma_0\sigma_1 \ldots \sigma_n \in \Sigma^*$, $n \in N$, in the usual way: $\delta(q, \sigma_0 v) = \delta(\delta(q, \sigma_0), v)$ for $\sigma_0 \in \Sigma$ and $v \in \Sigma^*$. We say that the finite word w satisfies φ if and only if $\delta(q_0, w) \in F$. The set of words satisfying φ is the *language* of the automaton $\mathcal{A}_\varphi$, denoted by $\mathcal{L}(\mathcal{A}_\varphi)$. The construction of A_φ for a given co-safe LTL formula φ can be automated using the tool in [24].

The discrete behavior encodes the sequence of labels visited by the state as it moves along its continuous trajectory. Specifically, the atomic propositions are evaluated only at the times when the label changes value. Thus, a trajectory $\mathbf{Traj}(x_0, [0, T], u)$ satisfies an LTL specification φ, denoted as $\mathbf{Traj}(x_0, [0, T], u) \models \varphi$ if and only if its discrete behavior is in the language $L(\mathcal{A}_\varphi)$.

Problem 1: Given a system in (1) and a co-safe LTL formula φ, computing a controller u that solves the following constrained optimal planning problem:

$$\begin{aligned}\min_{u:[0,T]\rightarrow U} J(x_0, u) &= \int_0^T \ell(x(t), u(t))dt + g(x(T), u(T)) \\ \text{subject to: } & \dot{x}(t) = f(x(t), u(t)), \\ & \mathbf{Traj}(x_0, [0, T], u) \models \varphi; \\ & x(t) \in X,\ u(t) \in U,\ x(0) = x_0.\end{aligned} \tag{2}$$

where T is the stopping time, $\ell : X \times U \rightarrow \mathbb{R}^+$ defines the running cost when the state trajectory traverses through x and the control input u is applied, and $g : X \times U \rightarrow \mathbb{R}^+$ defines the terminal cost. As an example, a running cost function can be a quadratic cost: $\ell(x, u) = \|x\|_R + \|u\|_Q$ for some

[1]Given a word $w \in \Sigma^*$ and $v \in \Sigma^*$, v is a prefix of w if and only if $w = vu$ for some $u \in \Sigma^*$. The word u is called the suffix of w.

positive semi-definite matrices R and Q, and a terminal cost can be $g(x,u) = \|x - x_f\|_R$, where x_f is a goal state.

C. Preliminary: Model Reference Adaptive Search

MRAS algorithm [11] aims to solve the following problem:

$$z^* \in \arg\max_{z \in Z} H(z), \quad z \in \mathbb{R}^n$$

where Z is the solution space and $H : \mathbb{R}^n \to \mathbb{R}$ is a deterministic function that is bounded from below. It assumes that the optimization problem has a unique solution, i.e., $z^* \in Z$ and for all $z \neq z^*$, $H(z) < H(z^*)$.

The following regularity conditions need to be met for the applicability of MRAS.

Assumption 1: For any given constant $\xi < H(z^*)$, the set $\{z \mid H(z) \geq \xi\} \cap Z$ has a strictly positive Lebesgue or discrete measure.
This condition ensures that any neighborhood of the optimal solution z^* will have a positive probability to be sampled.

Assumption 2: For any constant $\delta > 0$, $\sup_{z \in A_\delta} H(z) < H(z^*)$, where $A_\delta := \{z \mid \|z - z^*\| \geq \delta\} \cap Z$, and we define the supremum over the empty set to be $-\infty$.
Next, we present the MRAS algorithm which has the following key steps:

- Select a sequence of reference distributions $\{g_k(\cdot)\}$ with desired convergence properties. Specifically, the sequence $\{g_k(\cdot)\}$ will converge to a distribution that concentrates only on the optimal solution z^*.
- Select a parametrized family of distribution $f(\cdot, \theta)$ over Z with parameter $\theta \in \Theta$.
- Optimize the parameters $\{\theta_k\}$ iteratively by minimizing the following KL distance between $f(\cdot, \theta_k)$ and $g_k(\cdot)$.

  $$d(g_k, f(\cdot, \theta)) := \int_{\mathcal{Z}} \ln \frac{g_k(z)}{f(z, \theta)} g_k(z) \nu(dz).$$

  where $\nu(\cdot)$ is the Lebesgue measure defined over X. The sample distributions $\{f(\cdot, \theta_k)\}$ can be regarded as compact approximations of the reference distributions and will converge to an approximate optimal solution as $\{g_k(\cdot)\}$ converges provided with certain properties of $\{g_k(\cdot)\}$ which is retained in $f(\cdot, \theta_k)$.

Note that the reference distribution $\{g_k(\cdot)\}$ is unknown beforehand as the optimal solution is unknown. Thus, the MRAS algorithm employs the estimation of distribution algorithms [10] to estimate a reference distribution from elite samples (similar to CE) that guides the search. It is shown [11] that MRAS has better convergence rates and stronger guarantees than CE over several benchmark examples. To make the paper self-contained, we will cover some details of MRAS in the development of the planning algorithm.

III. Main results

A. Problem formulation

Given a DFA $\mathcal{A}_\varphi = \langle Q, 2^{\mathcal{AP}}, \delta, q_0, F \rangle$ representing the co-safe LTL formula, it can be shown that the optimal controller for infinite horizon is a *hybrid feedback policy* $u : X \times Q \to U$ that takes the current continuous state x and specification state q, outputs a control input $u(x,q)$. In other words, the control input is based on the feedback from both continuous state and discrete specification state. We denote the set of hybrid feedback policies to be Π.

For finite time horizon, the feedback controller can be dependent on time. In the scope of this work, we only consider controllers that do not depend on time explicitly by planning for a long horizon $[0, T]$ [5]. Extensions to time-varying controller will be discussed.

The temporal logic planning problem in (2) can be expressed as

$$\begin{aligned}
&\min_{u \in \Pi} J(x_0, u) \\
&\qquad = \int_0^T \ell(x(t), u(t)) dt + g(x(T), u(T)) \\
&\text{subject to: } \dot{x}(t) = f(x(t), u(t)), \\
&\qquad q(t_k)^+ = \delta(q(t_k)^-, L(x(t_k))), \\
&\qquad 0 \leq t_0 < t_1 < \ldots < t_H = T, \text{ for some } H \in \mathbb{N}, \\
&\qquad x(t) \in X, \ u(t) \in U, \\
&\qquad x(0) = x_0, q(t_0) = \delta(q_0, L(x(0))), q(T) \in F,
\end{aligned} \tag{3}$$

where $t_i, i = 0, \ldots, H$ are times of discrete transitions, $q(t)^-$ (resp., $q(t)^+$) is the specification state before (resp., after) a discrete transition at time t.

Problem (3) is indeed an optimal planning problem in a hybrid system over the hybrid state space $X \times Q$. A subsystem at discrete state q evolves according to the system dynamics in (1), and stays within the same discrete state as long as the label of the state does not change. When the label changes at time t_k, for some $k \geq 0$, the system takes a discrete transition to a discrete mode $q' = \delta(q, L(x(t_k)))$ and continue to evolve given the system dynamics in (1). A formal construction of this hybrid system is given in our previous work [25].

B. Approximate optimal planning with policy function approximations

Definition 2: Given an automaton state q of A_φ, a *local, time-invariant feedback policy* is a function $u(\cdot, q) : X \to U$ that maps a continuous state $x \in X$ to a control input $u \in U$. Let $\phi_q = [\phi_{1,q}, \phi_{2,q}, \ldots, \phi_{N_q,q}]^\intercal$ be a vector of total $N_q \in \mathbb{N}$ basis functions, a *local policy approximation* is a weighted sum of bases, denoted as $\langle w_q, \phi_q \rangle$, where $w_q = [w_{1,q}, w_{2,q}, \ldots, w_{N_q,q}]^\intercal$ are a vector of weights and $w_{i,q}$ is the weight of the basis $\phi_{i,q}$.
An example of basis functions can be a polynomial basis vector $\phi = [1, x, x^2, x^3, \ldots, x^N]^\intercal$. A commonly used class of basis functions is radial basis function (RBF). It can be constructed by determining a set of centers $c_1, \ldots, c_N \in X$, and then constructing RBFs $\phi_i = \exp(-\frac{\|x - c_i\|^2}{2\sigma^2})$, for each center c_i, where σ is a pre-defined parameter.

Suppose the set of specification states is $Q = \{q_0, q_1, \ldots, q_N\}$, we have $w = [w_{q_0}; \ldots; w_{q_N}]$ and $\phi = [\phi_{q_0}; \ldots; \phi_{q_N}]$, then $u(\cdot, q) = \langle w_q, \phi_q \rangle$ is the q-th entry in the vector $w^\intercal \phi = [w_{q_0}^\intercal \phi_{q_0}, \ldots, w_{q_N}^\intercal \phi_{q_N}]^\intercal$. For simplicity, we call w as the *weight vector* and the vector ϕ of bases as the *basis vector*[2] and the vector ϕ of bases as the *basis vector*[2]. We can write a controller compactly as $u = \langle w, \phi \rangle$.

Given the domain W of the weight vector, when the basis vector ϕ is pre-defined, the search space for control policies is then restricted to $\Pi_\phi = \{\langle w, \phi \rangle \mid w \in W, \langle w, \phi \rangle \in \Pi\}$. For input constraints, for example, input saturation, we will project the computed control input at time t on the constrained input space U during the simulation-based planning. The projection operation is defined to be $\mathsf{Proj}_U : \mathbb{R}^m \to U$ where m is the dimensionality of the input space.

Clearly, for any weight vector w, $J(x_0, \langle w, \phi \rangle) \geq \min_{u \in \Pi} J(x_0, u)$. Thus, we aim to solve for a controller within Π_ϕ that minimizes the total cost while satisfying the system constraints, so as to minimize the error in the optimal cost introduced by policy function approximation.

Definition 3 (Approximate optimal feedback policy): Given a basis vector ϕ, a weight vector w^* with respect to ϕ is *optimal* if and only if for all $w \in W$ such that $\langle w, \phi \rangle \in \Pi_\phi$, $J(x_0, \langle w^*, \phi \rangle) \leq J(x_0, \langle w, \phi \rangle)$. The approximate optimal feedback policy is $\langle w^*, \phi \rangle$.
It is straightforward to show the optimal weight w^* gives rise to a controller $\langle w^*, \phi \rangle$ that minimizes the error introduced by policy function approximation, i.e., the error between the global optimal cost and the optimal cost that can be achieved with a controller in Π_ϕ.

For clarity in notation, we denote $J(x_0, \langle w, \phi \rangle)$ by $J(x_0; w)$ as ϕ is a fixed basis vector throughout the development of the proposed method.

The optimal weight vector w^* w.r.t. ϕ is the solution to the following problem:

$$\begin{aligned}
&\min_{w \in W} J(x_0; w) \\
\text{subject to: } & \dot{x}(t) = f(x(t), u(t)), \\
& q(t_k^+) = \delta(q(t_k^-), L(x(t_k^-))), \\
& 0 \leq t_0 < t_1 < \ldots < t_H = T, \text{ for some } H \in \mathbb{N}, \\
& x(t) \in X, \ u(t) = \mathsf{Proj}_U(\langle w_{q(t)}, \phi_{q(t)}(x(t)) \rangle), \\
& x(0) = x_0, q(t_0) = \delta(q_0, L(x(0))), q(T) \in F,
\end{aligned} \tag{4}$$

where the constraints are the same in (3).

Moreover, if the actual optimal policy u can be represented by a linear combination of selected basis functions and a weight vector, then we obtain the optimal policy u^* by solving (4) and obtain the optimal weight vector, i.e., $u^* = \langle w^*, \phi \rangle$.

To prevent Zeno, i.e., infinite transitions in the specification automaton for any finite time, we add an additional transition cost in the cost function such that a Zeno behavior will be penalized for infinite cost. The total cost function is $J(x_0, u) = \int_{t=0}^T \ell(x, u) dt + g(x(T), u(T)) + \sum_{k=1}^H s(q(t_k)^-, q(t_k)^+)$, where $s : Q \times Q \to \mathbb{R}$ is a nonzero discrete transition cost that is incurred at the discrete transition times.

[2]w (ϕ) is indeed a vector of vectors, *i.e.*, matrix

C. Importance sampling-based temporal logic planning

The key idea of MRAS is based on updating the sampling distribution so as to have much mass concentrating on the elite samples, i.e., a set of controllers that satisfy the specification with lower costs.

First, we select multivariate Gaussian as the probability distribution which is used to sample in the weight vector space. Recall that the probability density of a multivariate Gaussian distribution is defined by

$$\begin{aligned}
& p(w; \theta) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} \exp(-\frac{1}{2}(x - \mu)^\intercal \Sigma^{-1} (x - \mu)), \\
& \theta = (\mu, \Sigma), \forall w \in W,
\end{aligned}$$

where μ is the mean vector and Σ is the covariance matrix, N is the dimension of weight vector $w \in W$, and $|\Sigma|$ is the determinant of Σ.

The main algorithm is described as follows:

1) **Initialization**: We select an initial distribution over weight vectors W, denoted $p(\cdot, \theta_0)$, for some $\theta_0 \in \Theta$. We specify a *quantile parameter* $\rho \in (0, 1]$, a small real $\varepsilon \in \mathbb{R}^+$ called *improvement parameter*, an initial sample size N_0, a *sample increment parameter* α, a *smoothing coefficient* $\lambda \in (0, 1]$, and a strictly decreasing and positive function $S : \mathbb{R} \to \mathbb{R}^+$ if the objective is to minimize the cost [3]. We select an initial distribution μ_1 and Σ_1 that allows the sampling to have a near uniform coverage of the sample space. Let $k = 1$ and move to step 2).
2) **Sampling**: At each iteration k, given the current distribution $p(\cdot, \theta_k)$, we generate a set of N_k samples W. For each $w \in W$ in the sample set, we evaluate the cost $J(x_0; w)$ from the initial state x_0 by simulating the controller $u = \langle w, \phi \rangle$ with system model in (1). The cost is determined because the system is deterministic and has a unique solution.
3) **Rejecting unsatisfiable policies**: Next, we eliminate any trajectory that violates the temporal logic constraints by directly rejecting its corresponding weight vector. The remaining set is W_{sat}.
4) **Select Elite samples and threshold**: Next, the set $\{J(x_0; w) \mid w \in W_{\mathsf{sat}}\}$ is ordered from largest (worst) to smallest (best) among given samples:

$$J_{k,(0)} \geq \ldots \geq J_{k,(N_{sat})} \text{ ,where } N_{sat} = |W_{sat}|.$$

We denote κ to be the estimated $(1 - \rho)$-quantile of cost $J(\cdot; w)$, i.e., $\kappa = J_{k, \lceil (1-\rho) N_{sat} \rceil}$.
The following cases are distinguished.
   - If $k = 0$, we introduce a threshold $\gamma = \kappa$.
   - If $k \neq 0$, the following cases are further distinguished:
     - $\kappa \leq \gamma - \varepsilon$, i.e., the estimated $(1 - \rho)$-quantile of cost has been reduced by the amount of ε from the last iteration, then γ is updated to equal κ. Let $N_{k+1} = N_k$ and continue to step 5).

[3]Possible choices can be $S(x) = \exp(-x)$ or $S(x) = \frac{1}{x}$ if x is strictly positive.

- Otherwise, if $\kappa > \gamma - \varepsilon$, we find the largest ρ', if it exists, such that the estimated $(1-\rho')$-quantile of cost $\kappa' = J_{k,\lceil(1-\rho')N_{sat}\rceil}$ satisfies $\kappa' \leq \gamma - \varepsilon$. Then we update γ with κ' and also set ρ to be ρ'. Let $N_{k+1} = N_k$ and continue to step 5). However, if no such ρ' exists, then there is no updating on the threshold γ and the sample size is increased by α-percentile, i.e., for the next iteration, we obtain $N_{k+1} = \lceil(1+\alpha)N_k\rceil$ samples from the distribution, where $\alpha \in [0,1)$. In the end, let $\theta_{k+1} = \theta_k$, $k = k+1$ and continue to step 2).

5) **Parameter update**: We update parameter θ_{k+1} for iteration $k+1$. First, we define a set $E = \{w \mid J(x_0; w) \leq \gamma, w \in W_{\mathsf{sat}}\}$ of *elite samples*. Note that the parameter update in θ is to ensure a higher probability for elite samples. To achieve that, for each elite sample $w \in E$, we associated a weight $S(J(x_0;w))^k/p(w,\theta_k)$. By doing so, a higher weight is associated with a weight vector with a lower cost and a lower probability in the current distribution. The next parameter θ_{k+1} is selected to maximize the weighted sum of probabilities of elite samples. To this end, we have

$$\theta^*_{k+1} = \arg\max_{\theta\in\Theta} \mathbb{E}_{\theta_k}\left(\frac{S(J(x_0;w))^k}{p(w,\theta_k)} I_{w\in E}\cdot p(w,\theta)\right) \tag{5}$$

The solution θ^*_{k+1} is used in step 2) for next sampling iteration.

For parameter update, we implement the Monte-Carlo version of (5),*i.e.*, $\theta^*_{k+1} \approx (\mu_{k+1}, \Sigma_{k+1})$ where

$$\mu_{k+1} = \frac{\mathbb{E}_{\theta_k}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E} w}{\mathbb{E}_{\theta_k}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}} \approx \frac{\sum_{w\in W}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E} w}{\sum_{w\in W}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}},$$

and $\Sigma_{k+1} =$

$$\frac{\mathbb{E}_{\theta_k}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}(w-\mu)(w-\mu)^\intercal}{\mathbb{E}_{\theta_k}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}} \approx \frac{\sum_{w\in W}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}(w-\mu)(w-\mu)^\intercal}{\sum_{w\in W}[S(J(x_0,w))^k/p(w,\theta_k)]I_{w\in E}},$$

where we approximate $\mathbb{E}_{\theta_k}(h(\mathbf{w}))$ with its estimate $\frac{1}{N_k}\sum_{w\in W} h(w)$ for $\mathbf{w} \sim p(\cdot,\theta_k)$, and $I_{w\in E}$ is the indicator function which equals 1 if $w \in E$ and 0 otherwise.

Further, we use a smoothing update and let $\theta_{k+1} = \lambda\theta_k + (1-\lambda)\theta^*_{k+1}$.

6) **Stopping criterion**: We stop the iteration if the covariance matrix Σ_k becomes near-singular.

Next, we show the global convergence of the algorithm is ensured by the property of MRAS.

Theorem 1 (Theorem 1 [11]): Under assumption 1,2, for continuous optimization problem in $\mathbb{R}^n$, if multivariate normal p.d.f.'s are used in MRAS, i.e., $\theta_k = (\mu_k, \Sigma_k)$, with probability one,

$$\lim_{k\to\infty} \mu_k = w^* \text{ and } \lim_{k\to\infty} \Sigma_k = 0_{n\times n}.$$

provided that (5) is solved exactly.

Using the multivariate Gaussian, we ensure that the algorithm converges to the global optimal solution provided with infinite number of samples. However, in practice, only finitely many samples are used. Similar to CE method, the algorithm may converge to sub-optimal solution.

The choice for the sample size N and the parameters in the algorithm depends on the size of the problem and the number of parameters. The reader is referred to [6], [11] for more details about the choice of sample size and parameters.

D. Rank-guided adaptive policy search

Under complex temporal logic constraints and large state space, a high-dimensional weight vector must be used to achieve near-optimal approximation of the policy function. In such a case, it is hard to find a policy that satisfies the temporal logic constraint, which can be considered as a rare event in the sampling-based approach.

To address this issue, we propose a rank-guided adaptive policy search which incorporates additional terminal cost associated with the rank of the specification states. Then, for a policy that does not satisfy the constraint, instead of rejecting its corresponding weight vector, we penalize the weight by adding a terminal cost, depending on a minimal number of transitions that remain to be triggered in order to satisfy the specification.

Formally, we introduce a rank function $\mathrm{rank}: Q \to \mathbb{N}$ that maps each automaton state $q \in Q$ to a non-negative integer and is defined iteratively by

1) $\mathrm{rank}(q) = 0$ for all $q \in F$ where F is the set of accepting states.
2) $\mathrm{rank}(q) = \min_{q'\in Q}\{\mathrm{rank}(q') + 1 \mid \exists A \in 2^{AP}, \delta(q,A) = q'\}$. Intuitively, the rank function corresponds to the minimum number of actions needed to make to reach the accepting states.

Given a weight w, using simulation we obtain a finite-time trajectory $\mathbf{Traj}(x_0,[0,T],u)$ where $u = \langle w, \phi\rangle$ such that $\delta(q_0, L(\mathbf{Traj}(x_0,[0,T],u)) = q(T) = q \notin F$, we add the total cost under this controller with a terminal cost $h(q)$ where $h: Q \to \mathbb{R}$ is defined by

$$h(q) = \begin{cases} 0 & \text{for all } q \in F. \\ \mathrm{rank}(q)\times c & \text{otherwise.}\end{cases}$$

where c is a constant. We select a constant c that is greater or equal to an estimated upper bound of the cost that can be incurred when the system trajectory triggers a transition in the specification automaton.

Next, we introduce a method of basis selection that explicitly accounts for the temporal logic specification φ.

Definition 4: Given a basis vector ϕ_q, two state x and x' are *ϕ_q-equivalent* if and only if they have the same evaluation for the bases, i.e., $\phi_q(x) = \phi_q(x')$.

For ϕ_q-equivalent states, the controller $w_q^\intercal\phi_q : X \to U$ will not distinguish these states during its execution. This leads to an insight that the choice of bases must distinguish two states whose labels lead to different automaton state.

To this end, for each label $\sigma \subseteq \mathcal{AP}$, we associate a finite vector of bases ϕ_σ such that for any $x, x' \in X$, if $\mathsf{val}(p,x) \neq \mathsf{val}(p,x')$, then $\phi_\sigma(x) \neq \phi_\sigma(x')$. Finally, given an automaton state $q \in Q$, the basis vector ϕ_q must include all elements in ϕ_σ, for any σ such that $\delta(q,\sigma) \neq q$. We say that such a choice of bases is *specification-dependent*. Besides, we need to include a set of basic bases in ϕ_q for all $q \in Q$ to have a good approximation of the optimal control policy.

In the special case, when each atomic proposition $p_i := c_i x \leq b_i$ is a linear inequality and states satisfies p_i constitute a half-space of $\mathbb{R}^N$. A subset $\sigma \subseteq \mathcal{AP}$ of atomic propositions may describe a polytope P_σ. A specification-dependent basis vector for P_σ of k-dimension can be computed by including at least $k-1$ basis vectors $\phi_{i,\sigma} = f(x - v_i)$ where $f(\cdot)$ is a surjective function and v_i, $i = 1, \ldots, k-1$ of different vertices's of the polytope, because any two states differ in at least one value of basis $\phi_{i,\sigma}$ for some $i = 1, \ldots, k-1$.

Remark: To reduce sample complexity, it is critical to select basis vectors of reasonable sizes in order to achieve a trade-off between minimizing approximation error and improving efficiency of the planning algorithm. In general, basis selection problem is important and challenging in function approximation and has been studied in different problem domains [31], [28]. A rigorous analysis of basis selection for approximate-optimal temporal logic planning subjects to future research.

Complexity analysis: The rate of convergence for MRAS is hard to quantify. However, since the samples are continuous controllers, the complexity of generating a continuous trajectory is time linear in the length of the trajectory. Checking if a trajectory satisfies the specification is done by simultaneously computing the discrete transitions in the specification automaton. Overall the complexity of each iteration is linear in the product of sample size, the size of A_φ, and the length of the trajectory. It is worthy to mention that the trajectory evaluation can be performed in parallel, and thus allow linear speedup with parallel computing.

IV. Simulation experiments

In this section, we experimentally evaluate the correctness and efficiency of the proposed algorithm over two examples. The first one is a linear system, for which temporal logic planning is usually performed with a finite-state transition system that abstracts the linear system dynamics [3]. The other one is temporal logic planning for a Dubins car. All experiments are running in MATLAB on Ubuntu 14.04. The critical specifications of the desktop are the following: CPU: Intel Xeon E5 @ 3.10GHz; System Memory: 16 GB.

A. Case study: Optimal Control of Linear systems under LTL constraints

Consider a linear system's dynamics: $\dot{x} = Ax + Bu$, where $x \in X$, $A = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$, $B = [0,1]^\intercal$. The cost function includes the running cost: $\int_0^T \|x\|_R + \|u\|_Q dt$, where $R = Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$ and $T = 15$, the terminal cost: $g(x(T), u(T)) = 10^2 \times \|x(T) - x_f\|$, where $x_f = (0,0)$ is the goal position. The initial position of this system is set as $x_0 = (10,0)$, and the main goal is to stabilize this system to the origin $(0,0)$ while satisfying the specifications: $\Diamond(A \wedge \Diamond(B \wedge \Diamond C)) \vee \Diamond(C \wedge \Diamond(B \wedge \Diamond A)) \wedge \Box\neg obs$ [4], i.e., eventually visit A and C regions and visit region B at some point between these two visits (see labels in Fig. 2), while avoiding the static obstacle marked in blue. The corresponding automaton is shown in Figure 1, where the transitions to a sink & non-accepting state with label obs are omitted. Using the computation of rank function, we have $Q_0 = \{5\}$, $Q_1 = \{2,4\}$, $Q_3 = \{1,3\}$ and $Q_4 = \{0\}$. Let $c = 10^4$, we determine terminal cost function $h(q) = \mathrm{rank}(q) \times c$.

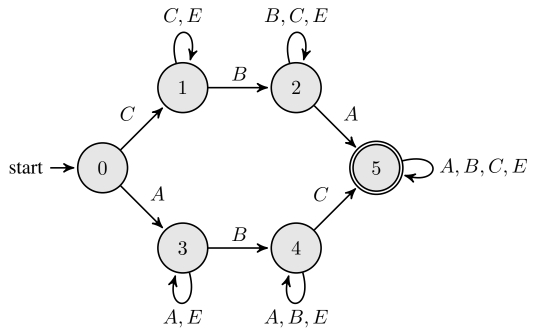


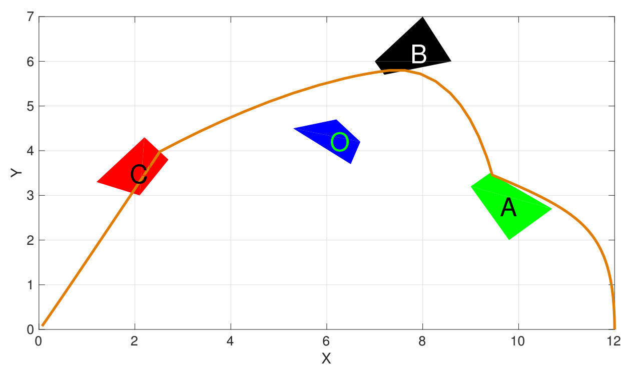
Fig. 1. Automaton $\Diamond(A \wedge \Diamond(B \wedge \Diamond C)) \vee \Diamond(C \wedge \Diamond(B \wedge \Diamond A)) \wedge \Box\neg obs$, where E denote everything else, including hitting the obstacle.

For basis selection, two class of bases are used: basic ones and specification-dependent bases. Here, we choose linear basis $\phi_{basic} = [1, x(1), x(2)]$ for basic ones. Then, we select a point inside each polygon region A, B, C, and the obstacle obs, denoted x_A, x_B, x_C, x_{obs} and let $\phi_\sigma = [x(1) - x_\sigma(1), x(2) - x_\sigma(2)]$ for $\sigma = A, B, C, \mathrm{obs}$. Next, for state $q \in Q$ of the automaton, we select $\phi_q = [\phi_{basic}^\intercal, \phi_{\sigma_1}^\intercal, \ldots, \phi_{\sigma_{n_q}}^\intercal]^\intercal$ for each $\sigma_1, \ldots, \sigma_{n_q}$ that labels the outgoing transitions from state q to state $q' \neq q$. For example, in state 0 where the outgoing transitions are labeled A, C, obs, we include $\phi_0 = [\phi_{basic}^\intercal, \phi_A^\intercal, \phi_C^\intercal, \phi_{obs}^\intercal]^\intercal$ as a set of linear bases. Similar choice of bases is performed for other specification states. The total number of weights is 78.

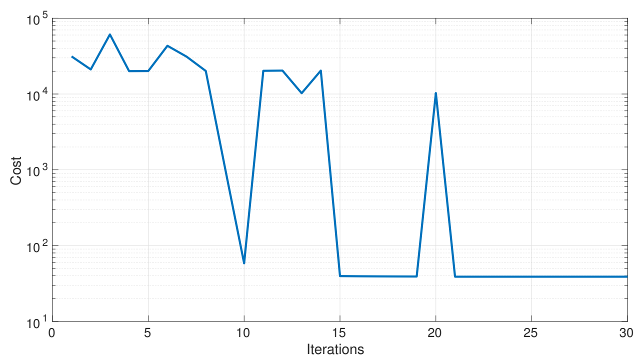
The following parameters are used: Initial sample size $N_0 = 1000$, improvement parameter $\epsilon = 0.1$, smoothing parameter $\lambda = 0.2$, sample increment percentage $\alpha = 0.1$, the estimated $(1-\rho)$-quantile of cost $\kappa = 1000$, and quantile parameter $\rho = 0.1$. Figure 2(a) trajectory is obtained after the algorithm converges, which satisfies the LTL formula.

Figure 2(b) shows the performance of obtained controllers during each iteration. The algorithm obtains a satisfying trajectory after 10 iterations within 2 minutes. It takes around 20 iterations to converge, and each iteration takes around 10 seconds. Out of 100 repeated runs of the algorithm, the mean of the optimal cost obtained with the controllers is 42.01 and the standard deviation is 2.77.

[4]The temporal operators are: $\Diamond$ - eventuality, $\Box$ - always.



(a)



(b)

Fig. 2. (a) The trajectory for the linear system calculated by the computed controller after the algorithm converges. (b) The mean cost achieved over iterations.

B. Case study: Approximate optimal control of a Dubins car under LTL constraints

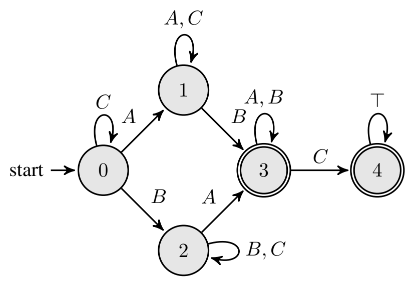
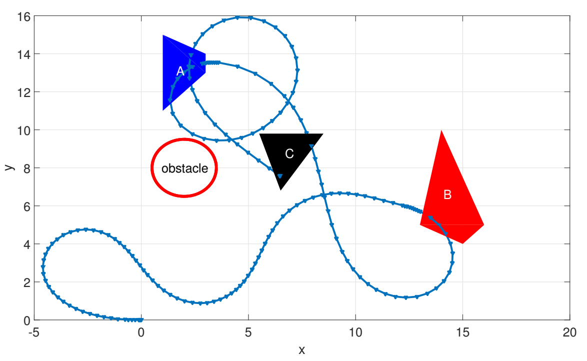
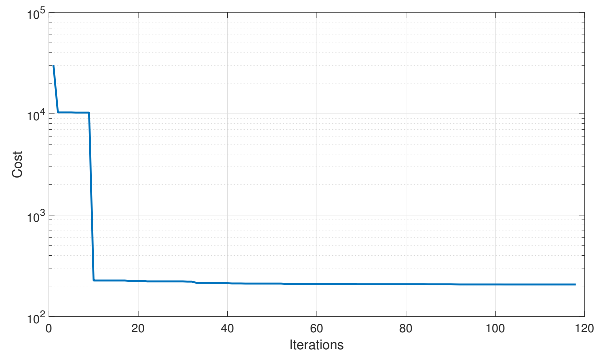


Fig. 3. Automaton $\mathcal{A}_{\varphi_2}$ for $\Diamond(\Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A) \wedge \Diamond C) \wedge \Box\neg\text{obs}$ where $\top$ is universally true. The self-loops with labels other than A, B, C, obs are omitted.

Given the Dubins car's Dynamics: $\dot{x} = u\cos(\theta), \dot{y} = u\sin(\theta), \dot{\theta} = v$, where $x = (x, y, \theta) \in \mathbb{R}^2 \times \mathbb{S}^1$ is the state and $u \in [-10, 10], v \in [-5, 5]$ are the inputs to the system and linear and angular velocities. We consider a LTL formula: $\Diamond(\Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A) \wedge \Diamond C) \wedge \Box\neg\text{obs}$. The corresponding specification automaton is shown in Fig. 3. In this case, the system needs to traverse regions A and B, and eventually reach region C while avoiding collisions with a static obstacle. The running cost function is defined by $x^2 + y^2 + \|u\|_Q$ for the same Q matrix as in the previous example. The terminal cost is 0. We define a rank cost:



(a)



(b)

Fig. 4. (a) The trajectory for Dubins car calculated with the computed controller after the algorithm converges. (b) The optimal cost achieved over iterations.

$h(q) = \text{rank}(q) \times 10^4$, where $\text{rank}(q)$ is the minimum number of transitions required to reach accepting states from $q(T)$.

We switch different basis functions based on the current state. In Dubins car's case, we select RBFs and define $\phi_{rbf} = [\phi_1, \ldots, \phi_N]$ where ϕ_i is a RBF with its center being one of the points picked from uniform grids in x-y coordinates with step sizes $\delta x = 5$ and $\delta y = 5$ with bounded state space $-5 \leq x \leq 20$ and $-5 \leq y \leq 20$. The σ parameter of each RBF is 2. Combining these points with the linear basis function $\phi_{linear} = [x - x_f]$, the basic basis vector can be represented as $\phi_{basic} = [\phi_{rbf}^\intercal, \phi_{linear}^\intercal]^\intercal$. Additional RBF bases are determined by picking their centers as the vertexes of regions A, B, and C. For example, in the state 2, since the region B has been visited, the vertexes of region A are taken into the consideration, so the basis function with the respect to the state 2 is described as the following: $\phi_2 = [\phi_{basic}^\intercal, \phi_A^\intercal]^\intercal$ where ϕ_A is the vector of RBFs with centers on the vertexes of A and a RBF parameter σ. The dimension of the weight vector is 179.

Let $N_0 = 200$ [5], the other parameters are the same as the ones for the linear system. The trajectory shown in the Figure 4(a) is the result of 118 iterations, each iteration takes

[5]Due to the adaptive sampling, MRAS increases the sample size if less elite samples are generated. We pick a small initial sample size to reduce time for each iteration. Considerably, the total time of iterations increases.

14 seconds. The optimal cost under such LTL constraints is 207.13. Although it takes a large number of iterations for the algorithm to converge and it computes a satisfying trajectory within 20 iterations, this indicates the potential of using this algorithm for anytime planning.

V. Conclusion

This paper presents a novel sampling-based planning method for nonlinear systems under temporal logic constraints. The optimal motion plan is given as a feedback policy that takes into consideration of the specification satisfied at runtime. Such a policy is approximated by a linear combination of basis functions whose weights are solved through importance sampling-based global optimization method. The method is probabilistically complete and has empirical good result for being able to generate a feasible trajectory and keep improving the solution when more time is allowed. Besides the anytime computation feature, it is noted that the algorithm is highly parallelizable and can be improved significantly in GPU-computing enabled systems.

For temporal logic constraints, we introduce the rank function to improve the sample efficiency without rejecting unsatisfied trajectories. The current limitation is that for complex specifications, the dimensionality of weight vectors grows polynomials in the size of the automaton and high-dimensional weight vectors requires a large sample size. For future work, we will exam different sample distributions and distributed synthesis method to improve the scalability of the algorithm. Additionally, in our future extension of this work, we would like to focus on the integration of the selection of initial guess and this planning algorithm, which helps to accelerate the speed of the convergence.

References

[1] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[3] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[4] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *IEEE International Conference on Robotics and Automation*, pages 625–632. IEEE, 2009.

[5] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[6] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[7] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[8] Luc C. G. J. M. Habets and Calin Belta. Temporal logic control for piecewise-affine hybrid systems on polytopes. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pages 195–202, July 2010.

[9] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[10] Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation*, 1(3):111–128, 2011.

[11] Jiaqiao Hu, Michael C Fu, and Steven I Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55(3):549–568, 2007.

[12] Sertac Karaman and Emilio Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 2222–2229. IEEE, 2009.

[13] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2010.

[14] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[15] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[16] Marin Kobilarov. Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871, 2012.

[17] Marin Kobilarov. Sample complexity bounds for iterative stochastic policy optimization. In *Advances in Neural Information Processing Systems*, pages 3114–3122, 2015.

[18] Orna Kupferman and Moshe Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, November 2001.

[19] Orna Kupferman and MosheY Vardi. Model checking of safety properties. In *International Conference on Computer Aided Verification*, pages 172–183. Springer, 1999.

[20] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[21] Jun Liu and Necmiye Ozay. Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *International Conference on Hybrid Systems: Computation and Control*, pages 293–302. ACM, 2014.

[22] Scott C Livingston, Eric M Wolff, and Richard M Murray. Cross-entropy temporal logic motion planning. In *International Conference on Hybrid Systems: Computation and Control*, pages 269–278. ACM, 2015.

[23] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[24] Denis Oddoux and Paul Gastin. LTL 2 BA : fast translation from LTL formulae to Büchi automata. `http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/`, 2008. [Online; accessed 13-Sep-2016].

[25] Ivan Papusha, Jie Fu, Ufuk Topcu, and Richard M. Murray. Automata theory meets approximate dynamic programming: Optimal control with temporal logic constraints. In *IEEE Conference on Decision and Control*, Dec 2016, accepted.

[26] Stephen L. Smith, Jana Tůmova, Calin Belta, and Daniela Rus. Optimal path planning for surveilance with temporal-logic constraints. *International Journal of Robotics Research*, 30(14):1695–1708, December 2011.

[27] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE, 1994.

[28] Vladimir Temlyakov. *Greedy approximation*, volume 20. Cambridge University Press, 2011.

[29] Cristian Ioan Vasile and Calin Belta. Sampling-based temporal logic path planning. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4817–4822. IEEE, 2013.

[30] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, November 2012.

[31] Huizhen Yu and Dimitri P Bertsekas. Basis function adaptation methods for cost approximation in mdp. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pages 74–81, 2009.