Sampling-based Approximate Optimal Control Under Temporal Logic Constraints

Jie Fu
Department of Electrical and Computer Engineering
Robotics Engineering Program
Worcester Polytechnic Institute
Worcester, MA, 01604
jfu2@wpi.edu

Ivan Papusha
Institute for Computational Engineering and Sciences
University of Texas at Austin
Austin, TX, USA
ipapusha@utexas.edu

Ufuk Topcu
Aerospace Engineering and Engineering Mechanics
University of Texas at Austin
Austin, TX, USA
utopcu@utexas.edu

ABSTRACT

We investigate a sampling-based method for optimal control of continuous-time and continuous-state (possibly nonlinear) systems under co-safe linear temporal logic specifications. We express the temporal logic specification as a deterministic, finite automaton (the specification automaton), and link the automaton's discrete transitions to the continuous system state as it passes through specified regions. The optimal hybrid controller is characterized by a set of coupled partial differential equations. Because these equations are difficult to solve exactly in practice in all cases, we propose instead a sampling based technique to solve for an approximate controller through approximate value iteration. We adopt model reference adaptive search—an importance sampling optimization algorithm—to determine the mixing weights of the approximate value function expressed in a finite basis. Under mild technical assumptions, the algorithm converges, with probability one, to an optimal weight that ensures the satisfaction of temporal logic constraints, while minimizing an upper bound for the optimal cost. We demonstrate the correctness and efficiency of the method through numerical experiments, including temporal logic planning for a linear system, and a nonlinear mobile robot.

Keywords

Approximate optimal control, formal methods, importance sampling, hybrid systems.

1. INTRODUCTION

In this work, we propose a novel sampling-based optimal control method for continuous-time and continuous-state nonlinear systems subject to a subclass of temporal logic constraints, i.e., co-safe Linear Temporal Logic (LTL) [15]. Co-safe LTL formulas are LTL formulas with satisfying traces that can be recognized by a deterministic finite automaton. Co-safe LTL is an expressive formal language that allows one to specify a variety of finite-time behaviors, including traditional reaching-a-goal, stability, obstacle avoidance, sequentially visiting regions of interest, and conditional reactive behaviors [21].

Typically, trajectory generation and control of continuous systems with formal specifications is performed using abstraction-based synthesis, for example, by computing a discrete transition system that abstracts or simulates the system dynamics, and then planning in the discrete state space. Abstraction-based synthesis methods have been studied extensively for continuous linear and nonlinear systems [12, 3, 5, 18, 2, 24, 27].

However, abstraction-based synthesis has several important limitations. Among these, we highlight three: first, because they rely on discretization, abstraction-based methods scale poorly with the dimension of the continuous state space; second, optimality is no longer guaranteed when a controller is synthesized with the discrete abstracted system; and third, depending on the specific abstraction method, it is possible that a control policy exists for the underlying continuous control system even when one does not appear to exist in the abstraction.

To address these concerns, the work [20] exploited the idea that continuous-time and continuous-state systems constrained by co-safe LTL specifications can be viewed as hybrid dynamical systems. The continuous state space is augmented with the discrete states of the specification automaton, derived from the co-safe LTL formula. Then, a hybrid feedback controller is obtained by solving or approximately solving the corresponding Hamilton–Jacobi–Bellman (HJB) equations for the hybrid value function. For linear systems, the approximate hybrid value function can be obtained by semidefinite programming. But for general nonlinear systems, the resulting optimization problem is semi-infinite [22], and difficult to solve.

We tackle the computational difficulty of this semi-infinite program in the optimal control synthesis for this class of hybrid systems by developing an importance sampling algorithm. The key idea is to regard the control problem as a problem of inference, from sampled trajectories, of a weight vector that defines an approximate value function in a given basis. For sample-efficient search through the weight parameter space, we use model reference adaptive search (MRAS) [9]. In our MRAS-inspired algorithm, specific policies are computed from the value function approximations. These weights are then ranked by the performance of their corre-

HSCC'17, April 18-20, 2017, Pittsburgh, PA, USA
© 2017 ACM. ISBN 978-1-4503-4590-3/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3049797.3049820

sponding controllers, and the best performing weights (elite samples) are used to improve the weight sampling distribution. The aim is to find a value function approximation that minimizes an upper bound on the total cost. Similar cross-entropy (CE) methods have been used for trajectory planning in the past [13, 19], with the goal of finding a sequence of motion primitives or a sequence of states for interpolation-based planning. In stochastic policy optimization [14], CE is also useful for computing linear feedback policies.

Under certain regularity assumptions, we show that our sampling method is probabilistically complete, i.e., it converges to the optimal value function approximation in a given basis as the sample size goes to infinity. To improve the sampling efficiency, we employ rank functions in the specification to guide the iterative update of the sampling distribution. Based on experimental studies, we show that our method can generate an initial feasible hybrid controller that satisfies the co-safe LTL specification, which is then continually improved as more computation time is permitted. However, when a limited number of samples is used for each iteration, the method may converge to local optima. In the last section, we discuss future extensions to address these problems and conclude the work.

2. PROBLEM DESCRIPTION

We consider a continuous-time and continuous-state dynamical system on $\mathbb{R}^n$. This system is given by

$$\dot{x} = f(x, u), \quad x(0) = x_0, \tag{1}$$

where $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ are the state and control signals at time t. For simplicity, we restrict f to be a Lipschitz continuous function of (x, u), and the control input u to be a piecewise right-continuous function of time, with finitely many discontinuities on any finite time interval. These conditions ensure the existence and uniqueness of solutions, and are meant to prevent Zeno behavior.

2.1 Co-safe Linear 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 finite set of atomic propositions, which are logical predicates that hold true when $x(t)$ is in a particular region of the state space $\mathcal{X}$. Then, define a labeling function $L : \mathcal{X} \to \Sigma$, which maps a continuous state $x \in \mathcal{X}$ to a finite set $\Sigma = 2^{\mathcal{AP}}$ of atomic propositions that evaluate to true at x. This function partitions the continuous space $\mathcal{X}$ into regions that share the same truth values in $\mathcal{AP}$. The labeling function also links the continuous system with its *discrete behavior*. In the following definition, $\phi(x_0, [0, T], u)$ refers to the trajectory of the continuous system with initial condition x_0 under the control input $u(t)$ over the time interval $[0, T]$.

Definition 1. *Let $t_0, t_1, \ldots, t_N$ be times, such that*

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

The discrete behavior*, denoted $\mathcal{B}(\phi(x_0, [0, T], u))$, is the discrete word $\sigma_0\sigma_1 \ldots \sigma_{N-1} \in \Sigma^*$, where $\sigma_k = L(x(t_k))$.*

In the scope of this paper, we consider a subclass of linear temporal logic (LTL) specifications called co-safe LTL. A co-safe LTL formula is an LTL formula such that every satisfying word has a *finite* good prefix[1] [15]. This subclass of LTL formulas can be used to express tasks that can be completed in a finite time horizon. Given a co-safe LTL specification φ over the set of atomic propositions $\mathcal{AP}$, there exists a corresponding deterministic finite-state automaton (DFA) $\mathcal{A}_\varphi = \langle Q, \Sigma, \delta, q_0, F \rangle$, where Q is a finite set of states (modes), $\Sigma = 2^{\mathcal{AP}}$ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is a *deterministic* transition function such that when the symbol $\sigma \in \Sigma$ is read at state q, the automaton makes a deterministic transition to state $\delta(q, \sigma) = q'$, $q_0 \in Q$ 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* $w = \sigma_0\sigma_1 \ldots \in \Sigma^*$, 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 $\mathcal{L}(\mathcal{A}_\varphi)$.

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 evaluation of an atomic proposition changes value, indicated by the sequence of discrete time-stamps $t_0, t_1, \ldots, t_N$. Thus a trajectory $\phi(x_0, [0, T], u)$ satisfies an LTL specification φ if and only if its discrete behavior is in the language $\mathcal{L}(\mathcal{A}_\varphi)$. The optimal control problem is formulated as follows.

Problem 1. *Consider the system (1), a co-safe LTL specification φ, and a final state $x_f \in \mathcal{X}$. Design a control law u that minimizes the cost function*

$$\begin{aligned} J(x_0, u) = &\int_0^T \ell(x(\tau), u(\tau))\, d\tau \\ &+ \sum_{k=0}^{N} s(x(t_k), q(t_k^-), q(t_k^+)) \end{aligned} \tag{2}$$

subject to the constraints that $\mathcal{B}(\phi(x_0, [0, T], u)) \in \mathcal{L}(\mathcal{A}_\varphi)$ and $x(T) = x_f$.

Here, $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is a continuous loss function, and $s : \mathcal{X} \times Q \times Q \to \mathbb{R}$ is the cost to transition between two states of the automaton whenever such a transition is allowed. The final state $x(T) = x_f$ is also specified. Similar problems have been studied in prior work [7, 8, 28, 5, 27, 26, 11]. Recently, the work [20] showed that determining an optimal controller for continuous-time systems under co-safe LTL behavior constraints can be translated into an optimal hybrid control problem. For linear and polynomial systems, an approximate optimal controller can be obtained by a special formulation of semi-infinite programming problems. The novelty in this paper is the development of a sampling-based algorithm that handles both nonlinear dynamical constraints, and co-safe LTL specifications. The algorithm is based on MRAS, which is briefly described next.

2.2 Model Reference Adaptive Search (MRAS)

[1]For two given words $v, w \in \Sigma^*$, the word 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.

MRAS [9] is a general sampling-based method for global optimization that aims to solve the following problem:

$$z^\star = \operatorname*{argmax}_{z \in Z} H(z),$$

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

The key idea of MRAS is similar to cross-entropy (CE) optimization methods. First, we sample the solution space Z with a parameterized distribution. Then, we select samples with the highest objective values, termed "elite samples," and update the parameters of the sampling distribution to assign a higher probability mass to the elite samples. Assuming the neighborhood of the optimal solution $z^\star$ has a positive probability of being sampled, and H satisfies certain regularity conditions [9, §2], the parametrized distribution converges to a distribution concentrated around $z^\star$. The MRAS algorithm consists of the following key steps:

- Define a sequence of reference distributions $\{g_k(\cdot)\}$ that converges to a distribution centered on the optimal solution $z^\star$, for example,

  $$g_k(z) = \frac{H(z) g_{k-1}(z)}{\int_Z H(z') g_{k-1}(z') \nu(dz')}, \quad k = 1, 2, \ldots,$$

  where $\nu(\cdot)$ is the Lebesgue measure defined over Z.

- Define a parameterized family of distributions $\{p(\cdot, \theta) \mid \theta \in \Theta\}$ over Z, onto which the exact reference distributions $\{g_k(\cdot)\}$ will be projected.

- Generate a sequence of parameters $\{\theta_k\}$ by minimizing (over $\theta \in \Theta$) the Kullback–Leibler (KL) divergence between $g_k(\cdot)$ and $p(\cdot, \theta)$,

  $$D_{KL}(g_k, p(\cdot, \theta)) := \int_Z \ln \frac{g_k(z)}{p(z, \theta)} g_k(z) \nu(dz),$$

  at each step $k = 1, 2, \ldots$

The sample distributions $\{p(\cdot, \theta_k)\}$ are meant to be compact approximations of the reference distributions $\{g_k(\cdot)\}$ that converge to the same optimal solution. Note that the reference distributions $\{g_k(\cdot)\}$ are unknown beforehand, as the optimal solution is unknown. In order to represent the reference distributions $\{g_k(\cdot)\}$, MRAS uses estimation of distributions [6] from elite samples (similar to CE). Then, the MRAS algorithm computes the parameter that minimizes the KL divergence between the sampling distribution and the reference distribution. As shown in [9], MRAS has better convergence rates and stronger guarantees than CE over several benchmark examples.

3. HYBRID SYSTEM FORMULATION OF CONTROL SYSTEMS UNDER TEMPORAL LOGIC CONSTRAINTS

To apply MRAS in solving Problem 1, we follow the setting and notation of [20] to define a hybrid system from the control system dynamics and the temporal logic constraints. The state space of the hybrid system is the product of the continuous state space $\mathcal{X}$ and the discrete state space Q of the specification automaton $\mathcal{A}_\varphi$, which is obtained from the co-safe LTL specification φ with existing tools [4, 16]. This produces a hybrid system where each mode is governed by the same continuous vector field (1). Switching between different discrete modes occurs when the continuous state crosses a boundary between two labeled regions. Specifically, we consider the following product hybrid system:

Definition 2. *The* product system $\mathcal{H} = \langle Q, \mathcal{X}, \Sigma, E, f, R, G \rangle$ *is an internally forced hybrid system, where*

- Q *is the set of discrete states (modes) of* $\mathcal{A}_\varphi$,
- $\mathcal{X} \subseteq \mathbb{R}^n$ *is the set of continuous states,*
- $\Sigma = 2^{\mathcal{AP}}$ *is the power set of atomic propositions,*
- $E \subseteq Q \times \Sigma \times Q$ *is a set of discrete transitions, where* $e = (q, \sigma, q') \in E$ *if and only if* $\delta(q, \sigma) = q'$,
- $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ *is the continuous vector field given by* (1),
- $R = \{R_q \mid q \in Q\}$ *is a collection of regions, where*

  $$R_{q,\sigma} = \{x \in \mathcal{X} \mid (q, \sigma, q) \in E, \text{ and } L(x) = \sigma\}, \quad q \in Q, \sigma \in \Sigma,$$

  $$R_q = \bigcup_{\sigma \in \Sigma} R_{q,\sigma}, \quad q \in Q,$$

- $G = \{G_e \mid e \in E\}$ *is a collection of guards, where*

  $$G_e = \{x \in \partial R_{q,\sigma} \mid \delta(q, L(x)) = q'\},$$

  for each $e = (q, \sigma, q') \in E$.

Each invariant region R_q refers to the continuous states $x \in \mathcal{X}$ that are reachable while the automaton is in mode q. For each discrete mode q, the continuous state evolves inside R_q until it enters a guard region $G_{(q,\sigma,q')}$ and a discrete transition to mode q' is made.

We can solve the optimal control problem with dynamic programming by ensuring that the optimal value function is zero at every accepting state of the automaton. Let $V^\star : \mathcal{X} \times Q \to \mathbb{R}$ be the optimal cost-to-go in (2), with $V^\star(x_0, q_0)$ denoting the optimal objective value when starting at initial condition (x_0, q_0), subject to the discrete behavior specification and final condition $x(T) = x_f$ for a free terminal time T. The existance of the controller is assumed.

In this setting, the cost-to-go satisfies a collection of mixed continuous-discrete Hamilton–Jacobi–Bellman (HJB) equations,

$$0 = \min_{u \in \mathcal{U}} \left\{ \frac{\partial V^\star(x, q)}{\partial x} \cdot f(x, u) + \ell(x, u) \right\}, \quad \forall x \in R_q,\ \forall q \in Q, \tag{3}$$

$$V^\star(x, q) = \min_{q'} \left\{ V^\star(x, q') + s(x, q, q') \right\}, \quad \forall x \in G_e,\ \forall e = (q, \sigma, q') \in E, \tag{4}$$

$$0 = V^\star(x_f, q_f), \quad \forall q_f \in F. \tag{5}$$

Equation (3) says that $V^\star(x, q)$ is an optimal cost-to-go inside the regions where the label remains constant. The next equation (4) is a shortest-path equality that must hold at every continuous state x where discrete state transition to a different label can happen. Finally, the boundary equation (5) fixes the value function.

4. APPROXIMATE OPTIMAL CONTROL UNDER TEMPORAL LOGIC CONSTRAINTS

In this section, we show that by approximating the optimal value function $V^\star(x,q)$ at each state $q \in Q$ with a linear combination $\hat{V}(x,q)$ of pre-defined basis functions, we can formulate a semi-infinite programming problem over the weight parameters. This value function candidate minimizes an upper bound of the optimal total cost.

Assumption 1. *For each mode $q \in Q$, the optimal value function $V^\star(\cdot,q)$ is C^1-differentiable. Moreover, the optimal value function $V^\star(\cdot,q)$ is positive semidefinite. This is guaranteed by the condition $\ell(x,u) > 0$, for all $x \in \mathcal{X} \setminus \{x_f\}$ and $u \in \mathcal{U}$, and $s(x,q,q') > 0$, for all $x \in \mathcal{X}$, $q,q' \in Q$.*

Recall from the Weierstrass higher-order approximation theorem [25] that there exists a complete independent set of bases $\{\phi_i(x,q) \mid i = 1,\dots,N_q\}$, such that the function $V^\star(x,q)$ is uniformly approximated as

$$V^\star(x,q) = \sum_{i=1}^{N_q} w_{i,q}\phi_{i,q}(x) + \sum_{i=N_q+1}^{\infty} w_{i,q}\phi_{i,q}(x), \text{ and}$$

$$\frac{\partial V^\star(x,q)}{\partial x} = \sum_{i=1}^{N_q} w_{i,q}\frac{\partial \phi_{i,q}(x)}{\partial x} + \sum_{i=N_q+1}^{\infty} w_{i,q}\frac{\partial \phi_{i,q}(x)}{\partial x},$$

for some weights $\{w_{i,q}\}$, where in both expressions, the last term converges uniformly to zero as $N_q \to \infty$.

Thus, it is justified to assume that for any state q, the optimal value function approximation $\hat{V}(x,q)$ in a given set $\{\phi_{i,q}(x)\}$ of bases is

$$\hat{V}(x,q) = \sum_{i=1}^{N_q} w_{i,q}\phi_{i,q}(x).$$

We write $\hat{V}(x,q)$ compactly as $\vec{w}_q^T \vec{\phi}_q(x)$ where

$$\vec{w}_q = [w_{1,q},\dots,w_{N_q,q}]^T, \quad \vec{\phi}_q(x) = [\phi_{1,q}(x),\dots,\phi_{N_q,q}(x)]^T.$$

With a slight abuse of notation, the overall piecewise continuous value function approximation is denoted $\hat{V} = \langle W, \Phi \rangle$ with $W = [\vec{w}_q]_{q\in Q}$ and $\Phi = [\vec{\phi}_q]_{q\in Q}$ and $\hat{V}(x,q) = \vec{w}_q^T\vec{\phi}_q(x)$, for all $x \in R_q$.

Given an approximate value function $\hat{V}$, we define a hybrid feedback control law $u : \mathcal{X} \times Q \to \mathcal{U}$ as

$$u(x,q) = \operatorname*{argmin}_{a\in\mathcal{U}} \left\{ \frac{\partial \hat{V}(x,q)}{\partial x} \cdot f(x,a) + \ell(x,a) \right\}. \tag{6}$$

For simulation-based optimization, we minimize an upper bound for the optimal total cost, and define the optimal value function approximation as follows.

Definition 3. *The* projected optimal value function approximation *is given by an optimal $W^\star \in \mathcal{W}$ that solves*

$$\begin{aligned}
\mathop{\mathit{minimize}}_{W\in\mathcal{W}} \quad & J(x_0,u) + \bar{J} \times (1 - 1_F(q_f)) && \text{(7a)}\\
\mathit{subject\ to} \quad & \hat{V} = \langle W, \Phi \rangle && \text{(7b)}\\
& u(x,q) = \operatorname*{argmin}_{a\in\mathcal{U}} \left\{ \frac{\partial \hat{V}(x,q)}{\partial x} \cdot f(x,a) + \ell(x,a) \right\}, \\
& \qquad \forall x \in R_q,\ \forall q \in Q, && \text{(7c)}\\
& \hat{V}(x,q) = \min_{q'} \{\hat{V}(x,q') + s(x,q,q')\}, \\
& \qquad \forall x \in G_e,\ \forall e = (q,\sigma,q') \in E, && \text{(7d)}\\
& \hat{V}(x_f,q_f) = 0,\ \forall q_f \in F && \text{(7e)}\\
& \dot{x} = f(x,u),\ x(0) = x_0, \\
& q(t_k^+) = \delta(q(t_k^-), L(x(t_k^+))), \\
& \qquad \forall t_k \ \mathit{such\ that}\ L(x(t_k^-)) \neq L(x(t_k^+)), && \text{(7f)}
\end{aligned}$$

where $\bar{J} \in \mathbb{R}_+$ is a pre-defined large penalty for violating the specification.

The choice of $\bar{J}$ will be discussed in more detail in Section 4.2. Note that the decision variable W is implicit in the objective function and explicit in the controller u, on which the cost function is dependent.

We only consider value function approximations of the form $\hat{V} = \langle W, \Phi \rangle$ for a given set of bases Φ. For each value function approximation, the corresponding controller is fixed under constraint (7c). The optimal value function approximation is the one that minimizes the the actual cost $J(x_0,u)$ under that fixed controller within the constrained policy space. Slightly abusing notation, let us denote by $J(x_0;W)$ the cost generated by applying the controller u computed from the value function approximation $\langle W, \Phi \rangle$, and by $\hat{V}(x_0;W)$ the value function approximation $\langle W, \Phi \rangle$ evaluated at the initial state x_0.

4.1 Sampling-based control design

Previously, variants of gradient descent [17, 1] have been used to implement approximate value iteration. Local optima can be problematic when there is discontinuity in the value function for hybrid systems. Besides, finding a reasonable starting point within the weight space is also critical for the successful application of the gradient descent. We instead employ MRAS to solve (7). The idea is to sample the weight space $\mathcal{W}$ according to a parameterized distribution $p(\cdot,\theta)$ for parameter $\theta \in \Theta$. For each sampled weight, we simulate the run of the corresponding optimal controller using a model of the system for a finite time T. The finite time T is an estimated upper bound on the time it takes for the system to stabilize to x_f with a final bounded error, for any controller with which the closed-loop system satisfies the specification. By evaluating the sampled trajectory, we update θ to bias the sampling distribution towards promising weight parameters. With probability one, the distribution will converge to a parameter $\theta^\star$, and the distribution $p(\cdot,\theta^\star)$ concentrates on the solution $W^\star$ to the optimization problem (7).

First, we select a multivariate Gaussian distribution as the sample distribution. Recall that the probability density of a multivariate Gaussian distribution is given by

$$p(W;\theta) = \frac{1}{\sqrt{(2\pi)^N|\Sigma|}} \exp(-\frac{1}{2}(W-\mu)^T\Sigma^{-1}(W-\mu)),$$
$$\theta = (\mu, \Sigma), \quad \forall W \in \mathcal{W},$$

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

Next, we iteratively update the mean and covariance of the sampling distribution until $p(\cdot, \theta_k)$ converges to a degenerate distribution with a vanishing covariance.

1) **Initialization**: Select an initial distribution over weight vectors $\mathcal{W}$, denoted $p(\cdot, \theta_0)$, for some $\theta_0 \in \Theta$. We specify a parameter $\rho \in (0, 1]$, a small real $\varepsilon \in \mathbb{R}_+$ called an *improvement parameter*, an initial sample size N_0, a *smoothing coefficient* $\alpha \in (0, 1]$, a strictly decreasing and positive function $S : \mathbb{R} \to \mathbb{R}_+$. Possible choices can be $S(x) = e^{-x}$ or $S(x) = \frac{1}{x}$ for x strictly positive (which is the case here because the total cost is always positive). Set $k := 1$ and go to step 2).

2) **Sampling**: At each iteration k, given the current distribution $p(\cdot, \theta_k)$, generate a set $\mathcal{SW}$ of N_k samples. For each $W \in \mathcal{SW}$ in the sample set, evaluate $J(x_0; W)$ by simulating the system model (1) with the approximate controller (6), where $\hat{V} = \langle W, \Phi\rangle$.

3) **Reject unsatisfiable policies**: Reject all weights for which the generated controllers do not satisfy the LTL formula. The remaining set of weights is denoted $\mathcal{SW}_{\mathsf{sat}}$.

4) **Select elite samples and threshold**: Order the set $\{J(x_0; W) \mid W \in \mathcal{SW}_{\mathsf{sat}}\}$ from largest (worst) to smallest (best) among the given samples,

$$J_{k,(0)} \geq \ldots \geq J_{k,(N_k)}.$$

Let κ be the estimated $(1-\rho)$-quantile of costs $J(\cdot; W)$, i.e., $\kappa = J_{k,\lceil(1-\rho)N_k\rceil}$.

   - If $k = 0$, we introduce a threshold $\gamma := \kappa$.
   - If $k \neq 0$, the following cases are further distinguished:
     - If $\kappa \leq \gamma - \varepsilon$, i.e., the estimated $(1-\rho)$-quantile of cost has been reduced by the amount ε from the last iteration, then update $\gamma := \kappa$, $N_{k+1} := N_k$, and go to step 5).
     - Otherwise, if $\kappa > \gamma - \varepsilon$, find the largest ρ', if it exists, such that the estimated $(1-\rho')$-quantile of cost $\kappa' = J_{k,\lceil(1-\rho')N_k\rceil}$ satisfies $\kappa' \leq \gamma - \varepsilon$. Then, update $\gamma := \kappa'$, and also the quantile $\rho := \rho'$. Set $N_{k+1} := N_0$ and go to step 5). If no such ρ' exists, increase the sample size by a factor of $(1 + \alpha)$, $N_{k+1} = \lceil(1 + \alpha)N_k\rceil$. Set $\theta_{k+1} := \theta_k$, $k := k + 1$, and go to step 2).

5) **Parameter update**: Update parameters θ_{k+1} for iteration $k + 1$ as follows. First, define a set $\mathcal{EW} = \{W \mid J(x_0; W) \leq \gamma, W \in \mathcal{SW}_{\mathsf{sat}}\}$ of *elite samples*. The samples in $\mathcal{EW}$ are *accepted* and samples not in $\mathcal{EW}$ are *rejected* during the parameter update defined next. Select the next parameter θ_{k+1} to maximize the weighted sum of probabilities of elite samples according to a weighting

$$\theta^\star_{k+1} = \operatorname*{argmax}_{\theta\in\Theta} \sum_{W\in\mathcal{EW}} \frac{S(J(x_0;W))^k}{p(W,\theta_k)} p(W,\theta) \tag{8}$$

that puts a higher weight on those weight vectors with the lowest costs. Since the optimal parameter $\theta^\star_{k+1}$ cannot be determined analytically, we use maximum likelihood estimate of the elite sample distribution $\theta^\star_{k+1} \approx (\mu_{k+1}, \Sigma_{k+1})$, where

$$\begin{aligned}\mu_{k+1} &= \frac{\mathbb{E}_{\theta_k}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)\cdot W}{\mathbb{E}_{\theta_k}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)} \\ &\approx \frac{\sum_{W\in\mathcal{SW}}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)\cdot W}{\sum_{W\in\mathcal{SW}}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)},\end{aligned} \tag{9}$$

and

$$\begin{aligned}\Sigma_{k+1} &= \frac{\mathbb{E}_{\theta_k}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)\cdot(W-\mu)(W-\mu)^T}{\mathbb{E}_{\theta_k}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)} \\ &\approx \frac{\sum_{W\in\mathcal{SW}}\left(\frac{S(J(x_0,W))^k}{p(W,\theta_k)}\right) I_{\mathcal{EW}}(W)\cdot(W-\mu)(W-\mu)^T}{\sum_{W\in\mathcal{SW}}\frac{S(J(x_0;W))^k}{p(W,\theta_k)} I_{\mathcal{EW}}(W)}.\end{aligned} \tag{10}$$

Note that we approximate $\mathbb{E}_{\theta_k} h(r)$ with its sample estimate, $\frac{1}{N_k}\sum_{W\in\mathcal{SW}} h(W)$ for $r \sim p(\cdot, \theta_k)$, and $I_{\mathcal{EW}} : W \to \{0, 1\}$ is the indicator function that equals 1 if $W \in \mathcal{EW}$ and 0 otherwise.

6) **Smoothing update:** The actual parameter is smoothed with parameter $\lambda \in (0, 1)$ as

$$\theta_{k+1} := \lambda\theta_k + (1-\lambda)\theta^\star_{k+1}. \tag{11}$$

7) **Stopping criterion:** Stop the iteration if the covariance matrix Σ_k becomes near singular, that is, the determinant of Σ_k approaches 0. The reason for this choice of stopping criterion is given next.

In the algorithm, elite samples can be reused by including in the current sample $\mathcal{SW}$ at iteration k, denoted $\mathcal{SW}_k$, the set of elite samples from the previous iteration $\mathcal{EW}$, denoted $\mathcal{EW}_{k-1}$ for $k > 1$. The current set of elite samples is obtained from the ρ-quantile of $\mathcal{SW}_k \cup \mathcal{EW}_{k-1}$.

We show that the global convergence of the algorithm is ensured by the properties of MRAS under the following additional assumptions. Let $W^\star$ be the optimal weight parameterizing the projected optimal value function approximation.

Assumption 2. *For any given constant $\xi > J(x_0; W^\star)$, the set $\{W \mid J(x_0; W) \leq \xi\} \cap \mathcal{W}$ has a strictly positive Lebesgue measure.*

This condition ensures that any neighborhood of the optimal solution $W^\star$ will be sampled with a positive probability.

Assumption 3. *For any $\delta > 0$, we have $\inf_{W\in\mathcal{W}_\delta} J(x_0; W) > J(x_0; W^\star)$, where $W_\delta := \{W \mid \|W - W^\star\| \geq \delta\} \cap \mathcal{W}$.*

Theorem 1 (Adapted from Theorem 1 [9])**.** *Under assumptions 1–3, the algorithm converges (w.p.1) to*

$$\lim_{k\to\infty} \mu_k = W^\star \text{ and } \lim_{k\to\infty} \Sigma_k = 0_{n\times n},$$

provided that (8) *is solved exactly.*

Since our algorithm directly uses the multivariate Gaussian distribution, it would converge to the global optimal solution after finitely many iterations if it could be provided with an infinite number of samples. However, in practice, only finitely many samples can be used. In addition, the parameter update (8) is not solved exactly based on the entire set of elite samples. Instead, we provide a maximum likelihood estimation of θ for the next iteration using a finite number of elite samples. Similar to the CE method, the resulting algorithm may converge to local optimal solutions if the sample size is not sufficient. In general, the number of samples for each iteration is polynomial in the number of the decision variables [23], which, in this context, is the number of unknown weights.

It should be noted that the value function approximation computed with this sampling-based algorithm is not necessarily a control Lyapunov function. In fact, there is no guarantee ahead of time that the space of value function approximations in a given fixed bases contains a control Lyapunov function. In this case, this method performs approximate optimal feedback planning in the hybrid system instead of control design that stabilizes the system.

4.2 Rank-guided sample weighting

A direct implementation of the algorithm should enable the convergence to a global optimal weight given a chosen bases. However, for a high-dimensional weight vector space, it is not likely that many samples will satisfy the specifications in the first few iterations. As a consequence, we are left with limited information to update the sampling distribution and the resulting control policy.

To address this issue, we propose a rank-guided adaptive policy search, which incorporates an additional terminal cost associated with the rank of the specification state. This allows us to remove the rejection step, so that all sampled weight vectors will be associated with costs.

Definition 4. *The rank function* rank : $Q \to \mathbb{Z}_+$ *maps each specification state* $q \in Q$ *to a nonnegative integer and is defined recursively as*

$$\mathrm{rank}(q) = \begin{cases} 0, & \text{if } q \in F, \\ \min_{\sigma \in 2^{\mathcal{AP}}} \{1 + \mathrm{rank}(\delta(q,\sigma))\}, & \text{otherwise.} \end{cases}$$

It can be shown that given a set Q_k of states with rank k, the set of states Q_{k+1} with rank $k+1$ can be computed

$$Q_{k+1} = \{q \in Q \mid \forall 0 \le i \le k \text{ such that } q \notin Q_i \\ \text{and } \exists \sigma \in 2^{AP} \text{ such that } \delta(q,\sigma) \in Q_k\}.$$

In other words, Q_{k+1} contains a state which is not included in Q_i, for $i \le k$, and can take a labeled transition to visit a state q' in Q_k.

Next, we redefine the terminal cost function related to the specification state $h : Q \to \mathbb{R}$ defined by

$$h(q) = \begin{cases} 0 & \text{if } q \in F, \\ \mathrm{rank}(q) \cdot c & \text{otherwise.} \end{cases}$$

where c is a constant. We must select a constant c at least as large as the the cost incurred when the system trajectory triggers a transition in the specification automaton. In practice, this cost is chosen based on an estimated upper bound and does not need to be precise.

In summary, we developed a sampling-based method to search for an optimal weight defining a value function approximation. The sample distribution is iteratively updated based on simulated runs that assign a higher probability to those sampled weights that produce controllers with the best performance. The method is probabilistically complete. That is, with infinite number of samples, the algorithm is ensured to converge to the optimal solution of (7). Furthermore, to address the sample scarcity caused by rejecting unsatisfying weights, we introduced a meta-heuristic using the rank function of the specification automaton. Instead of rejecting weights, the penalty associated with nonzero rank allows us to assign lower probabilities to those weights that do not generate correct controllers.

5. EXAMPLES

In this section, we illustrate the correctness and efficiency of the proposed method with two case studies. The first is an example linear system, and the second is a nonlinear system—a Dubins car. The experiments are carried out in Matlab on an Intel(R) Core(TM) i7 CPU with 16 GB RAM.

5.1 Linear system with halfspace labels

We consider the linear quadratic system on $\mathcal{X} = \mathbb{R}^2$ with the specific parameters

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, Q = I, \quad R = 1, \quad \xi = 1,$$
$$x_0 = (0.5, 0.5), \quad x_f = (0, 0).$$

Let $\mathcal{AP} = \{a, b, c\}$ consist of atomic propositions that are true whenever the continuous state enters a specific region,

$$a : (x_1 \le -1), \quad b : (-1 < x_1 \le 1), \quad c : (x_1 > 1).$$

Note that each atomic proposition corresponds to a half space. Using $\mathcal{AP}$, we partition the state space into three regions $R_A = \{x \in \mathbb{R}^2 \mid x_1 \le -1\}$, $R_B = \{x \in \mathbb{R}^2 \mid -1 < x_1 \le 1\}$, and $R_C = \{x \in \mathbb{R}^2 \mid x_1 > 1\}$, with the following LTL specification

$$\varphi_1 = (A \to \Diamond B) \land (C \to \Diamond B) \land (\Diamond A \lor \Diamond C).$$

This specification ensures that either R_A or R_C must be reached, after which the system must eventually visit R_B. The automaton for this specification is shown in Fig. 1.

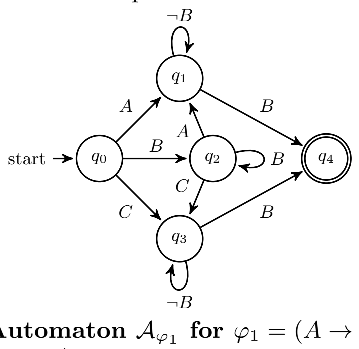


Figure 1: Automaton $\mathcal{A}_{\varphi_1}$ for $\varphi_1 = (A \to \Diamond B) \land (C \to \Diamond B) \land (\Diamond A \lor \Diamond C)$.

We consider value a function approximation with polynomial bases. The set of bases are $\{x_1^2, x_2^2, x_1x_2, x_1, x_2, 1\}$ for any specification state $q \in Q$. Thus, the total number of unknown weights is 30. Next, we employ the proposed algorithm to search for the approximate optimal weight vector $W^\star$. The following parameters are used: Initial sample size $N_0 = 100$, improvement parameter $\epsilon = 0.1$, smoothing parameter $\lambda = 0.2$, sample increment percentage $\alpha = 0.1$, and quantile parameter $\rho = 0.1$. The algorithm took 11 iterations to converge to the approximate optimal value function with the stopping criterion $\|\Sigma\| \leq 0.001$. The cost of the corresponding controller is 9.8384 with a terminal cost $100 \times \|x\|^2$ and a finite time horizon $T = 10$.

We test the controller for different initial states. As shown in Fig. 2b, a trajectory starting at $(0.5, 0.5)$ (near region R_C) visits R_C, then R_B and stabilizing to the origin. A trajectory starting at $(-0.5, -0.5)$ (near region R_A) visits R_A, then R_B in this order.

Remark. *When no terminal cost is considered, the approximate optimal controller can be obtained by convex optimization [20]. Using the proposed sampling-based method with sample size* 500 *for each iteration, the cost of the computed optimal controller is* 3.723*. The optimality can be improved by increasing sample size, which requires more computation. Overall, for linear quadratic systems with co-safe LTL specifications, the direct solution [20] is much more efficient. However, this solution does not apply to nonlinear systems, prompting the development of the sampling-based method.*

5.2 Dubins car system with obstacle

Given Dubins car dynamics $\dot{x} = u\cos(\theta)$, $\dot{y} = u\sin(\theta)$, and $\dot{\theta} = v$, where $x = (x, y, \theta) \in SE(2)$ is the state, and u and v are the inputs to the system (linear and angular velocities), we consider an LTL formula $\varphi_2 = \Diamond\,(\Diamond\,(A \wedge \Diamond\, B) \vee \Diamond\,(B \wedge \Diamond\, A) \wedge \Diamond\, C) \wedge \Box\neg\text{obs}$. The corresponding specification automaton is shown in Fig. 3. In this case, the system needs to traverse regions A and B in any order, and eventually reach C while avoiding collisions with a static obstacle. The running cost ℓ is the same as in the previous example, with terminal cost $100 \times \|(x, y) - (x_f, y_f)\|$. We define an additional cost $h(q) = 2 \times 10^3\,\text{rank}(q)$, where $\text{rank}(q)$ is the minimum number of transitions required to reach an accepting state from $q(T)$, where the terminal time T is 50.

In Dubins car case, we select radial basis function (RBF) bases. An RBF is defined by

$$\phi(x) = \exp(-\|x - x_c\|^2/2\sigma^2),$$

where x_c is the center of the basis element, and σ is a free parameter. We define $\phi_{\text{rbf}} = [\phi_1, \ldots, \phi_N]^T$ where the ϕ_is are RBFs centered on a uniform grid in x-y coordinates with step sizes $\delta x = 5$ and $\delta y = 5$. The workspace is bounded $0 \leq x \leq 30$ and $0 \leq y \leq 30$. The free parameter σ of each RBF is 5. We also define the bases $\phi_\theta = [\sin(\theta), \cos(\theta)]^T$ such that for each state $q \in Q$, the vector of basic bases is $\phi_{\text{basic}} = [\phi_{\text{rbf}}^T, \phi_\theta^T]^T$.

Additional RBF bases are determined by picking their centers as the centers of disk regions A, B, C, and obs. For example, in the state q_2, since the region B has been visited, the center of region A is taken into the consideration, so the basis function with the respect to the state q_2 is: $\phi_2 = [\phi_{\text{basic}}^T, \phi_A^T, \phi_{\text{obs}}^T]^T$ where ϕ_A (resp. ϕ_{obs}) is an RBF with its center on the center of disk A (resp. obs) and an RBF parameter $\sigma = 5$. Finally, the total number of basis functions (weight parameters) is 530. For the sampling-based algorithm, the same set of parameters is used, except that the initial sample size is chosen to be 500. A smaller sample size leads to faster updates, but longer iterations.

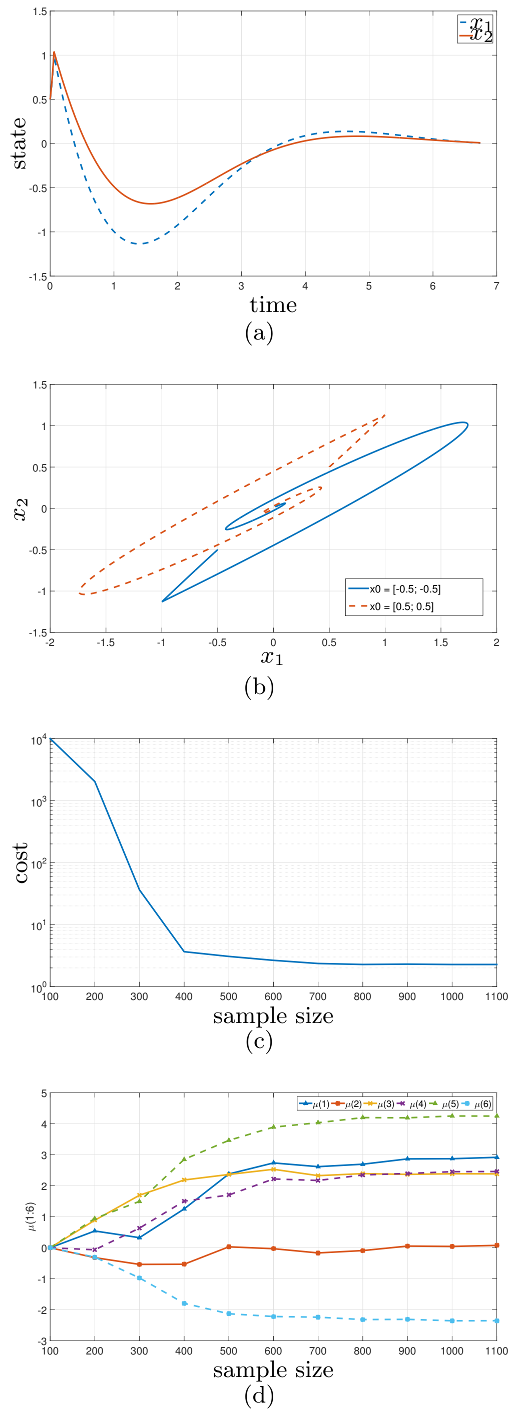


Figure 2: (a) The state trajectory for the linear system with $x_0 = (0.5, 0.5)$ calculated by the computed controller after the algorithm converges. (b) Approximate minimal cost trajectories satisfying the specification. The trajectories start with different initial states $x_0 = (-0.5, -0.5)$ and $x_0 = (0.5, 0.5)$ while the same controller is applied. (c) The cost over iterations. (d) The mean of the Gaussian distribution over iterations for the first three components in the mean vector.

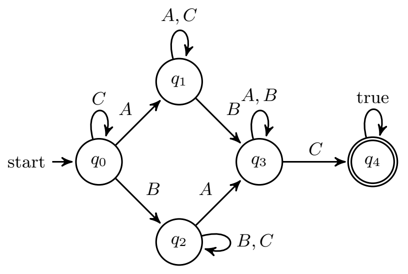


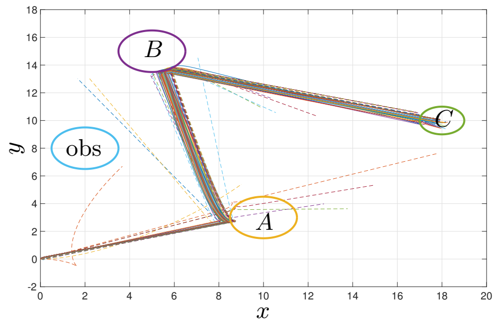
Figure 3: Automaton $\mathcal{A}_{\varphi_2}$ for $\varphi_2 = \Diamond(\Diamond(A \wedge \Diamond B) \vee \Diamond(B \wedge \Diamond A) \wedge \Diamond C) \wedge \Box\neg\mathrm{obs}$. The self-loops with labels other than A, B, C, and obs are omitted.

Fig. 4a shows the system trajectories computed using the value function approximation parameterized by μ over 179 iterations. Each iteration takes 20 to 30 seconds and the algorithm converges to an optimal controller after 73 iterations, with the same stopping criterion as in the linear example. Interestingly, the algorithm quickly finds a control policy that satisfies the specification after 15 iterations. Figs. 4b and 4c show the x and y state trajectories with and without initial error. The solid lines are the state trajectories with small initial errors. Even with small initial errors, the controller ensures satisfaction of the temporal logic constraints. Finally, Fig. 4d shows the convergence of the total cost at each iteration. The x-axis is the iteration number k, while the y-axis is the total cost value of the mean μ_k. Note that the mean of the distribution upon convergence may not be optimal due to the finite sample size. As a consequence, when the algorithm terminates under the terminal condition $\|\Sigma_k\| \leq 0.005$, the distribution may converge to the sub-optimal solution.

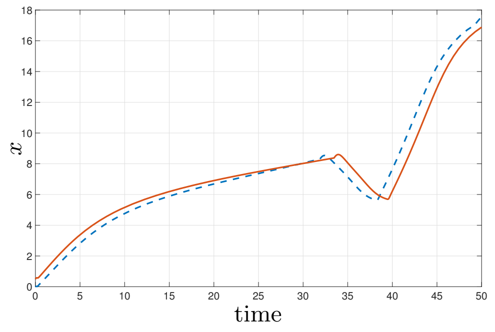
6. CONCLUSION

In this work, we presented a sampling-based method for optimal control with co-safe LTL constraints. Through a hybrid system formulation, the objective is to solve a sequence of value functions over a hybrid state space, where the continuous component comes from the continuous-time and continuous-state dynamics of the system, and the discrete component comes from the specification automaton. By approximating the value functions with weighted combinations of pre-defined bases, we employ model reference adaptive search (MRAS)—a general sampling-based optimization method—to directly search over weight parameter space. The method often works in a near-anytime fashion: it quickly finds a hybrid controller that satisfies the temporal logic constraint, and improves the value function approximation by minimizing an upper bound on the actual value function as more computation time is permitted. The algorithm converges, with probability one, to an optimal value function approximation. This procedure does not rely on discretizing the time/state space.

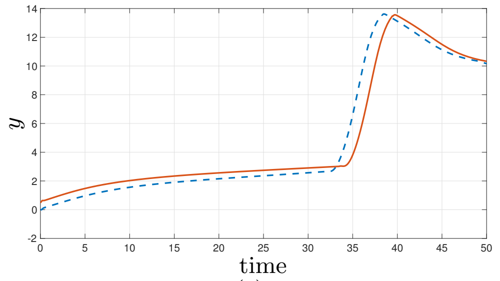
Building on this result, we consider several further developments. At this stage, this approach is limited to a subset of LTL specifications that admit deterministic and finite (rather than Büchi) automaton representations. Extensions to the general class of LTL specifications that admit deterministic Büchi automaton are subjects of current work. We will also consider decomposition-based distributed sampling so that the algorithm scales to problems in a high-dimensional weight parameter space. We will further extend



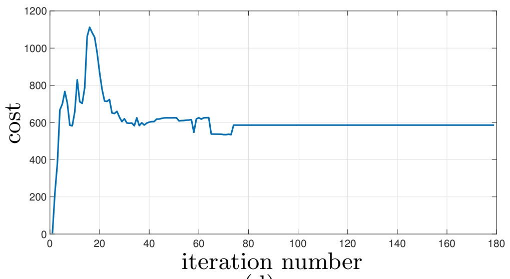
(a)



(b)



(c)



(d)

Figure 4: (a) trajectories for the Dubins car with the controller $u(x, q)$ computed from $\hat{V}(x; \mu)$ where μ is the mean of the Gaussian distribution over iterations. Dashed lines represent trajectories that do not satisfy the temporal logic constraints. (b–c) state trajectories of the Dubins car under the converged controller and errors in the initial states. (d) cost over iterations.

the proposed method as a building block in anytime optimal and provably correct decision making systems for nonlinear robotic systems. Finally, the sampling algorithm is highly parallelizable, suggesting the use of GPU-accelerated computing to speed up computations.

Acknowledgments

This work was supported in part by grants from AFRL #FA8650-15-C-2546, ONR #N00014-15-IP-00052, and NSF #1550212.

7. REFERENCES

[1] M. Abu-Khalaf and F. L. Lewis. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach. *Automatica*, 41(5):779–791, 2005.

[2] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[3] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[4] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In G. Berry, H. Comon, and A. Finkel, editors, *International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65, Paris, France, July 2001. Springer.

[5] L. C. G. J. M. Habets and C. 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.

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

[7] S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3972–3977, 1999.

[8] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Transactions on Automatic Control*, 47(9):1536–1540, 2002.

[9] J. Hu, M. C. Fu, and S. I. Marcus. A model reference adaptive search method for global optimization. *Operations Research*, 55(3):549–568, 2007.

[10] M. Johansson and A. Rantzer. Computation of piecewise quadratic Lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):555–559, Apr. 1998.

[11] N. Kariotoglou, S. Summers, T. Summers, M. Kamgarpour, and J. Lygeros. Approximate dynamic programming for stochastic reachability. In *European Control Conference (ECC)*, pages 584–589, July 2013.

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

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

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

[15] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3):291–314, Nov. 2001.

[16] T. Latvala. Efficient model checking of safety properties. In T. Ball and S. K. Rajamani, editors, *International SPIN Workshop on Model Checking of Software*, volume 2648 of *Lecture Notes in Computer Science*, pages 74–88. Springer, 2003.

[17] F. Lewis, S. Jagannathan, and A. Yesildirak. *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.

[18] J. Liu and N. Ozay. Abstraction, discretization, and robustness in temporal logic control of dynamical systems. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 293–302. ACM, 2014.

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

[20] I. Papusha, J. Fu, U. Topcu, and R. M. Murray. Automata theory meets approximate dynamic programming: Optimal control with temporal logic constraints. In *IEEE Conference on Decision and Control (CDC)*, Dec. 2016.

[21] N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive (1) designs. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 364–380. Springer, 2006.

[22] R. Reemtsen and J.-J. Rückmann. *Semi-infinite programming*, volume 25. Springer Science & Business Media, 1998.

[23] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[24] S. L. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning for surveilance with temporal-logic constraints. *International Journal of Robotics Research*, 30(14):1695–1708, Dec. 2011.

[25] K. Weierstrass. Über die analytische Darstellbarkheit sogenannter willhülicher Functionen einer reellen Veränderlichen. *Berliner Berichte*, 1885.

[26] E. M. Wolff, U. Topcu, and R. M. Murray. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4332–4339, Nov. 2013.

[27] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817–2830, Nov. 2012.

[28] X. Xu and P. J. Antsaklis. Optimal control of switched systems based on parameterization of the switching instants. *IEEE Transactions on Automatic Control*, 49(1):2–16, 2004.