Sampling-based Trees of Polytopes for Approximate Optimal Control of Piecewise Affine Systems

Sadra Sadraddini and Russ Tedrake

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
32 Vassar st, Cambridge, MA 02139
{sadra,russt}@mit.edu

Abstract. Piecewise affine (PWA) systems are widely used to model highly nonlinear behaviors such as contact dynamics in robot locomotion and manipulation. Existing control techniques for PWA systems have computational drawbacks, both in offline design and online implementation. In this paper, we introduce a method to obtain feedback control policies and a corresponding set of admissible initial conditions for discrete-time PWA systems such that all the closed-loop trajectories reach a goal polytope, while a cost function is optimized. The idea is conceptually similar to LQR-trees [21], which consists of 3 steps: (1) open-loop trajectory optimization, (2) feedback control for computation of "funnels" of states around trajectories, and (3) repeating (1) and (2) in a way that the funnels are grown backward from the goal in a tree fashion and fill the state-space as much as possible. We show PWA dynamics can be exploited to combine step (1) and (2) into a single step that is tackled using mixed-integer convex programming, which makes the method more suitable for dealing with hard constraints. Illustrative examples on contact-based dynamics are presented.

1 Introduction

Many interesting behaviors in robotics are captured by highly nonlinear models. A prominent class is control through multiple contacts, where the robot must make and break contact with the environment in order to achieve its objectives. Examples include walking [4,6,8], running [9], dexterous manipulation [17], and push-recovery [18]. A popular way to approach contact-based dynamics is using piecewise affine (PWA) models that characterize the system using multiple linearizations with appropriate switching rules (e.g., those caused by contact). While more accurate robot models are fully nonlinear, PWA models can be viewed as a natural extension of local linear approximations in multi-contact setting [15,23]. Apart from robotics, PWA models are also popular in systems such as signalized traffic networks [16] and gene circuits [5].

Controlling PWA systems is difficult since the controller has to determine both the temporal order of modes and the inputs applied at each one. Completeness is important - not finding a solution when one exists is undesirable. Given an initial condition and a goal, the (optimal) trajectory that steers the state to the goal while respecting the dynamics and state/control constraints can be obtained using mixed-integer convex programming (MICP). Time is discretized to obtain a finite number of decision variables. Given a discrete-time PWA model and a fixed time horizon (steps required to get into the goal), MICP approaches are sound and complete - they find optimal solutions if they exist. However, they come at a large computational price. Their unreliable computation time, even for obtaining a feasible solution instead of the optimal one, hinders online implementation for robotic tasks with fast dynamics. While there is a great deal of research on improving the runtime of MICP solvers, they are still orders of magnitude too slow for most robotic control problems.

It is desirable to move much of the computational burden to offline phase so the controller at real-time is a lookup table of simple control laws. However, existing approaches are not efficient even for relatively small systems. One candidate is state/control space discretization, which results in (non-deterministic) finite-state abstractions. The controller is found by solving standard graph games [25]. But discretization scales poorly with the number of state and control dimensions. An alternative approach is synthesizing PWA control laws corresponding to stabilizing piecewise quadratic (PWQ) Lyapunov functions [10]. However, the state-control partition that produces the control laws is assumed to be the same as those of PWA dynamics, which is very restrictive and the method may fail.

A theoretically popular approach is using multi-parametric programming [7] for model predictive control (MPC), which results in explicit hybrid MPC schemes that also provide PWA control laws [1]. This approach is complete if the horizon is fixed. But explicit MPC using parametric programming does not scale beyond very simple problems. A method was proposed in [15] to bridge the gap between explicit and full-blown online hybrid MPC, where a set of useful mode sequences was identified in advance - a similar approach was taken in [11] - and sampling-based polyhedral projection methods was used to replace computationally intensive traditional variable elimination, enabling its application to moderate sized multi-contact problems (e.g., a push-recovery scenario with 10 states and 5 inputs). However, this approach still suffers from the computational complexity caused by the number of integers - for those initial conditions that the horizon is long, computations become prohibitive. Moreover, it relies on vertex representation of polytopes, which is undesirable in high dimensions.

Since the problem is of reachability class, it is amenable to ideas in sampling-based motion planning [12], where methods such as rapidly exploring random trees (RRTs) and its variants are well-studied [13]. While there has been work on RRTs for hybrid systems [3], they provide little robustness understanding as the nodes in the tree correspond to points in the state-space. Therefore, there is no formal guarantee that a trajectory that deviates from the points is able to recover and achieve the goal. Moreover, PWA constraints pose a challenge for choosing an appropriate metric in exploring the state-space.

The authors in [21] used sums-of-square (SOS) programming to obtain regions of attractions (funnels) for (time-varying) linear-quadratic regulators (LQR) that stabilized goal-reaching trajectories, which are obtained using nonlinear optimization in advance. These regions are recomputed and grown backward from the goal in a tree fashion. This technique is called *LQR-trees* and was originally introduced for smooth continuous-time systems, but was also applied to limit cycle stabilization of hybrid systems in [19]. Transverse dynamics was studied to deal with switching surfaces [14]. However, this approach becomes complicated for systems with many modes and state/input constraints and is highly dependent on nonlinear optimization. We desire an approach that exploits the properties of PWA systems and mitigates all the mentioned concerns.

In this paper, we propose a method to control PWA systems that uses ideas both in hybrid MPC using MICP, and regions of admissible states similar to funnels in SOS-based verification in LQR-trees. The state-space is filled by growing trees of such admissible sets. The approach can be viewed as a discrete-time PWA version of LQR-trees. It has the significance that the MICP is used to compute trajectories and admissible regions of states, which are polytopes, simultaneously. Once the tree of polytopes is computed, online implementation become as simple as few matrix multiplications/small convex programs. We also consider optimality subject to a given cost and obtain a cost-to-go function that over-approximates the optimal one. The main contributions of this paper are:

– A framework for fusing trajectory optimization with corresponding polytopes of admissible states into a single MICP problem.
– Sampling-based heuristics (similar to RRT/RRT* [12]) to grow a tree of polytopes backward from the goal such that the union of polytopes cover the state-space, and all the closed loop trajectories are close to optimal, both as much as possible. Similar to LQR-trees, we prove probabilistic feedback coverage: as the number of samples go to infinity, the union of polytopes cover the whole region of admissible initial conditions with probability one.

This paper is organized as follows. In Sec. 2, we provide the notation. The problem is stated in Sec. 3. Technical details on polytopes computation and tree construction are provided in Sec. 4 and Sec. 5, respectively. Implementation details are discussed in Sec. 6. Examples are presented in Sec. 7.

2 Notation

The set of real, non-negative real, integer numbers, and empty set are denoted by $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{N}$, and $\emptyset$, respectively. Given $A_1 \in \mathbb{R}^{n_1 \times m}, A_2 \in \mathbb{R}^{n_2 \times m}$, we use $[A_1, A_2] \in \mathbb{R}^{(n_1+n_2)\times m}$ to denote the matrix obtained by stacking A_1 and A_2 vertically. Given a set $\mathbb{S} \subset \mathbb{R}^n$, its interior and volume are denoted by $\text{int}(\mathbb{S})$ and $\text{Vol}(\mathbb{S})$, respectively. Given $\mathbb{S} \subset \mathbb{R}^n$, $\mathbb{S}_1, \cdots, \mathbb{S}_n, \mathbb{S}_i \subset \mathbb{R}^n$, construct a partition of $\mathbb{S}$ if $\mathbb{S}_i \neq \emptyset$, $i = 1, \cdots, n$, $\bigcup_{i=1}^n \mathbb{S}_i = \mathbb{S}$, and $\text{int}(\mathbb{S}_i) \cap \text{int}(\mathbb{S}_j) = \emptyset$ if $i \neq j$. Given $\mathbb{S} \subset \mathbb{R}^n$ and $A \subset \mathbb{R}^{n_A \times n}$, we interpret $A\mathbb{S}$ as $\{As | s \in \mathbb{S}\}$. Given two sets $\mathbb{S}_1, \mathbb{S}_2 \subset \mathbb{R}^n$, their Minkowski sum is denoted by $\mathbb{S}_1 \oplus \mathbb{S}_2 = \{s_1 + s_2 | s_1 \in \mathbb{S}_1, s_2 \in \mathbb{S}_2\}$. The vector of all ones is denoted by $\underline{1}$, where the dimension is unambiguously interpretable from the context. A *polyhedron* $\mathbb{H} \subset \mathbb{R}^n$ is the intersection of a finite number of closed half-spaces in the form $\mathbb{H} = \{x \in \mathbb{R}^n | Hx \leq h\}$, where $H \in \mathbb{R}^{n_H \times n}, h \in \mathbb{R}^{n_H}$. All inequality relations are interpreted element-wise in this paper. A bounded polyhedron is called a *polytope*.

3 Problem Formulation

We study discrete-time systems of the form

$$x_{t+1} = F(x_t, u_t), \tag{1}$$

where $x_t \in \mathbb{X}, \mathbb{X} \subset \mathbb{R}^n$, is the state at time t, $u_t \in \mathbb{U}, \mathbb{U} \subset \mathbb{R}^m$, is the control input at time t, $t \in \mathbb{N}$, and $F : \mathbb{X} \times \mathbb{U} \rightarrow \mathbb{X}$ is a PWA function given as:

$$F(x,u) = \begin{cases} A_1 x + B_1 u + c_1, & [x,u] \in \mathbb{H}_1, \\ \vdots & \vdots \\ A_{n_\mathcal{M}} x + B_{n_\mathcal{M}} u + c_{n_\mathcal{M}}, & [x,u] \in \mathbb{H}_{n_\mathcal{M}}, \end{cases} \tag{2}$$

where $n_\mathcal{M} \in \mathbb{N}$ is the number of modes, $\mathbb{H}_1, \cdots, \mathbb{H}_{n_\mathcal{M}}$, construct a partition of $\mathbb{X} \times \mathbb{U}$, and $A_i \in \mathbb{R}^{n \times n}, B_i \in \mathbb{R}^{n \times m}$ and $c_i \in \mathbb{R}^n$ are constant matrices, representing affine dynamics in mode i. We use the shorthand notation $\mathcal{M} := \{1, \cdots, n_\mathcal{M}\}$. We assume $\mathbb{X}$, $\mathbb{U}$ and all partitions are polytopes.

Problem 1. Given a PWA system (2), a goal polytope $\mathbb{X}_{\text{Goal}} \subset \mathbb{X}$, find the largest set of initial conditions $\mathbb{X}_{\text{initial}} \subseteq \mathbb{X}$ and a control policy $\pi : \mathbb{X} \rightarrow \mathbb{U}$ such that all the points in $\mathbb{X}_{\text{initial}}$ are steered into $\mathbb{X}_{\text{Goal}} \subset \mathbb{R}^n$. Moreover, if multiple strategies are available, choose the one that minimizes the following cost function:

$$J = \sum_{t=0}^{T_f} \gamma_i(x_t, u_t), \tag{3}$$

where $\gamma_i : \mathbb{H}_i \rightarrow \mathbb{R}, i \in \mathcal{M}$, are linear/quadratic functions, x_0 is the initial state, $(x_t, u_t) \in \mathbb{H}_i$, $u_t = \pi(x_t), t = 0, 1, \cdots, T_f - 1, x_{T_f} \in \mathbb{X}_{\text{Goal}}$, and $T_f \in \mathbb{N}$.

The solution to Problem 1 consists of both the control policy π and the set of admissible initial conditions $\mathbb{X}_{\text{initial}}$. Finding representations for both is difficult. However, given $x \in \mathbb{X}$ and $T \in \mathbb{N}$, we can write the following MICP problem:

$$\begin{array}{rl} u_0^*, u_1^*, \cdots, u_T^* = \arg\min & \sum_{t=0}^{T_f} \gamma_i(x_\tau, u_\tau) \\ \text{such that} & x_{\tau+1} = F(x_\tau, u_\tau), (x_\tau, u_\tau) \in \mathbb{H}_i, \\ & \tau = 0, 1, \cdots, T-1, x_0 = x, x_T \in \mathbb{X}_{\text{target}}. \end{array} \tag{4}$$

Eq. (4) provides an open-loop plan. We have $x \in \mathbb{X}_{\text{initial}}$ if and only if (4) is feasible for some $T \in \mathbb{N}$. Optimality is more subtle, as one has to check solutions for all $T \in \mathbb{N}$ and pick the best. For some problems all the optimal solutions have the smallest T - an obvious example is time-optimality where $\gamma_i = 1, \forall i \in \mathcal{M}$.

As discussed in the introduction, solving (4) can be too slow for online implementation. We desire faster feedback policies. Parametric MICP is not the wise solution as it leads to enumerating all possible mode sequences, which quickly becomes intractable. We introduce a method that incrementally builds a set of admissible initial conditions, asymptotically reaches $\mathbb{X}_{\text{initial}}$, but sacrifices strong claims on optimality. The details are discussed in the following sections.

4 Trajectories of Polytopes

In this section, we introduce the first part of our solution to Problem 1. We propose a method for obtaining trajectories of polytopes to a target polytope, which is not necessarily the same as $\mathbb{X}_{\text{Goal}}$ - it corresponds to a node in the tree that is formalized in Sec. 5. We focus on solving the following subproblem throughout this section.

Subproblem 1 *Given a target polytope $\mathbb{X}_{target}$ and $T \in \mathbb{N}$, find a sequence of polytopes $\mathbb{X}_0, \mathbb{X}_1, \cdots, \mathbb{X}_T$ such that:*

1. *(polytope-to-polytope dynamics) for all $x \in \mathbb{X}_\tau, 0 \leq \tau < T$, there exists $u \in \mathbb{U}$ such that $F(x,u) \in \mathbb{X}_{\tau+1}$ and $(x,u) \in \mathbb{H}_i$ for some $i \in \mathcal{M}$.*
2. *(target constraint) $\mathbb{X}_T \subseteq \mathbb{X}_{target}$.*

Note that Subproblem 1 often does not have a unique solution. We will add cost criteria later in the paper mainly in order to obtain large polytopes.

4.1 Parameterization

We characterize polytopes by affine transformations of a pre-defined polytope $\mathbb{P} \subset \mathbb{R}^{n_p}$, where $\mathbb{P} := \{x \in \mathbb{R}^{n_p} | Px \leq \underline{1}\}$. The user has to choose $\mathbb{P}$. For example, the unit cube with $n_p = n$ is a simple option; we highlight its advantages and limitations later in the paper. The polytope of possible states at time t is given by:

$$\mathbb{X}_t = \{\bar{x}_t\} \oplus G_t \mathbb{P}, \tag{5}$$

where $\bar{x}_t \in \mathbb{R}^n$ and $G_t \in \mathbb{R}^{n \times n_p}$ are parameters that we search over. Given $x \in \mathbb{X}_t$, we compute $p(x) \in \mathbb{P}$ such that $x = \bar{x}_t + G_t p(x)$. Note that $p(x)$ may not be unique, e.g., consider when $n_p > n$ and G_t is full rank. A special case is when $n = n_p$ and G_t is an invertible matrix, where we have the linear relation $p(x) = G^{-1}(x - \bar{x}_t)$. Otherwise, given $\bar{x}_t$ and G_t, $p(x)$ is determined from the following linear program (with zero or some ad-hoc cost):

$$p(x) = \text{find } p \in \mathbb{P} \text{ such that } x = \bar{x}_t + G_t p$$

We propose the following control law:

$$u_t(x) = \bar{u}_t + \theta_t p(x). \tag{6}$$

where $\bar{u}_t \in \mathbb{R}^m$ and $\theta_t \in \mathbb{R}^{m \times n_p}$ are parameters that, again, we search over. The set of all possible control inputs induced by (6) at time t is $\mathbb{U}_t := \{\bar{u}_t\} + \theta_t \mathbb{P}$.

4.2 Mixed-Integer Formulation

Trajectory: Let $i \in \mathcal{M}$ be such that $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{S}_i$. In other words, we restrict that the product of each state polytope and its corresponding polytope of control inputs lie in a single mode of the PWA system - the constraint ensuring this is discussed shortly. Using (6), we arrive at the following evolution for $\mathbb{X}_t$:

$$\bar{x}_{t+1} = A_i \bar{x}_t + B_i \bar{u}_t + c_i \tag{7a}$$

$$G_{t+1} = A_i G_t + B_i \theta_t \tag{7b}$$

We encode (7) using *big-M method*, which is a standard procedure for translating PWA systems into mixed-integer constraints. We introduce binary variables $\delta_t^i \in \{0,1\}$, which are used in a way that δ_t^i takes 1 if mode at time t is i, and zero otherwise. Thus, we have the constraint:

$$\sum_{i \in \mathcal{M}} \delta_t^i = 1, t = 0, \cdots, T. \tag{8}$$

Eq. (7) is encoded as follows. For all $i \in \mathcal{M}$, we have:

$$A_i \bar{x}_t + B_i \bar{u}_t + c_i - \mathrm{M}(1-\delta_t^i) \leq \bar{x}_{t+1} \leq A_i \bar{x}_t + B_i \bar{u}_t + c_i + \mathrm{M}(1-\delta_t^i), \tag{9a}$$

$$A_i G_t + B_i \theta_t - \mathrm{M}(1-\delta_t^i) \leq G_{t+1} \leq A_i G_t + B_i \theta_t + \mathrm{M}(1-\delta_t^i), \tag{9b}$$

where M is a sufficiently large positive number. We omit discussions on binary big-M encoding and its soundness conditions, as they are thoroughly studied in the literature, see, e.g., [2].

Subset Maintenance: In order to ensure that (7) is sound, we need to enforce that for some $i \in \mathcal{M}$, $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{H}_i$. Given $i \in \mathcal{M}$, it becomes a set of linear constraints due to the following lemma.

Lemma 1 (Extended Farkas' Lemma [20]). *Consider two polytopes* $\mathbb{Y}_1 = \{y \in \mathbb{R}^{n_1} \big| H_1 y \leq h_1\}$ *and* $\mathbb{Y}_2 = \{y \in \mathbb{R}^{n_2} \big| H_2 y \leq h_2\}$*. Given* $L \in \mathbb{R}^{n_1 \times n_2}$ *and* $\bar{y} \in \mathbb{R}^{n_2}$*, the following statements are equivalent:*

1. $L\{\bar{y} \oplus \mathbb{Y}_2\} \subseteq \mathbb{Y}_1$.
2. $\exists \Lambda \geq 0$ such that $\Lambda H_1 = H_2 L$ *and* $\Lambda h_1 \leq h_2 - H_2 \bar{y}$.

We use Lemma (1) in the following means:

1. The target constraint $\mathbb{X}_T \subseteq \mathbb{X}_{\text{target}}$ is equivalent to

$$\Lambda_T \in \mathbb{R}_+^{n_p \times n_{h_{\text{target}}}}, \Lambda_T P = H_{\text{target}} G_T, \Lambda_T \underline{1} \leq h_{\text{target}} - H_{\text{target}} \bar{x}_T, \tag{10}$$

   where $\mathbb{X}_{\text{target}} = \{x \in \mathbb{R}^n | H_{\text{target}} x \leq h_{\text{target}}\}$, and $n_{h_{\text{target}}}$ is the number of rows in H_{target}. Note that this is a linear constraint.
2. We need mode constraint $\exists i \in \mathcal{M}$ such that $\mathbb{X}_t \times \mathbb{U}_t \subseteq \mathbb{H}_i$. However, this is not a convex constraint as the mode is not specified. We need a mixed-integer version of Lemma 1 that is encoded using big-M method as follows:

$$\Lambda_{t,i} \in \mathbb{R}_+^{n_p \times n_{h_i}}, \Lambda_{t,i} P = H_i [G_t, \theta_t], \Lambda_{t,i} \underline{1} \leq h_i - H_i [\bar{x}_t, \bar{u}_t] + \mathrm{M}(1-\delta_t^i), \tag{11}$$

   where $\mathbb{H}_i = \{[x,u] \in \mathbb{R}^{n+m} | H_i [x,u] \leq h_i\}$, $t = 0, 1, \cdots, T-1$, n_{h_i} is the number of rows in H_i, and $i \in \mathcal{M}$.

4.3 Computation

The solution to Subproblem 1 is characterized by finding numerical values for $\zeta := \{\bar{x}_\tau, \bar{u}_\tau, G_\tau, \theta_\tau\}_{\tau=0,1,\cdots,T}$, which is obtained by solving the following optimization problem:

$$\begin{aligned} \zeta^* = \arg\min \quad & \alpha(\zeta) \\ \text{such that} \quad & (8), (9), (10), (11) \end{aligned} \tag{12}$$

where α is a cost function with appropriate domain. Note that if a trajectory of points $(\bar{x}_0, \bar{u}_0), \cdots, (\bar{x}_T, \bar{u}_T)$ is feasible, a trivial solution $G_\tau = 0, \theta_\tau = 0, \tau = 0, \cdots, T$ will be obtained, i.e., singletons instead of full-dimensional polytopes. We address this issue by introducing heuristics for α to obtain large polytopes.

4.4 Volume Maximization

We desire polytopes $\mathbb{X}_0, \cdots, \mathbb{X}_T$ that are large in the sense they cover the state-space as much as possible. An ideal approach is maximizing the volume of the union of polytopes, which is a nonlinear objective. Note that polytopes may overlap. When $n_p = n$, a simple heuristic is to maximize the trace of G_0 - a linear objective. If the choice $\mathbb{P}$ is a symmetric set (i.e., $\forall p \in \mathbb{P}$, we have $-p \in \mathbb{P}$), then multiplying any column of G_0 by -1 does not change $\mathbb{X}_0$. Therefore, for G_0, we can safely assume that all the diagonal terms are positive without any restriction posed on the space of solutions. A drawback of trace maximization is that it may still lead to zero volume. Additionally, it often leads to sparse solutions in a way that few diagonal terms take large values, but others become zero. We found that including the following heuristics is useful in (12) when $\mathbb{P} = [-1, 1]^n$: First, weighted combination of l_1 and l_∞ norms of the vector of diagonal terms of G_0 included in $\alpha(\zeta)$. Second, Restrict G_0 to be an upper/lower triangular matrix, and constrain all its diagonal terms to be greater than a small positive value. In this case, feasible solutions certainly have non-zero volume. Third, include weighted summation of traces of $G_1, \cdots, G_T$ in $\alpha(\zeta)$. This promotes larger polytopes for $\mathbb{X}_1, \cdots, \mathbb{X}_T$. But there is no guarantee for this heuristic to work as dynamical constraints dominate the relation between subsequent polytopes. Finally, if very small volume is reached, reject the solution and resolve the optimization problem with a different objective. Note that the mentioned heuristics make (12) a a mixed-integer linear program (MILP) problem.

Remark 1. Maximizing the determinant of a square matrix subject to linear constraints can be cast as a semidefinite program (SDP) [24], which may prove useful for our application. While we leave investigation of this approach to future work, we note that it converts (12) to mixed-integer semidefinite programming (MISDP), for which solvers are still not as mature as MILP/MIQP solvers.

4.5 Complexity

The complexity of MICP solvers grow exponentially with number of integers involved, in general. The method introduced in this section introduces $Tn_{\mathcal{M}}$

binary variables, which is the same as (4). However, the price is introducing more continuous variables and constraints due to Lemma 1. The reward is that we obtain a family of trajectories and a set of admissible initial conditions.

5 Random Trees of Polytopes

In this section, we provide the solution to Problem 1 by solving Subproblem 1 multiple times such that a tree is grown backward from the goal polytope. The tree is formalized in Sec. 5.1. The technique is conceptually similar to the procedure in RRT and LQR-trees, which is detailed in Sec. 5.2. In Sec. 5.3, we briefly explain a modification aimed at improving optimality, which is similar to rewiring nodes in RRT*. Formal guarantees are discussed in Sec. 5.4.

5.1 Tree Structure

Definition 1. *A directed, rooted, labeled tree (which we simply refer to as* tree *in the rest of the paper) is a tuple $\mathcal{T} = (\mathcal{V}, v_0, \text{child}, \mathcal{C})$, where $\mathcal{V}$ is the set of nodes, $v_0 \in \mathcal{V}$ is the root node,* $\text{child} : \mathcal{V} \setminus \{v_0\} \to \mathcal{V}$ *is a function that maps each node to its unique successor (child) in the graph - the root does not have a child, and $\mathcal{C} : \mathcal{V} \to \mathbb{R}$ is a cost function that maps each node to a real value.*

The tree has the property that for all nodes $v \in \mathcal{V} \setminus \{v_0\}$, a unique path toward the root exists: $v, \text{child}(v), \text{child}(\text{child}(v)), \cdots, v_0$. The correspondence of the elements in Definition 1 with the solution to Problem 1 are as follows. Each node is a polytope in $\mathbb{X}$. We denote polytope corresponding to $v \in \mathcal{V}$ by $\mathbb{X}_v$. The root node is the goal polytope: $\mathbb{X}_{v_0} = \mathbb{X}_{\text{goal}}$. For all $\mathbb{X}_{\text{child}(v)}$, $v \in \mathcal{V} \setminus \{v_0\}$, we have already computed a control policy $\mu : \mathbb{X}_v \to \mathbb{U}$ such that 1) (unique mode) there exists $i \in \mathcal{M}$ such that $\{[x, \mu(x)] | x \in \mathbb{X}_i\} \subseteq \mathbb{H}_i$, 2) (successor constraint) $\{A_i x + B_i \mu(x) + c_i | x \in \mathbb{X}_v\} \subseteq \mathbb{X}_{\text{child}(v)}$. The cost $\mathcal{C}(\mathbb{X}_v)$ is the worst-case cost induced by moving from a point in $\mathbb{X}_v$ to $\mathbb{X}_{\text{child}(v)}$ using the available control strategy $\mathcal{C}(v) = \max_{x \in \mathbb{X}_v} c_i(x, \mu(x))$. Using control law (6), the cost is computed through the following optimization problem:

$$\mathcal{C}(\mathbb{X}_v) = \max_{p \in \mathbb{P}} \gamma_i(\bar{x}_v + G_v p, \bar{u}_v + \theta_v p), \tag{13}$$

where $\mathbb{X}_v = \{\bar{x}_v\} + G_v \mathbb{P}$, $G_{\text{child}(v)} = A_i G_v + B_i \theta_v, \bar{x}_{\text{child}(v)} = A_i \bar{x}_v + B_i \bar{u}_v + c_i$. The choice of γ_i's are made such that (13) is a convex program or it can be easily computed - over-approximations instead of exact solutions are valid for statements made in Sec. 6. Once the tree is available, we construct the value or cost-to-go function $V : \mathcal{V} \to \mathbb{R}$, which is recursively computed as:

$$V(v) = \mathcal{C}(v) + V(\text{child}(v)), \tag{14}$$

where $v \in \mathcal{V} \setminus \{v_0\}$, and $V(v_0) = 0$. As explained in Sec. 6, the cost-to-go of nodes provides an upper-bound for the cost-to-go of states, and is a key component of the controller. The set of all states in the tree is given as $\mathbb{X}_{\text{tree}} = \bigcup_{v \in \mathcal{V}} \mathbb{X}_v$.

5.2 Growing the tree

Distance From a Polytope. First, we require a routine that computes the distance between $x \in \mathbb{X}$ and $\mathbb{X}_v = \{\bar{x}_v\} + G_v\mathbb{P}$ - it should be zero if and only if $x \in \mathbb{X}_v$. A natural candidate is the following optimization problem:

$$d(x, \mathbb{X}_v) = \begin{array}{ll} \min & \|\delta\|_l \\ \text{such that} & x + \delta = \bar{x}_v + G_v p, p \in \mathbb{P}, \end{array} \tag{15}$$

where $l \in \{1, 2, \infty\}$ makes (15) a LP/QP. However, we desire faster closed-form solutions since we implement the distance function many times both in tree construction (offline) and controller implementation (online). In the rest of the paper, we ease the computations with the following assumption.

Assumption 1 *We have* $\mathbb{P} = [-1, 1]^n$.

Note that by making Assumption 1, all polytopes become paralleltopes. Then $x = \bar{x}_v + G_v p, p \in \mathbb{P}$, can be rewritten as $\|G_v^{-1}(x - \bar{x}_v)\|_\infty \leq 1$. However, it requires G_v to be invertible. Despite heuristics for obtaining full-dimensional polytopes, it is still possible that some polytopes have zero volume. We circumvent this numerical issue as follows. We use singular value decomposition (SVD) to write $G_v = USV^T$, where U and V are unitary matrices and S is a diagonal matrix of singular values. Then we replace S by $S^\epsilon := S + \epsilon I$, where $\epsilon > 0$ is a small number, typically in the order of machine precision. Let $G_v^\epsilon := US^\epsilon V$, which is an invertible matrix. Define $\mathbb{X}_v^\epsilon = \{\bar{x}_v\} + G_v\mathbb{P}$. Notice that $\mathbb{X}_v^\epsilon$ is not that different from $\mathbb{X}_v$ as it is easy to verify $\mathbb{X}_v^\epsilon \subseteq \mathbb{X}_v \oplus \epsilon\mathbb{P}$. Introduce $p_v(x) := {G_v^\epsilon}^{-1}(x - \bar{x}_v)$. Notice that $x \in \mathbb{X}_v^\epsilon \Leftrightarrow p_v \in \mathbb{P}$. Then define $p_v^*(x) := \min(1, \max(p_v(x), -1))$, where min and max operations are implemented row-wise. Then define

$$d_v(x) := \lim_{\epsilon \to 0} \|G_v^\epsilon(p_v(x) - p_v^*(x))\|_\infty \tag{16}$$

which is basically the l_∞ distance between x and the point in $\mathbb{X}_v^\epsilon$ which their l_∞ distance is minimal in the space transformed by ${G_v^\epsilon}^{-1}$ (which inherits metric properties as it is a one-to-one transformation). The main advantage of the unconventional distance (16) is that it mostly cancels out the effect of ϵ, and more importantly, it can be cast as elementary matrix operations (multiplications and row-wise min, max), so it is efficiently implementable in a computer program for a large number of polytopes in parallel - basically we stack (16) for all $v \in \mathcal{V}$.

Sample and Rejection. We assume we are given a function `sample` which randomly selects points from $\mathbb{X}$ with total support, and is repeated in an independent and identically distributed (i.i.d.) fashion. We select points from $\mathbb{X} \backslash \mathbb{X}_{\text{tree}}$ by rejecting samples in $\mathbb{X}_{\text{tree}}$. Note that $x \in \mathbb{X} \setminus \mathbb{X}_{\text{tree}}$ iff $\{d_v(x) = 0 | v \in \mathcal{V}\} = \emptyset$.

Tree initialization We initially solve Subproblem 1 with no constraints for $\bar{x}_0$ and focus on obtaining large volumes for polytopes. There is a trade-off in

Algorithm 1 Random Trees of Polytopes

```
Require: Initial $\mathcal{T}^1 = (\mathcal{V}, v_0, \text{child}, \mathcal{C})$, $K$ = number of iterations
 1: for step 2 to $K$ do
 2:     Select $x_{\texttt{sample}} \in \mathbb{X} \setminus \mathbb{X}_{\text{tree}}^{K-1}$
 3:     for $v \in \mathcal{V}$ do
 4:         Choose $T$ randomly from $\{1, \cdots, T_{\max}\}$
 5:         Solve Subproblem 1 with $\bar{x}_0 = x_{\texttt{sample}} + \eta$ and $\mathbb{X}_{\text{target}} = \mathbb{X}_v$
 6:         if feasible solution exists and meets criteria then
 7:             Add $v_0, \cdots, v_{T-1}$, corresponding to $\mathbb{X}_0, \cdots, \mathbb{X}_T$, to $\mathcal{V}$
 8:             $\text{child}(v_\tau) = v_{\tau+1}, \tau = 0, \cdots, T-2$, $\text{child}(v_{T-1}) = v$.
 9:             Update cost and value functions of $\mathcal{T}^K$.
10:             break
```

choosing T. If T is small, then computations are faster, but we obtain fewer polytopes. Larger T leads to larger problem size, and often (but not necessarily) finds larger polytopes. Once a solution to Subproblem 1 is obtained, we initialize the tree by adding nodes $v_0, \cdots, v_{T-1}$ corresponding to $\mathbb{X}_0, \cdots, \mathbb{X}_{T-1}$, and set $\text{child}(v_\tau) = v_{\tau+1}, \tau = 0, \cdots, T-2$, $\text{child}\, v_{T-1} = v_0$. The cost and value functions are constructed unambiguously.

Tree Growth Once we have $x_{\texttt{sample}} \in \mathbb{X} \setminus \mathbb{X}_{\text{tree}}$, we solve Subproblem 1 for $\mathbb{X}_{\text{target}} = \mathbb{X}_v, v \in \mathcal{V}$. We choose T randomly between 1 and user-defined $T_{\max}$, determined based on computational resources. A heuristic that is useful and is essential in ensuring probabilistic coverage in Sec. 5.4 is replacing $\bar{x}_0 = x_{\texttt{sample}}$ by $\bar{x}_0 = x_{\texttt{sample}} + \eta$ in line 5:, where $\eta_k \in \beta \eta_{\max k} \mathbb{P}$ (subscript k stands for k'th cartesian direction), $\eta_{\max k}$ a positive number such that $\mathbb{X} \subseteq \prod_{k=1}^{n} \eta_{\max k}[-1, 1]$. This heuristic allows moving from the sample for the reward of gaining feasibility/larger polytopes. We allow $\beta \in [0, 1]$ to be selected randomly. Note that we lose the guarantee that $\mathbb{X}_0 \not\subseteq \mathbb{X}_{\text{tree}}$. It suffices to reject solutions if the centroid and (all or some of) vertices of $\mathbb{X}_0^*$ are in $\mathbb{X}_{\text{tree}}$ - rapidly checked using (16) instead of checking the feasibility of $|\mathcal{V}|$ linear programs. As $\mathbb{X}_{\text{tree}}$ gets larger, we can bias β toward smaller values. If for some $v \in \mathcal{V}$ Subproblem 1 is feasible and leads to sufficiently large polytopes, we do not continue computations for the rest of $\mathcal{V}$. We add nodes corresponding to the obtained polytopes to the tree $\mathcal{T}$. The child, cost and value functions are updated accordingly. The procedure is outlined in Algorithm 1. The default criteria in Line 6: is $\text{Vol}(\mathbb{X}_0) > 0$. The nodes in Line 3: are ordered such that the loop is likely terminated early. Given $x_{\texttt{sample}}$ and T, we simulate system (1) with nominal (default choice is zero) control inputs for T time steps to arrive at $x_{\texttt{sample}}^T$ - if $x_{\texttt{sample}}^T$ is out of $\mathbb{X}$, we pick a smaller T. Then we sort nodes in $\mathcal{V}$ in increasing distance from $x_{\texttt{sample}}^T$.

5.3 Rewiring for Optimality

So far, we have not considered cost function (3) in the design of the tree - though it can be combined with Subproblem 1 to obtain trajectories with lower cost. There is no optimality consideration in Algorithm 1, hence it can lead to unnecessarily large cost-to-go values. Inspired by the rewiring procedure in RRT* [12], child function can be modified if it leads to lower cost-to-go values for some nodes according to (14). Due to the space limit, the details are omitted here but we remark that we our framework can accommodate this procedure. Note that unlike RRT*, we do not seek strong asymptotic optimality guarantees. The reason is that we do not sample points from $\mathbb{X}_{tree}$, therefore some necessary rewiring procedures go undetected.

5.4 Probabilistic Feedback Coverage

Similar to LQR-trees, Algorithm 1 achieves probabilistic feedback coverage of the set of all admissible initial states, as stated by the following statement.

Theorem 1. *Let $\mathbb{X}_{\text{tree}}^K$ be the set of states covered by the tree in the K'th iteration in Algorithm 1. Then we have the following with probability 1:*

$$\lim_{K\to\infty} (\mathbb{X}_{initial} \setminus \mathbb{X}_{\text{tree}}^K) = \emptyset. \tag{17}$$

Proof. see appendix and remarks within.

6 Control

Once we obtain the tree, obtaining the controller for Problem 1 is straightforward. In this section, we consider both the cases where the state belongs to $\mathbb{X}_{\text{tree}}$ and not in $\mathbb{X}_{\text{tree}}$ - for the latter we do not have formal guarantees that we can steer the state to $\mathbb{X}_{\text{goal}}$, but we provide heuristics.

6.1 States in $\mathbb{X}_{\text{tree}}$

The following theorem provides the control policy. The proof is immediately followed from the construction in Sec. 4 and the tree structure.

Theorem 2. *Let $\mu_{\mathbb{X}_{\text{tree}}} : \mathbb{X}_{\text{tree}} \to \mathbb{U}$ be the following control policy*

$$\mu_{\mathbb{X}_{\text{tree}}}(x) = \bar{u}_{v^*} + \theta_{v^*} p_{v^*}(x) \tag{18}$$

where $v^* = \arg\min_{v\in V}\{V(v)|x \in \mathbb{X}_v\}$. Then, given $x_0 \in \mathbb{X}_{tree}$, by implementing μ_{tree}, we obtain a trajectory $x_0, \cdots, x_T$, such that $x_\tau \in \mathbb{X}_{tree}, \tau = 0, \cdots, T$, $x_T \in \mathbb{X}_{goal}$, and total cost J in (3) is upper bounded by $V(v^*)$.

In words, the control policy (18) finds the set of polytopes which the current state belongs to, selects the one with the least cost-to-go, and implements the control law to get to its child polytope. All operations required for implementing (18) are basic matrix operations. The polytope search routine can be further computationally optimized using binary search tree techniques that are well-studied for online implementation of PWA control laws [22].

6.2 States in $\mathbb{X} \setminus \mathbb{X}_{tree}$

In case $x \notin \mathbb{X}_{tree}$, we still want to feed the system with some control input - no matter if the state can be steered toward $\mathbb{X}_{\text{goal}}$ or not. We propose the following heuristic: find the control input that steers the state toward $\mathbb{X}_{tree}$ as much as possible. Let v^* be the node in the tree for which the distance to its polytope given as (16) is minimal. We aim for brining the state into child(v^*) by solving the following convex program:

$$\begin{aligned} \mu_{\mathbb{X}\setminus\mathbb{X}_{\text{tree}}}(x) = \arg\min \quad & \|\delta\|_l \\ \text{such that} \quad & x^+ = F(x,u), x^+ + \delta \in \mathbb{X}_{\text{child}(v^*)}, \end{aligned} \tag{19}$$

where $l \in \{1, 2, \infty\}$. Note that if $\delta = 0$ is feasible, then we have succeeded in getting x back into the tree. We do not have any formal guarantees for the policy driven by (19), but we show evidence of its successful implementations via an example in Sec. 7, even when x is very far from $\mathbb{X}_{tree}$. Note that more elaborate formulations of (19) are possible in the price of higher computational cost. For example, we can consider multiple polytopes/steps to search a wider range of get-to-tree possibilities. This is in contrast with the full-blown online hybrid MPC as regions/policies are at least partially computed in advance.

7 Examples

Software We have developed python scripts that are publicly available[1]. Three examples (including two shown in this paper) are currently included. The interested user can define its own problem and use the method. This is an ongoing work and the current software is at its earliest stages.

Example 1 (Bouncing Ball). Consider the following two dimensional PWA system with two modes that models the motion of a ball falling under the gravity and its impacts on the ground - see Fig. 1 [Left]. The control input is acceleration to the ball, which adds to the gravity. The matrices are as follows:

$$A_1 = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, B_1 = \begin{pmatrix} 0 \\ \Delta t \end{pmatrix}, c_1 = \begin{pmatrix} 0 \\ -g\Delta t \end{pmatrix},$$

where other matrices are zero, $\Delta t = 0.025s$ is time-discretization step, $g = 9.8m/s^2$ and x_1 and x_2 denote height and velocity, respectively. Mode 1 is active when $x_1 \geq 0$, and mode 2 is when $x_1 < 0$. The dynamics is double integrator with drift during free flight. Velocity abruptly changes sign after hitting the ground. Note that we slightly abuse the notion of time versus steps in dynamics when the ball hits the ground. We set $\mathbb{U} = \{u | \|u\| \leq 3m/s^2\}$, which indicates that the control actuation is not powerful enough to keep the ball from falling. The goal is to design a feedback strategy and a set of admissible initial conditions such that the ball height and velocity reach $\{(x_1, x_2) | 1m \leq x_1 \leq 1.2m, -0.5m/s \leq x_1 \leq 0.5m/s\}$, while always satisfying the hard constraint $x_2 \in [-5, 5]m/s^2$.

[1] `https://github.com/sadraddini/PWA-Control`

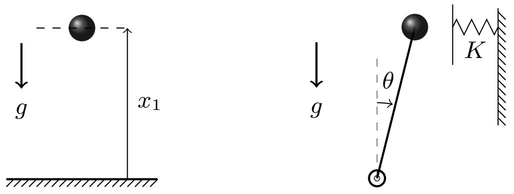

Fig. 1. Examples: [Left] Bouncing Ball [Right] Inverted Pendulum with Wall

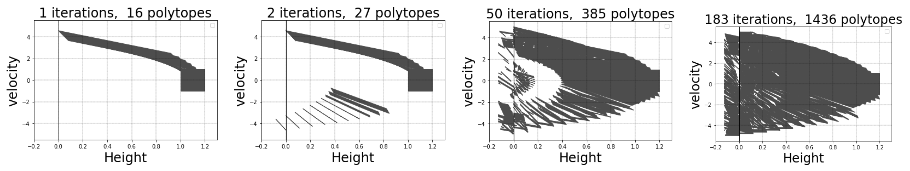

Fig. 2. Example 1: $\mathbb{X}_{\text{tree}}^{K}$ for $K = 1, 2, 50, 183$.

Similarities exist between this example and swinging up inverted pendulum in [21]. Both have nonlinear nature, where the nonlinearity in the pendulum is smooth, but is discontinuous here. In both cases, applying maximal control inputs in the direction of velocity increases energy, which is required to get into the goal. In case of inverted pendulum, stabilization is also involved, which is not the case here. In contrast to the inverted pendulum example in [21], we have hard constraints, and it turns out that $\mathbb{X}_{\text{initial}} \neq \mathbb{X}$.

We used Algorithm 1. The initialization is performed for $T = 16$, where all polytopes belong to mode 1. At the second iteration, the algorithm "discovers" that by using impacts, more states can be driven into the goal. By increasing the number of iterations, nearly all states close to the origin are covered as the number of bounces are increased. Note that finding polytopes becomes more difficult as the space shrinks and polytopes close to the switching surface become small. Snapshots of $\mathbb{X}_{\text{tree}}$ are shown in Fig. 2. We randomly selected 500 points in $\mathbb{X}$ and found 359 of them are in $\mathbb{X}_{\text{tree}}^{183}$. By checking feasibility of (4) for different values of T, we found 416 points are drivable to the goal in less than 80 steps - the tree has covered nearly 86 percent of them. However, when (4) is performed with smaller horizons, the tree has much more coverage. For $T \leq 20$, only 106 of 359 points in $\mathbb{X}_{\text{tree}}$ lead to feasible (4). This number rises to 220 for $T \leq 30$, 311 for $T \leq 40$, 351 for $T \leq 50$, and finally, 359 for $T \leq 60$. The cost-to-go values are shown using different colors in Fig. 3 [Left] for $c_0 = c_1 = 1$ (steps-to-go toward the goal). Two trajectories using control policies in (18) and (19) (implemented on a much lesser grown tree) are shown in Fig. 3 - note the success of the latter.

Example 2 (Inverted Pendulum with Wall). We adopt example 1 from [15]. Consider an inverted pendulum hitting a vertical wall as in Fig. 1 [Right]. The contact model is characterized by linear spring $K = 1000$, and the PWA matrices are:

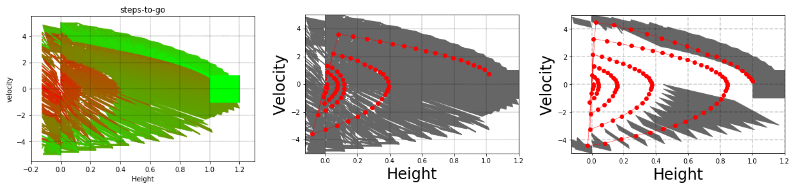

Fig. 3. Example 1: [Left] Steps-to-go. Implementing [Middle] $\mu_{\mathbb{X}_{\text{tree}}}$ [Right] $\mu_{\mathbb{X}\setminus\mathbb{X}_{\text{tree}}}$.

$$A_1 = \begin{pmatrix} 1 & 0.01 \\ 0.1 & 1 \end{pmatrix}, A_2 = \begin{pmatrix} 1 & 0.01 \\ -9.9 & 1 \end{pmatrix}, B_1 = B_2 = \begin{pmatrix} 0 \\ 0.01 \end{pmatrix}, c_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

$c_1 = 0$, $x_1 \in [-0.12, 0.12]$, $x_2 \in [-1, 1]$ and $u \in [-4, 4]$. Mode 1 (free) corresponds to $x_1 \leq 0.1$, and mode 2 (contact) to $x_1 \leq 0.1$. The goal is to steer the state to the origin - a singleton. The authors in [15] precomputed the maximal set of admissible states in mode 1, denoted by Ω, that can be driven into origin using linear feedback. Explicit hybrid MPC was used to drive the other states into Ω. Here, we deliberately ignore exploiting precomputing Ω and entirely rely on our method to find $\mathbb{X}_{\text{initial}}$. Ideally, our method should recover Ω. The tree growth snapshot are shown in Fig. 4. In comparison to the result in [15], illustrated in Fig. 5 [Left], not only we recover most of Ω, but we also find a fairly large set of initial conditions in the top right of the state-space that the method in [15] did not find. The reason is that the MPC horizon was limited to 10 in [15], whereas our method does not directly suffer from short horizons. Physically, these are solutions where pendulum angular velocity is too high in the positive direction for bringing it back to the origin without hitting the wall - making the contact with the wall is necessary. The steps-to-go function is shown in Fig. 5 [Right].

8 Discussion and Future work

A clear disadvantage of the method in this paper versus LQR-trees in [21] is significantly larger number of iterations required to fill the state-space. This issue is mainly due to the discrete-time nature as it leads to discontinuities. For instance, see $\mathbb{X}_{\text{tree}}$ in the second iteration in Fig. 2 has many thin regions that

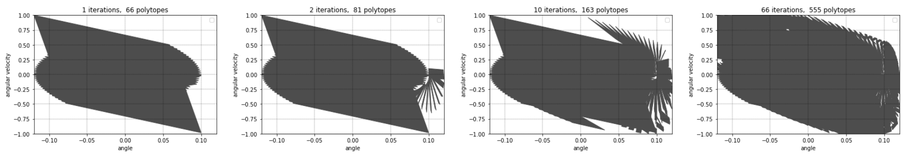

Fig. 4. Example 2: $\mathbb{X}_{\text{tree}}^{K}$ for $K = 1, 2, 10, 66$.

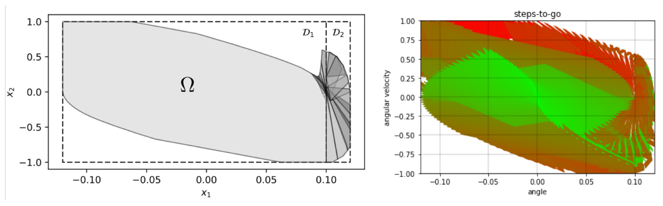

Fig. 5. [Left] The result in [15]. [Right] steps-to-go plot for the result in this work

if are connected in continuous-time, they form thick tubes that cover bigger portions of the state-space. However, discrete-time formulation prevents this - we leave further investigation of this issue to our future work. Second, we restricted all polytopes to be paralleltopes. Polytopes with higher complexity may be more efficient in filling the state-space. However, several issues have to be addressed before relaxing the assumptions made in this paper tailored specifically to paralleltopes. Future work will extend this work to scenarios with higher dimensions, multiple goals, and temporal logic specifications.

Acknowledgements. This work was partially supported by ONR MURI 81825-10911 and the MIT Lincoln Laboratory. We thank Cristian Ioan Vasile and Tobia Marcucci from MIT for fruitful discussions on ideas in this manuscript.

References

1. Bemporad, A., Borrelli, F., Morari, M.: Piecewise linear optimal controllers for hybrid systems. In: American Control Conference, 2000. Proceedings of the 2000, vol. 2, pp. 1190–1194. IEEE (2000)
2. Bemporad, A., Morari, M.: Control of systems integrating logic, dynamics, and constraints. Automatica **35**(3), 407–427 (1999).
3. Branicky, M.S., Curtiss, M.M., Levine, J., Morgan, S.: Sampling-based planning, control and verification of hybrid systems. IEE Proceedings-Control Theory and Applications **153**(5), 575–590 (2006)
4. Collins, S.H., Ruina, A.: A bipedal walking robot with efficient and human-like gait. In: Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pp. 1983–1988. IEEE (2005)
5. De Jong, H., Gouzé, J.L., Hernandez, C., Page, M., Sari, T., Geiselmann, J.: Qualitative simulation of genetic regulatory networks using piecewise-linear models. Bulletin of mathematical biology **66**(2), 301–340 (2004)
6. Deits, R., Tedrake, R.: Footstep planning on uneven terrain with mixed-integer convex optimization. In: Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on, pp. 279–286. IEEE (2014)
7. Dua, V., Pistikopoulos, E.N.: An algorithm for the solution of multiparametric mixed integer linear programming problems. Annals of operations research **99**(1-4), 123–139 (2000)
8. Grizzle, J.W., Chevallereau, C., Sinnet, R.W., Ames, A.D.: Models, feedback control, and open problems of 3d bipedal robotic walking. Automatica **50**(8), 1955–1988 (2014)

9. Grizzle, J.W., Hurst, J., Morris, B., Park, H.W., Sreenath, K.: Mabel, a new robotic bipedal walker and runner. In: American Control Conference, 2009. ACC'09., pp. 2030–2036. IEEE (2009)
10. Han, W., Tedrake, R.: Feedback design for multi-contact push recovery via lmi approximation of the piecewise-affine quadratic regulator. In: n Proceedings of the 2017 IEEE-RAS International Conference on Humanoid Robots, 2017. IEEE-RAS (2017)
11. Hogan, F.R., Rodriguez, A.: Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. arXiv preprint arXiv:1611.08268 (2016)
12. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011)
13. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
14. Manchester, I.R., Tobenkin, M.M., Levashov, M., Tedrake, R.: Regions of attraction for hybrid limit cycles of walking robots. IFAC Proceedings Volumes **44**(1), 5801–5806 (2011)
15. Marcucci, T., Deits, R., Gabiccini, M., Biechi, A., Tedrake, R.: Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In: Humanoid Robotics (Humanoids), 2017 IEEE-RAS 17th International Conference on, pp. 31–38. IEEE (2017)
16. Mehr, N., Sadigh, D., Horowitz, R., Sastry, S.S., Seshia, S.A.: Stochastic predictive freeway ramp metering from signal temporal logic specifications. In: American Control Conference (ACC), 2017, pp. 4884–4889. IEEE (2017)
17. Okamura, A.M., Smaby, N., Cutkosky, M.R.: An overview of dexterous manipulation. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on, vol. 1, pp. 255–262. IEEE (2000)
18. Pratt, J., Carff, J., Drakunov, S., Goswami, A.: Capture point: A step toward humanoid push recovery. In: Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pp. 200–207. IEEE (2006)
19. Rajasekaran, S., Natarajan, R., Taylor, J.D.: Towards planning and control of hybrid systems with limit cycle using lqr trees. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5196–5203 (2017).
20. Raković, S., Kerrigan, E.C., Mayne, D.Q., Kouramas, K.I.: Optimized robust control invariance for linear discrete-time systems: Theoretical foundations. Automatica **43**(5), 831–841 (2007)
21. Tedrake, R., Manchester, I.R., Tobenkin, M., Roberts, J.W.: Lqr-trees: Feedback motion planning via sums-of-squares verification. The International Journal of Robotics Research **29**(8), 1038–1052 (2010)
22. Tøndel, P., Johansen, T.A., Bemporad, A.: Evaluation of piecewise affine control via binary search tree. Automatica **39**(5), 945–950 (2003)
23. Valenzuela, A.K.: Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain. Ph.D. thesis, Massachusetts Institute of Technology (2016)
24. Vandenberghe, L., Boyd, S., Wu, S.P.: Determinant maximization with linear matrix inequality constraints. SIAM journal on matrix analysis and applications **19**(2), 499–533 (1998)
25. Yordanov, B., Tumova, J., Cerna, I., Barnat, J., Belta, C.: Temporal Logic Control of Discrete-Time Piecewise Affine Systems. IEEE Transactions on Automatic Control **57**(6), 1491–1504 (2012). DOI 10.1109/TAC.2011.2178328.

Appendix

Proof for Theorem 1.

Proof. First, observe that $\mathbb{X}_{\text{tree}}^{K}$ monotonically grows: $\mathbb{X}_{\text{tree}}^{K} \subseteq \mathbb{X}_{\text{tree}}^{K+1}$, and is upper bounded by $\mathbb{X}$, so the limit exists. Also note that all sets are closed as Problem 1 is formulated with all sets being compact, and the same holds for Subproblem 1 as all inequalities are non-strict in MICP formulation.

We recall the results of explicit hybrid MPC. We know that given a fixed T, the explicit solution to (4) produces a finite number of polyhedral partitions with affine feedback law in each one [1]. By varying T, the whole $\mathbb{X}_{\text{initial}}$ can be filled with polytopes with affine feedback law in each one. However, there is no claim about the volumes of these polyhedral partitions - some may be less than n-dimensional. In fact, due to hybrid dynamics and discrete-time nature, there is no guarantee that $\mathbb{X}_{\text{initial}}$ is even simply connected. It may consist of disconnected regions of zero Lebesgue measure, for which we do not have means to cover using sampling-based approaches. The key to overcome this issue lies in moving away from $x_{\texttt{sample}}$ to search for feasibility, as highlighted in line 5: in Algorithm 1.

First, we prove that we are able to cover the "full-dimensional neighborhoods" of $\mathbb{X}_{\text{initial}}$, or equivalently prove the following property:

$$\lim_{K\to\infty} \text{Vol}(\mathbb{X}_{\text{initial}} \setminus \mathbb{X}_{\text{tree}}^{K}) = 0. \tag{20}$$

We verify (20) by showing that when $\text{Vol}(\mathbb{X}_{\text{initial}} \setminus \mathbb{X}_{\text{tree}}^{K}) \neq 0$, then $\mathbb{X}_{\text{tree}}^{K+1} \setminus \mathbb{X}_{\text{tree}}^{K}$ has non-zero volume with non-zero probability. First, given $T_{\max}$, let $\mathbb{X}_{\text{initial}}^{T_{\max}}$ be the set of all states that can be driven into $\mathbb{X}_{\text{goal}}$ within $T_{\max}$ steps. First, we prove that (17) holds for $\mathbb{X}_{\text{initial}}^{T_{\max}} \setminus X_{\text{tree}}^{K}$. Then by induction, we prove the argument for any multiplies of $T_{\max}$, and thus $\mathbb{X}_{\text{initial}} \setminus X_{\text{tree}}^{K}$.

When $\text{Vol}(\mathbb{X}_{\text{initial}}^{T_{\max}} \setminus \mathbb{X}_{\text{tree}}^{K}) \neq 0$, then there is a non-zero probability that $x_{\texttt{sample}}$ is selected from $\mathbb{X}_{\text{initial}}^{T_{\max}} \setminus \mathbb{X}_{\text{tree}}^{K}$. Let T be the number of steps that is required to steer $x_{\texttt{sample}}$ into $\mathbb{X}_{\text{goal}}$. Then by non-zero probability, $T \in \{1, \cdots, T_{\max}\}$ is chosen in Line 4:. Now we need to prove that the solution to Subproblem 1 with $\mathbb{X}_{\text{target}} = \mathbb{X}_{\text{goal}}$ returns a sequence of polytopes such that $\mathbb{X}_0$ has non-zero volume with non-zero probability. This fact follows from polyhedral partition of explicit hybrid MPC. We know that the explicit solution to (4) with horizon T produces a finite number of polyhedral partitions with affine feedback law in each one. Therefore, $x_{\texttt{sample}}$ with probability 1 belongs to the interior of one of the polyhedral partitions with non-zero volume. Let it be denoted by $\mathbb{P}_{\texttt{sample}}$. Thus, the solution to Subproblem 1 is guaranteed to produce a non-zero volume $\mathbb{X}_0 \subseteq \mathbb{P}_{\texttt{sample}}$ with positive volume constraint- recall the upper/lower triangular restriction for G_0 mentioned in Sec. 4.4.

For the case $\mathbb{X}_{\text{initial}} \setminus \mathbb{X}_{\text{tree}}^{K}$ we replace $\mathbb{X}_{\text{goal}}$ by polytopes in $\mathbb{X}_{\text{initial}}^{T_{\max}}$ and we arrive a similar argument for $\mathbb{X}_{\text{initial}}^{\kappa T_{\max}}, \kappa = 1, 2, \cdots$, and the rest of the proof follows.

Now we prove that we cover polyhedral partitions that have less than n dimensions by relaxing the requirement that $\mathrm{Vol}(\mathbb{X}_0) > 0$. Let $\mathbb{P}_{<n}$ be such a polytope with dimension $q < n$. There is non-zero probability that $x_{\mathtt{sample}}$ is chosen from η-neighborhood of $\mathbb{P}_{<n}$ such that no other partition is intersecting with this neighborhood. Therefore, when solving Subproblem 1, there is non-zero probability that we "land" on $\mathbb{P}_{<n}$, and obtain a q-dimensional local polytope (claim A) that its intersection with $\mathbb{P}_{<n}$ has non-zero measure in q-dimensional local coordinates, which completes the proof.

(claim A): Our solver for Subproblem 1 is able to find polytopes with maximal dimension. In other words, we are able to maximize the rank of G_0. We already have shown that we can obtain full-rank G_0 with upper/lower triangular restriction. When G_0 can not be full rank, we can re-parametrize it by $G_0 = G^f G^q$, where G^f is a parameterized full rank square matrix and G_q is a given matrix with rank $q < n$ such that G^f exists for q but not for $q+1$ - the value of q can be obtained by line search.

Remark 2. In comparison to [21], we have dropped two key assumptions leading to probabilistic feedback coverage. First, it was assumed in [21] that nonlinear trajectory optimization is always able to find trajectories connecting to the tree, if any exists, with non-zero probability. Since our trajectory optimization method is based on MICP and is complete, we do not require this assumption. Second, it was assumed in [21] that it is always possible to obtain funnels with non-zero volume around any nominal trajectory. This assumption is not reasonable in our setting. But we know from explicit MPC that any "funnel" with maximum dimension, if exists, can already be obtained by local affine control laws, which our controller is already based on.

Remark 3. We have not discussed about the rate of convergence in Theorem 1. In practice, achieving reasonable feedback coverage can be very challenging. Moreover, there is no straightforward guidance to when to switch to polytopes with dimension less than n. We note that such polytopes occur in many interesting problems in contact-based robotics. For example, inelastic contacts lead to dimension reduction in the state of the system.