Optimal Control with Regular Objectives Using an Abstraction-Refinement Approach

Yoke Peng Leong, Pavithra Prabhakar

***Abstract*— This paper presents a method to synthesize a sequence of control inputs for a discrete-time piecewise linear system equipped with a cost function, such that the controlled system behavior satisfies a finite-word linear-time temporal objective with minimal cost. An abstract finite state weighted transition system is constructed, such that the cost of the optimal control on the abstract system provides an upper bound on the cost of the optimal control for the original system. Specifically, the abstract system is constructed from finite partitions of the state and input spaces by solving optimization problems, and a sequence of suboptimal controllers is obtained by considering a sequence of uniformly refined partitions. Furthermore, the costs achieved by the sequence of suboptimal controllers converge to the optimal cost for the piecewise linear system. The abstraction refinement algorithm is implemented in the tool OptCAR. The feasibility of this approach is illustrated on an example, by constructing automatically, sub-optimal controllers with improving optimal costs.**

I. Introduction

Formal synthesis is a paradigm for automatic controller design which are correct-by-construction, reducing the verification overhead. A mathematical model of a system to be controlled and formal specifications of properties that are expected of the controlled system are given as inputs to compute a controller that ensures the system's controlled behavior satisfies the properties. Since the work of Church [1] in automated synthesis, multiple directions are pursued including synthesizing finite state systems with respect to temporal logic objectives [2], [3] and controlling discrete event systems [4]. Early works in hybrid control systems focused on identifying subclasses of systems for which controller synthesis is decidable including timed automata [5], rectangular hybrid automata [6] and o-minimal automata [7], [8]. However, these systems have limited continuous and discrete dynamics, and the synthesis problem becomes undecidable for a relatively simple class of hybrid systems [9].

Abstraction based controller synthesis was introduced as a promising direction [10] for systems with complex dynamics. An abstract model, often a finite state system, is constructed from a given system, such that an abstract model's controller can be refined into a controller for the given system. The abstract model's controller is constructed using automata theory, and then, implemented in the given system. This method has been applied in controller synthesis of switched systems [11], [12], and robotic path planning [13], [14].

Often, in addition to designing a correct controller, an application may require optimality condition. For instance, a robot should reach a desired state with minimum battery. Several previous works have explored optimal controller synthesis using formal approaches [15]–[18].

In this paper, we investigate an abstraction refinement approach to synthesize optimal controller with regular properties that allow for specifications such as reaching a target region or traversing a sequence of regions. Regular properties is specified as a (possibly) infinite set of finite traces interpreted as the allowed behaviors of the system, and generated by a finite state automaton. The foundation of our abstraction refinement procedure and its correctness rely on defining appropriate preorders on the class of hybrid systems which preserve the optimal cost. This paper shows that the preordering defined satisfies the fact that if a system $\mathcal{H}_2$ is higher up in the ordering than a system $\mathcal{H}_1$, then the cost of the optimal controller for $\mathcal{H}_1$ is at most that of $\mathcal{H}_2$.

In our approach, first, an "abstraction" — a simplified finite state system — is constructed from partitions of the state and input spaces, and the edges of the system are annotated with weights which over-approximate the costs in the original system. Then, a two player game on the finite state system is solved to obtain a controller for the abstraction, and subsequently, a controller for the original dynamical system. This approach iteratively consider finer partitions of the state and input spaces, corresponding to grids of size $C/2^i$ for some constant C and $i = 0, 1, 2, \ldots$. In fact, for discrete-time piecewise linear systems, if the cost function is continuous and the optimal control for the original system satisfies a simple assumption, the cost of the sequence of controllers constructed converges to the optimal cost.

The main contribution of this paper is the optimality guarantee on the controllers synthesized, which is lacking in most previous works on optimal controller synthesis using formal approaches [15]–[19]. Our approach is more general than classical finite horizon optimal control problems [20], [21] because the time horizon is not fixed a priori. Furthermore, our method allows for a larger class of cost functions in comparison to previous works [15], [17], [23], and it does not place any prior restriction on the structure of the controllers [18], [22]. Hence, control engineers can synthesize controllers with more flexible structure and cost.

The method introduced in this paper applies to the general class of discrete-time hybrid systems. However, the optimizations that computes the weights depend on the cost function and the dynamics. A prototype tool OptCAR that implements the abstraction refinement algorithm is presented, and it is used to synthesize a (finite) sequence of controllers for a linear piecewise system with reachability objective.

The rest of this paper is organized as follows: Mathematical notations and definitions are presented in Section II. Weighted transition system and its relevant concepts

are defined in Section III, and the preorders for optimal control are explained in Section IV. Section V develops the abstraction and refinement of a weighted transition system. The optimal control problem formulation for piecewise linear systems, the refinement procedure (OptCAR) and the cost analysis are presented in Section VI. Section VII describes the value iteration scheme to compute an optimal strategy for a finite transition system. Implementation of OptCAR is presented in Section VIII. Lastly, Section IX summarizes the paper and states future directions of this work. Due to space constraints, either the proofs are omitted or only sketches of proofs are provided. However, full proofs can be found in the extended version of this paper [24].

II. Notations

The real numbers, non-negative real numbers, integers and non-negative integers are represented as $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$ and $\mathbb{Z}_+$, respectively. The set of integers $\{0, \ldots, k\}$ is written as $[k]$ and a sequence $x_0, \ldots, x_k$, is denoted as $\{x_i\}_{i\in[k]}$.

If $M \in \mathbb{R}^{n\times m}$ is a matrix, $\|M\|_\infty = \max_i \sum_{j=1}^{m} |M_{ij}|$. If $z = (z_0, \ldots, z_k) \in \mathbb{R}^{k+1}$ is a vector, $\|z\|_\infty = \max_t \{|z_t|\}_{t\in[k]}$.

An ϵ-ball around x is defined as $\mathcal{B}_\epsilon(x) = \{x' \in \mathbb{R}^n \mid \|x - x'\|_\infty \leq \epsilon\}$. Let $S \subseteq \mathbb{R}^k$ be a k-dimensional subset. The function *Grid* splits S into rectangular sets with ϵ width. That is, $\mathit{Grid}(S, \epsilon) =$

$$\left\{ S' \;\middle|\; \exists d_1, \ldots, d_k \in \mathbb{Z}, S' = S \bigcap \prod_{i=1}^{k} (d_i\epsilon, (d_i+1)\epsilon] \right\}.$$

Given a function, $f : \mathcal{A} \to \mathcal{B}$, for any $A \subseteq \mathcal{A}$, $f(A) = \{f(a) | a \in A\}$. The domain of a function f is denoted as $\mathit{dom}(f)$. Given an equivalence relation $R \subseteq A \times A$ and an element a, $[a]_R = \{b \,|\, (a, b) \in R\}$ denotes the equivalence class of R containing a.

III. Weighted Transition Systems

This section defines a semantic model for discrete time hybrid systems with cost (i.e., weighted transition systems) and formalizes the optimal control problem.

Definition 1. *A weighted transition system is defined as* $\mathcal{T} = (\mathcal{S}, \mathcal{S}^{init}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W})$*, where:*

- $\mathcal{S}$ *is a set of states;*
- $\mathcal{S}^{init} \subseteq \mathcal{S}$ *is a set of initial states;*
- $\mathcal{U}$ *is a set of control inputs;*
- $\mathcal{P}$ *is a set of propositions;*
- $\Delta \subseteq \mathcal{S} \times \mathcal{U} \times \mathcal{S}$ *is a transition relation;*
- $\mathcal{L} : \mathcal{S} \to \mathcal{P}$ *is a state labeling function, and*
- $\mathcal{W} : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to \mathbb{R}_+$ *is the transition cost function.*

In the sequel, a weighted transition system is referred as a transition system. For any $s \in \mathcal{S}$, define the set $\mathit{Enabled}(s) = \{u \in \mathcal{U} \mid \exists s' \in \mathcal{S} \; s.t. \; (s, u, s') \in \Delta\}$ to represent all inputs that do not transitions the state s out of the predefined set $\mathcal{S}$. A transition system is *finite* if $\mathcal{S}$ and $\mathcal{U}$ are finite. A finite state *automaton* (denoted $(\mathcal{T}, P_f)$) is a finite transition system $\mathcal{T}$ along with a proposition $P_f \in \mathcal{P}$ which represents the final states. For the rest of the section, fix the transition system $\mathcal{T} = (\mathcal{S}, \mathcal{S}^{init}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W})$.

a) Paths and traces: A *path* of the transition system $\mathcal{T}$ is a sequence of states and inputs, $\zeta = s_0u_0s_1u_1s_2\ldots$, where $s_0 \in \mathcal{S}^{init}$, $s_i \in \mathcal{S}$, $u_i \in \mathcal{U}$, and $(s_i, u_i, s_{i+1}) \in \Delta$. The set of all finite paths of $\mathcal{T}$ is denoted $\mathit{Paths}(\mathcal{T})$. A trace of a transition system is the sequence of state labels of a path. The trace of ζ, denoted $\mathit{Tr}(\zeta)$, is the sequence $\mathcal{L}(s_0)\mathcal{L}(s_1)\ldots$.

b) Properties: A *property* describes the desired behaviors of the system. This paper focuses on linear time properties over finite behaviors of systems. Formally, a property Π over a set of propositions $\mathcal{P}$ is a set of finite sequences $\pi = p_0p_1\ldots p_k$, where each $p_i \in \mathcal{P}$.

A property is *regular* if it consists of the traces of a finite state automaton $(\mathcal{T}, P_f)$, the set of all traces of paths of $\mathcal{T}$ which start in an initial state and end in a state labelled by P_f. This paper considers regular property that is specified by a finite state automaton $(\mathcal{T}, P_f)$. The properties expressed by popular logics such as finite words linear-time temporal logic (LTL) are regular, but their translations into finite transition system representations can lead to an exponential blow up in the number of states with respect to the formula size [25].

c) Strategies: A strategy specifies the control inputs to a transition system. Specifically, a *strategy* σ for the transition system $\mathcal{T}$ is a partial function $\sigma : \mathit{Paths}(\mathcal{T}) \to \mathcal{U}$ such that for a path $\zeta = s_0u_0s_1\ldots u_{i-1}s_i$, $\sigma(\zeta) \in \mathit{Enabled}(s_i)$. A path $\zeta = s_0u_0s_1u_1s_2\ldots$ is said to *conform* to a strategy σ, if for all i, $\sigma(s_0u_0\ldots s_i) = u_i$.

A finite path $\zeta = s_0u_0\ldots s_k$ maximally conforms to a strategy σ, if ζ conforms to σ and there is no extension $\zeta' = s_0u_0\ldots s_ku_ks_{k+1}$ of ζ which conforms to σ. Let $\mathit{Paths}^m_\sigma(\mathcal{T}, s_0)$ denotes the maximally conforming finite paths of $\mathcal{T}$ with respect to σ starting at a state s_0. Let $\mathit{Str}(\mathcal{T})$ denote the set of all strategies which have no infinite paths conforming to them. Note that the length of the paths which conforms to a strategy in $\mathit{Str}(\mathcal{T})$ could still be unbounded.

To synthesize a strategy for $\mathcal{T}$ from $s_0 \in \mathcal{S}^{init}$ such that all maximal executions conforming to it reach a state in $\mathcal{S}_f \subseteq \mathcal{S}$, label the states in $\mathcal{S}_f$ with a unique proposition. Then, let the property Π be the set of all traces for paths which start in $\mathcal{S}^{init}$ and end in $\mathcal{S}_f$, and do not visit $\mathcal{S}_f$ in the middle.

Definition 2. *A strategy* σ *for the transition system* $\mathcal{T}$ *and an initial state* $s_0 \in \mathcal{S}$ *is* ***winning*** *with respect to property* Π *over the propositions* $\mathcal{P}$*, if* $\sigma \in \mathit{Str}(\mathcal{T})$ *and* $\mathit{Tr}(\mathit{Paths}^m_\sigma(\mathcal{T}, s_0)) \subseteq \Pi$.

d) Cost of strategies: The cost of a path is the sum of the weights on the individual edges. Given a path $\zeta = s_0u_0s_1\ldots$, define:

$$\mathcal{W}(\zeta) = \sum_j \mathcal{W}(s_j, u_j, s_{j+1}).$$

Consequently, the following proposition holds.

Proposition 1. *Given* $\zeta = s_0u_0s_1\ldots s_k$ *and* $\zeta' = s'_0u'_0 s'_1\ldots s'_k$*, if* $\mathcal{W}(s_j, u_j, s_{j+1}) \leq \mathcal{W}(s'_j, u'_j, s'_{j+1})$ *for all* j*, then* $\mathcal{W}(\zeta) \leq \mathcal{W}(\zeta')$.

This monotonicity property seems trivial, but plays an important role in later analysis. In fact, results in this paper carry over for several other cost functions such as average weight and maximum weight. Over infinite paths, average cost, maximum cost or discounted sum are more natural. The analysis only relies on the fact that the cost of a path is monotonic with respect to the cost on the transitions. For simplicity, we fix one of the definitions.

The cost of a strategy σ of the transition system $\mathcal{T}$ with respect to an initial state s_0 is defined as

$$\mathcal{W}(\mathcal{T}, \sigma, s_0) = \sup\{\mathcal{W}(\zeta) \,|\, \zeta \in \mathit{Paths}^m_\sigma(\mathcal{T}, s_0)\}.$$

Accordingly, given a property Π over $\mathcal{P}$, the optimal cost of winning $\mathcal{T}$ from an initial state s_0 with respect to a property Π is defined as $\mathcal{W}(\mathcal{T}, s_0, \Pi) =$

$$\inf\{\mathcal{W}(\mathcal{T}, \sigma, s_0) \,|\, \sigma \in \mathit{Str}(\mathcal{T}), \mathit{Tr}(\mathit{Paths}^m_\sigma(\mathcal{T}, s_0)) \subseteq \Pi\}.$$

The cost is taken to be infinity if the minimization is over an empty set. Denote an optimal strategy that achieves the optimal cost as $\sigma(\mathcal{T}, s_0, \Pi)$. Note that the optimal strategy may not be unique or exist.

e) Optimal control problem: Given the transition system $\mathcal{T}$, an initial state s_0 and a property Π, the optimal control problem is to compute an optimal winning strategy from s_0 with respect to Π, if it exists, and the optimal cost of winning.

IV. Preorders for optimal control

In this section, a preorder on the class of transition systems is defined such that it preserves the optimal cost of winning. In other words, the optimal cost of winning in a system higher up in the ordering is an upper bound on the optimal cost of winning in a system below it. For this, the definition of alternating simulations [26] is extended to include costs.

Definition 3. *Given two transition systems $\mathcal{T}_i = (\mathcal{S}_i, \mathcal{S}_i^{init}, \mathcal{U}_i, \mathcal{P}, \Delta_i, \mathcal{L}_i, \mathcal{W}_i)$, for $i = 1, 2$, a simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$ is a pair of relations (α, β), where $\alpha \subseteq \mathcal{S}_1 \times \mathcal{S}_2$ and $\beta \subseteq \mathcal{S}_1 \times \mathcal{U}_1 \times \mathcal{S}_2 \times \mathcal{U}_2$, such that:*

1) $\forall\ (s_1, s_2) \in \alpha$, $\mathcal{L}_1(s_1) = \mathcal{L}_2(s_2)$.
2) $\forall\ s_1 \in \mathcal{S}_1^{init}$, $\exists\ s_2 \in \mathcal{S}_2^{init}$ *such that* $(s_1, s_2) \in \alpha$;
3) $\forall\ (s_1, s_2) \in \alpha$ *and* $u_2 \in \mathit{Enabled}(s_2)$, $\exists\ u_1 \in \mathit{Enabled}(s_1)$ *such that:*
   a) $(s_1, u_1, s_2, u_2) \in \beta$
   b) $\forall\ (s_1, u_1, s_1') \in \Delta_1$, $\exists\ (s_2, u_2, s_2') \in \Delta_2$ *such that* $(s_1', s_2') \in \alpha$ *and* $\mathcal{W}_1(s_1, u_1, s_1') \leq \mathcal{W}_2(s_2, u_2, s_2')$.

Let $\mathcal{T}_1 \preceq_{(\alpha,\beta)} \mathcal{T}_2$ denote that (α, β) is a simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$. If there exists some (α, β) such that $\mathcal{T}_1 \preceq_{(\alpha,\beta)} \mathcal{T}_2$, then $\mathcal{T}_2$ simulates $\mathcal{T}_1$, and it is denoted as $\mathcal{T}_1 \preceq \mathcal{T}_2$.

Theorem 2. *$\preceq$ is a preorder on the class of transition systems over a set of propositions $\mathcal{P}$.*

Proof. Define (α, β) to be identity relations on the state and input spaces, then $\mathcal{T} \preceq_{(\alpha,\beta)} \mathcal{T}$, and $\preceq$ is reflexive. Next, show that $\preceq$ is transitive. Let $\mathcal{T}_1 \preceq_{(\alpha_1,\beta_1)} \mathcal{T}_2$ and $\mathcal{T}_2 \preceq_{(\alpha_2,\beta_2)} \mathcal{T}_3$. Define α such that $(s_1, s_3) \in \alpha$ if $(s_1, s_2) \in \alpha_1$ and $(s_2, s_3) \in \alpha_2$ for some s_2, and β such that $(s_1, u_1, s_3, u_3) \in \beta$ if $(s_1, u_1, s_2, u_2) \in \beta_1$ and $(s_2, u_2, s_3, u_3) \in \beta_2$ for some (s_2, u_2). Then, $\mathcal{T}_1 \preceq_{(\alpha,\beta)} \mathcal{T}_3$. □

The next result shows that $\preceq$ is an ordering on the transition systems which "preserves" optimal control.

Theorem 3. *Given two transition systems $\mathcal{T}_i = (\mathcal{S}_i, \mathcal{S}_i^{init}, \mathcal{U}_i, \mathcal{P}, \Delta_i, \mathcal{L}_i, \mathcal{W}_i)$ for $i = 1, 2$, let Π be a property over a set of propositions $\mathcal{P}$, $\mathcal{T}_1 \preceq_{(\alpha,\beta)} \mathcal{T}_2$ and $(s_0, s_0') \in \alpha$ for $s_0 \in \mathcal{S}_1^{init}$ and $s_0' \in \mathcal{S}_2^{init}$. If a winning strategy σ_2 for $\mathcal{T}_2$ from s_0' with respect to Π exists, then a winning strategy σ_1 for $\mathcal{T}_1$ from s_0 with respect to Π exists such that $\mathcal{W}_1(\mathcal{T}_1, \sigma_1, s_0) \leq \mathcal{W}_2(\mathcal{T}_2, \sigma_2, s_0')$. Hence, $\mathcal{W}_1(\mathcal{T}_1, s_0, \Pi) \leq \mathcal{W}_2(\mathcal{T}_2, s_0', \Pi)$.*

Proof. Let σ_2 be a strategy for $\mathcal{T}_2$ and s_0'. In addition, define a partial mapping $G : \mathit{Paths}(\mathcal{T}_1) \to \mathit{Paths}(\mathcal{T}_2)$ such that the domain of G is the set of all paths from s_0 that conform to σ_1, and for any path ζ_1 in the domain of G, $\mathcal{L}_1(\zeta_1) = \mathcal{L}_2(G(\zeta_1))$, and $\mathcal{W}_1(\zeta_1) \leq \mathcal{W}_2(G(\zeta_1))$. This construction ensures that if σ_2 is winning from s_0' with respect to Π, then so is σ_1 from s_0 and $\mathcal{W}_1(\mathcal{T}_1, \sigma_1, s_0) \leq \mathcal{W}_2(\mathcal{T}_2, \sigma_2, s_0')$. We also ensure that if $G(\zeta_1) = \zeta_2$, then $(s_k, s_k') \in \alpha$, where s_k and s_k' are the end states of ζ_1 and ζ_2, respectively. Further, for any ζ_1 in the domain of G, ζ_1 is a maximal path conforming to σ_1 if and only if ζ_2 is a maximal path conforming to ζ_2.

Next, define σ_1 and G by induction on the length of words in their domain. Set $G(s_0) = s_0'$. Suppose σ_1 for paths of length $k-1$ and G for paths of length k, are defined such that the invariant holds. Let $\zeta_1 = s_0 u_0 s_1 \dots s_k$ conform to σ_1. Then, $G(\zeta_1)$ is defined. Let $G(\zeta_1) = s_0' u_0' s_1' \dots s_k'$ and $(s_k, s_k') \in \alpha$. If $G(\zeta_1)$ is a maximal path conforming to σ_2, then $\sigma_1(\zeta_1)$ is not defined (i.e., ζ_1 is not in the domain of σ_1). Otherwise $\sigma_2(G(\zeta_1)) = u_k'$. Then, from the second condition of simulation, there exists u_k such that $(s_k, u_k, s_k', u_k') \in \beta$. Choose $\sigma_1(\zeta_1) = u_k$. For any $\zeta_2 = s_0 u_0 s_1 \dots s_{k+1}$, define $G(\zeta_2) = s_0' u_0' s_1' \dots s_{s+1}'$ such that $(s_{k+1}, s_{k+1}') \in \alpha$ and $\mathcal{W}_1(s_k, u_k, s_{k+1}) \leq \mathcal{W}(s_k', u_k', s_{k+1}')$. It can be verified that the construction satisfies the inductive invariant. □

V. Abstraction/Refinement

In this section, a method for constructing finite state systems which simulate a given transition system is presented. The state and input spaces are divided into finite number of parts, and they are used as symbolic states and inputs, respectively, in the abstract transition system. Henceforth, fix a transition system $\mathcal{T} = (\mathcal{S}, \mathcal{S}^{init}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W})$.

A. Abstraction

An abstraction function constructs an abstract transition system $\mathit{Abs}(\mathcal{T}, \equiv_S, \equiv_U)$ given the transition system $\mathcal{T}$, and two equivalence relations $\equiv_S$ and $\equiv_U$ on the state-space $\mathcal{S}$ and the input-space $\mathcal{U}$, respectively. An equivalence relation $\equiv_S$ on $\mathcal{S}$ respects $\mathcal{L}$, if for all $(s_1, s_2) \in \equiv_S$, $\mathcal{L}(s_1) = \mathcal{L}(s_2)$. Furthermore, an equivalence relation $\equiv_S$ on $\mathcal{S}$ respects $\mathcal{S}^{init}$, if for all $(s_1, s_2) \in \equiv_S$ where $s_1 \in \mathcal{S}^{init}$, $s_2 \in \mathcal{S}^{init}$.

Definition 4. *Let $\equiv_S \subseteq \mathcal{S} \times \mathcal{S}$ and $\equiv_U \subseteq \mathcal{U} \times \mathcal{U}$ be two equivalence relations of finite index such that $\equiv_S$ respects the labeling function $\mathcal{L}$ and the initial states $\mathcal{S}^{init}$. $Abs(\mathcal{T}, \equiv_S, \equiv_U) = (\mathcal{S}', \mathcal{S}^{init'}, \mathcal{U}', \mathcal{P}, \Delta', \mathcal{L}', \mathcal{W}')$, where:*

- $\mathcal{S}' = \{[s]_{\equiv_S} \mid s \in \mathcal{S}\}$ *is the equivalence classes of* $\equiv_S$.
- $\mathcal{S}^{init'} = \{[s]_{\equiv_S} \mid s \in \mathcal{S}^{init}\} \subseteq \mathcal{S}'$.
- $\mathcal{U}' = \{[u]_{\equiv_U} \mid u \in \mathcal{U}\}$ *is the equivalence classes of* $\equiv_U$.
- $\Delta' = \{(S_1, U, S_2) \mid \exists s \in S_1, s' \in S_2, u \in U,\ s.t.\ (s, u, s') \in \Delta\}$.
- *For* $S \in \mathcal{S}'$, $\mathcal{L}'(S) = \mathcal{L}(s)$ *for any* $s \in S$.
- *For* $(S_1, U, S_2) \in \Delta'$, $\mathcal{W}'(S_1, U, S_2) = \sup\{\mathcal{W}(s_1, u, s_2) \mid s_1 \in S_1, s_2 \in S_2, u \in U, (s_1, u, s_2) \in \Delta\}$.

Call $\mathcal{T}$ the concrete system and $Abs(\mathcal{T}, \equiv_S, \equiv_U)$ the abstract system. The next proposition states that the abstract system simulates the concrete system.

Proposition 4. $\mathcal{T} \preceq Abs(\mathcal{T}, \equiv_S, \equiv_U)$.

Proof. Define $\alpha = \{(s, [s]_{\equiv_S}) \mid s \in \mathcal{S}\}$, and $\beta = \{(s, u, [s]_{\equiv_S}, [u]_{\equiv_U}) \mid s \in \mathcal{S} \text{ and } u \in \mathcal{U}\}$. □

B. Refinement

We can construct a sequence of abstract systems which are closer to the original system than their predecessors in the sequence, by choosing finer equivalence relations on the state and input spaces.

Definition 5. *Let $\mathcal{T}_1$ and $\mathcal{T}_3$ be transition systems such that $\mathcal{T}_1 \preceq \mathcal{T}_3$. A transition system $\mathcal{T}_2$ is said to be a refinement of $\mathcal{T}_3$ with respect to $\mathcal{T}_1$, if $\mathcal{T}_1 \preceq \mathcal{T}_2 \preceq \mathcal{T}_3$.*

Proposition 5. *Let $\equiv_S, \equiv'_S \subseteq \mathcal{S} \times \mathcal{S}$ and $\equiv_U, \equiv'_U \subseteq \mathcal{U} \times \mathcal{U}$ be equivalence relations of finite index such that $\equiv'_S \subseteq \equiv_S$ and $\equiv'_U \subseteq \equiv_U$. Then, $Abs(\mathcal{T}, \equiv'_S, \equiv'_U)$ is a refinement of $Abs(\mathcal{T}, \equiv_S, \equiv_U)$ with respect to $\mathcal{T}$.*

Proof. First, $\mathcal{T} \preceq Abs(\mathcal{T}, \equiv'_S, \equiv'_U)$ follows from Proposition 4. Define $\alpha = \{([s]_{\equiv_{S'}}, [s]_{\equiv_S}) \mid s \in \mathcal{S}\}$ and $\beta = \{([s]_{\equiv_{S'}}, [u]_{\equiv'_U}, [s]_{\equiv_S}, [u]_{\equiv_U}) \mid s \in \mathcal{S} \text{ and } u \in \mathcal{U}\}$. Then, properties in Definition 3 are satisfied for $Abs(\mathcal{T}, \equiv'_S, \equiv'_U) \preceq_{(\alpha,\beta)} Abs(\mathcal{T}, \equiv_S, \equiv_U)$, and thus, $\mathcal{T} \preceq Abs(\mathcal{T}, \equiv'_S, \equiv'_U) \preceq Abs(\mathcal{T}, \equiv_S, \equiv_U)$. □

VI. Optimal Control of Piecewise Linear Systems

This section considers an optimal control problem for discrete-time piecewise linear systems. The abstraction refinement approach is applied to construct a series of controllers with improving suboptimal costs that converge to the optimal cost under a simple condition on the optimal control.

A. Problem Formulation

A discrete-time piecewise linear system is a tuple $(\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i\in[m]}, \mathcal{L}, \mathcal{J})$, where the state-space $\mathcal{X} \subseteq \mathbb{R}^n$ and the input-space $\mathcal{U} \subseteq \mathbb{R}^p$ are compact sets, $\mathcal{X}^{init} \subseteq \mathcal{X}$ is the set of initial states, $\mathcal{P}$ is a finite set of propositions, $A_i \in \mathbb{R}^{n\times n}, B_i \in \mathbb{R}^{n\times p}$ and P_i is a polyhedral set, such that $\{P_i\}_{i\in m}$ is a polyhedral partition of $\mathcal{X}$, $\mathcal{L} : \mathcal{X} \to \mathcal{P}$ a labeling function and $\mathcal{J} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_+$ is a continuous cost function. Note that A_i and B_i can be the same for different i. Given an initial state $x_0 \in \mathcal{X}^{init}$ and a sequence of control inputs $\mathbf{u} = \{u_t\}_{t\in[k]}$, where $u_t \in \mathcal{U}$, $\phi(x_0, \mathbf{u}) = \{x_t\}_{t\in[k+1]}$ is the sequence of states visited under the control $\mathbf{u}$, where $x_{t+1} = A_t x_t + B_t u_t$, and $(A_t, B_t) = (A_i, B_i)$ if $x_t \in P_i$. The cost of the sequence $\phi(x_0, \mathbf{u})$, $\mathcal{J}(\phi(x_0, \mathbf{u}))$, is given by $\sum_{t\in[k]} \mathcal{J}(x_{t+1}, u_t)$.

Problem 1 (Optimal control problem)**.**
Given an n-dimensional discrete-time piecewise linear system $\mathcal{D} = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i\in[m]}, \mathcal{L}, \mathcal{J})$, a state $x_0^* \in \mathcal{X}^{init}$ and a regular property Π over $\mathcal{P}$, find a sequence of control inputs $\mathbf{u}^*$ for which $\mathcal{L}(\phi(x_0^*, \mathbf{u}^*)) \in \Pi$ and $\mathcal{J}(\phi(x_0^*, \mathbf{u}^*))$ is minimized.

Remark 1. *Although the property Π is over finite sequences, it could potentially contain finite sequences of unbounded length. Thus, Problem 1 is not the same as a classical finite horizon problem, because the optimal control sequence length is not fixed a priori.*

B. Solution

A discrete-time piecewise linear system $\mathcal{D} = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \{(A_i, B_i, P_i)\}_{i\in[m]}, \mathcal{L},\ \mathcal{J})$ can be represented as a weighted transition system, $\mathcal{T}_\mathcal{D} = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W})$ where $\Delta = \{(x_t, u_t,\ x_{t+1}) \in \mathcal{X} \times \mathcal{U} \times \mathcal{X} \mid x_{t+1} = A_t x_t + B_t u_t,\ \text{where}\ (A_t, B_t) = (A_i, B_i) \text{ if } x_t \in P_i\}$, and $\mathcal{W}(x_t, u_t, x_{t+1}) = \mathcal{J}(x_{t+1}, u_t)$ for all $t \in [k]$. Consequently, Problem 1 is equivalent to the following problem:

Problem 2 (Optimal strategy problem)**.**
Given $\mathcal{T}_\mathcal{D} = (\mathcal{X}, \mathcal{X}^{init}, \mathcal{U}, \mathcal{P}, \Delta,\ \mathcal{L}, \mathcal{W})$, a state $x_0^* \in \mathcal{X}^{init}$ and a regular property Π over $\mathcal{P}$, find an optimal winning strategy $\sigma(\mathcal{T}_\mathcal{D}, x_0^*, \Pi)$ for which the optimal cost of winning $\mathcal{T}_\mathcal{D}$ with respect to Π, $\mathcal{W}(\mathcal{T}_\mathcal{D}, x_0^*, \Pi)$, is achieved.

In general, solving Problem 2 is difficult; hence, we focus on synthesizing suboptimal strategies using Algorithm 1. It first partitions the state space into grids of a particular size, and constructs an abstract system for $\mathcal{T}_\mathcal{D}$. Then, it computes the optimal cost J and the abstract system's strategy through a two-player game. A suboptimal strategy for $\mathcal{T}_\mathcal{D}$ can then be extracted from the abstract system's strategy with the cost upper bounded by J. If J does not satisfy the predefined cost requirement, refine the state space partitions using smaller grids, and repeat the whole process to reduce J. As a result, this algorithm outputs a sequence of suboptimal strategies, whose costs converge to that of the optimal cost.

More precisely, in each iteration, Algorithm 1 first constructs a finite state abstraction $\hat{\mathcal{T}}$ of $\mathcal{D}$ using the function $ConsAbs(\mathcal{D}, \epsilon)$. $ConsAbs(\mathcal{D}, \epsilon)$ outputs $Abs(\mathcal{T}_\mathcal{D}, \equiv^\epsilon_X, \equiv^\epsilon_U)$, where $\equiv^\epsilon_X$ and $\equiv^\epsilon_U$ are equivalence relations whose equivalences classes are the elements of $Grid(\mathcal{X}, \epsilon)$ and $Grid(\mathcal{U}, \epsilon)$, respectively. The initial abstract state $\hat{x}_0 := [x_0^*]_{\equiv^\epsilon_X}$. Next, $SolveFiniteGame(\hat{\mathcal{T}}, \hat{x}_0, \Pi)$ computes the optimal cost of winning $J = \mathcal{W}(\hat{\mathcal{T}}, \hat{x}_0, \Pi)$ with respect to Π in $\hat{\mathcal{T}}$ and the corresponding strategy $\hat{\sigma} = \sigma(\hat{\mathcal{T}}, \hat{x}_0, \Pi)$ for $\hat{\mathcal{T}}$ through a two-player game (see Algorithm 2 of Section VII for more details). Finally, $Extract(\hat{\sigma}, \hat{\mathcal{T}}, \mathcal{D})$ outputs a suboptimal

Algorithm 1 OptCAR (Abstraction Refinement Procedure)

Require: System $\mathcal{D}$, Property Π as a finite state automaton, initial state x_0^*, rational number $0 < \epsilon_0$

Set $\epsilon := \epsilon_0$

while true **do**

$\hat{\mathcal{T}}, \hat{x}_0 := \mathit{ConsAbs}(\mathcal{D}, \epsilon)$

$J, \hat{\sigma} := \mathit{SolveFiniteGame}(\hat{\mathcal{T}}, \hat{x}_0, \Pi)$

$\sigma_{\mathcal{D}} := \mathit{Extract}(\hat{\sigma}, \hat{\mathcal{T}}, \mathcal{D})$

Output $\sigma_{\mathcal{D}}$ and J

$\epsilon := \frac{\epsilon}{2}$

end while

strategy $\sigma_{\mathcal{D}}$ whose cost is bounded by the optimal cost J for the abstract system. The existence of $\sigma_{\mathcal{D}}$ given $\hat{\sigma}$ is guaranteed by Theorem 3. Essentially, $\sigma_{\mathcal{D}}$ provides the sequence of inputs u^* as required by Problem 1.

To illustrate the relationship between $\sigma_{\mathcal{D}}$ and $\hat{\sigma}$, let $u_0^*, u_1^*, \ldots, u_{t-1}^*$ be the inputs which have been computed, and let $s_0^*, s_1^*, \ldots, s_t^*$ be the sequence of state generated by the inputs. The t-th control input u_t^* is obtained by finding the minimum cost transition $(s_t^*, u_t^*, s_{t+1}^*)$, where $u_t^* \in U$, $s_{t+1}^* \in S'$, $U = \hat{\sigma}([s_0^*]_{\equiv_X^\epsilon}[u_0^*]_{\equiv_U^\epsilon} \cdots [s_t^*]_{\equiv_X^\epsilon})$, and S' is the union of all S'' such that $([s_t^*]_{\equiv_X^\epsilon}, U, S'')$ is a transition of $\hat{\mathcal{T}}$. When the cost function is linear and the equivalence classes are polyhedral sets, u_t^* is computed by solving a linear program.

Initially, when the partitioning is coarse, a winning strategy $\hat{\sigma}$ might not exist even if the underlying system $\mathcal{D}$ has an optimal solution. However, if one continues to refine the grid, a winning strategy will exist if $\mathcal{D}$ has an optimal solution, and its cost of winning will converge to the optimal cost. See Section VI-C for the proof. The algorithm can be terminated at a specific iteration based on applications and computational resources.

Algorithm 1 can in fact be instantiated to any class of hybrid systems. However, the computational complexity of the optimization problems that will need to be solved in the construction of the abstract system and the extraction of a winning strategy will depend on the class of dynamics and the type of the cost function. For a piecewise linear system with linear cost function, the maximization during the abstraction procedure is a linear program, because the partitions of $\equiv_X$ and $\equiv_U$ are polyhedral sets (grid elements).

C. Analysis of Algorithm 1

Next, the output of Algorithm 1 is shown to converge to the optimal cost. Before that, we state an assumption we make on the solution of Problem 1.

Assumption 1. *A solution $\mathbf{u}^*$ exists for Problem 1 such that in the sequence $\phi(x_0^*, \mathbf{u}^*) = \{x_t^*\}_{t\in[k+1]}$, for every $t \in [k+1]$, there exists $\epsilon_t > 0$ such that $\mathcal{B}_{\epsilon_t}(x_t^*) \subseteq P_i$, where P_i is the polyhedral set in $\mathcal{D}$ containing x_t^*. Define $\epsilon_T = \min_{t\in[k+1]} \epsilon_t$.*

This assumption ensures that there exists a control input sequence such that none of the states reached by it lie on the boundary of a region P_i. Notice that the boundaries of the partitions form a set of measure zero. Hence, if an optimal solution exists, the solution is very likely to satisfy Assumption 1. Let us denote the elements in the iteration of Algorithm 1 corresponding to a particular ϵ as $\hat{\mathcal{T}}_\epsilon$ for $\hat{\mathcal{T}}$, $\hat{x}_0^\epsilon$ for $\hat{x}_0$, J_ϵ for J, $\hat{\sigma}_\epsilon$ for $\hat{\sigma}$ and σ_ϵ for $\sigma_{\mathcal{D}}$, respectively.

Henceforth, let $\mathbf{u}^* = \{u_t^*\}_{t\in[k]}$ be an optimal control input sequence and $\zeta^* = \phi(x_0^*, \mathbf{u}^*) = \{x_t^*\}_{t\in[k+1]}$ be an optimal trajectory for Problem 1 which satisfies Assumption 1. Later, the proof of Theorem 6 requires a special kind of strategy which chooses a unique transition on the control input from any path that conforms to the strategy.

Definition 6. *A **chain strategy** for a transition system $\mathcal{T}$ and an initial state s_0 is a strategy $\sigma \in Str(\mathcal{T})$ such that there is one path in $Paths_\sigma^m(\mathcal{T}, s_0)$.*

Theorem 6. *Under Assumption 1, the sequence of sub-optimal costs $\{J_{\epsilon_0/2^i}\}_{i\in\mathbb{Z}_+}$ output by Algorithm 1 converge to the optimal cost $J_{opt} = \mathcal{W}(\mathcal{T}_{\mathcal{D}}, x_0, \Pi)$. Furthermore, for each sub-optimal cost $J_{\epsilon_0/2^i}$, there exists a suboptimal winning strategy $\sigma_{\epsilon_0/2^i}$ with cost of winning $J_{\epsilon_0/2^i}$.*

Proof. A sketch of the proof is provided.

First, let $\epsilon_T > \epsilon_x > 0$ and $\epsilon_u > 0$. For any trajectory whose initial state and inputs have a bounded deviation from that of the optimal trajectory, the error between any of those trajectories is bounded. Let $x_0 \in \mathcal{B}_{\epsilon_x}(x_0^*)$ and $u_t \in \mathcal{B}_{\epsilon_u}(u_t^*)$ $\forall t \in [k]$, where $\mathbf{u} = \{u_t\}_{t\in[k]}$ and $\zeta = \phi(x_0, \mathbf{u}) = \{x_t\}_{t\in[k+1]}$. Then, for all $t \in [k]$, $\left\|x_{t+1} - x_{t+1}^*\right\|_\infty \leq c_1\epsilon_x + c_2\epsilon_u$ where $c_1 \geq 0$ and $c_2 \geq 0$ are constants that depends only on A_i, B_i, and t. In addition, given the cost function in Problem 2, $|\mathcal{W}(\zeta) - \mathcal{W}(\zeta^*)| \leq c_3\epsilon_x + c_4\epsilon_u$ where $c_2 \geq 0$ and $c_4 \geq 0$ are constants that does not depend on ϵ_x and ϵ_u. In other words, because the cost function is continuous, the suboptimal cost is bounded and decreases to zero if ϵ_x and ϵ_u decreases to zero.

Next, given a specific cost sub-optimality, a chain strategy that satisfies a certain cost error bound exists. More specifically, a neighborhood N_t^x is constructed around each x_t^* and N_t^u around u_t^* such that all the transitions from N_t^x on N_t^u will end in N_{t+1}^x. This construction exists by Assumption 1 and that the system is linear (continuous). Under this construction, all executions from N_0^x will be in N_t^x after t steps. This chain of neighborhoods gives a chain strategy. See Figure 1 for an illustration of the chain strategy. To

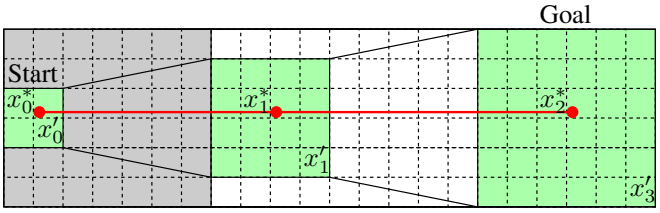


Fig. 1. An illustration of chain strategy and refinement. The domain is separated into two areas (gray and white) where two different dynamics apply. The red dots are the optimal path.

ensure that the cost of the strategy is within $\delta > 0$ of the optimal cost, choose the N_t^x and N_t^u to be contained in some $\epsilon_T > \epsilon_x > 0$ and $\epsilon_u > 0$ balls such that $c_3\epsilon_x + c_4\epsilon_u \leq \delta$. These two observations ensure that $|\mathcal{W}(\zeta) - \mathcal{W}(\zeta^*)| \leq \delta$ for any path ζ starting in an ϵ_x ball around x_0^*.

In addition, choose the N_j^x and N_j^u in such a way that they correspond to an element of an $\epsilon_0/2^i$ grid for some i (not necessarily the same i for all neighborhoods). Define $\equiv_X$ and $\equiv_U$ such that the N_j^x and N_j^u are all equivalence classes of $\mathcal{X}$ and $\mathcal{U}$, respectively. Note that we need to ensure that for any i, j, N_j^x is the same as N_i^x or the two are disjoint, and, a similar condition for N_j^u holds. This condition can be easily ensured during the construction by picking small enough ϵ.

Now, assume that every N_t^x corresponds to an element of *Grid*$(\mathcal{X}, \epsilon_0/2^{i_t})$ for some i_t, and similarly, N_t^u corresponds to an element of *Grid*$(\mathcal{U}, \epsilon_0/2^{j_t})$ for some j_t. Let i be the maximum of the i_ts and j_ts. Note that *Grid*$(\mathcal{X}, \epsilon_0/2^i)$ refines N_t^x and similarly, *Grid*$(\mathcal{U}, \epsilon_0/2^i)$ refines N_t^u. In Figure 1, the squares around x_t^* with bold borders are N_t^x, and the dashed squares which are contained in them correspond to the refined partition. One can define a strategy σ_ϵ (not necessarily a chain anymore) for $\mathcal{T}_\epsilon$ which correspond to following the neighborhoods N_t^x. Hence, all the paths in $\mathcal{T}_\epsilon$ which conform to σ_ϵ will be contained in the neighborhoods N_t^x. Therefore, the cost of σ_ϵ is bounded by that of σ which is at most δ away from the optimal cost, and the optimal cost of $\mathcal{T}_\epsilon$ is at most δ away from that of $\mathcal{T}_\mathcal{D}$.

Observe that $J_{\epsilon_0/2^i} \leq J_{\epsilon_0/2^j}$ for all $i > j$. Furthermore, for any $\delta > 0$, there exists $\epsilon = \epsilon_0/2^i$, such that $|\mathcal{W}(\mathcal{T}_\epsilon, x_0^\epsilon, \Pi) - \mathcal{W}(\mathcal{T}_\mathcal{D}, x_0^*, \Pi)| \leq \delta$. Note $J_\epsilon = \mathcal{W}(\mathcal{T}_\epsilon, x_0^\epsilon, \Pi)$ and $J_{opt} = \mathcal{W}(\mathcal{T}_\mathcal{D}, x_0^*, \Pi)$ is the optimal cost. Hence, $|J_\epsilon - J_{opt}| \leq \delta$, and $J_{\epsilon_0/2^i}$ converges to J_{opt} as i goes to infinity. Lastly, for each sub-optimal cost $J_{\epsilon_0/2^i}$, there exists a suboptimal winning strategy $\sigma_{\epsilon_0/2^i}$ with cost of winning $J_{\epsilon_0/2^i}$. □

At this point, we have shown that the strategy given by OptCAR incurs a suboptimal cost that converges to the optimal cost of $\mathcal{D}$. The strategies used in the proof of Theorem 6 have the property that the length of the maximal paths which conform to the strategy are finite and have a bound (in fact, they are all of the same length). Further, the trace of all the paths is the same. However, during implementation, Algorithm 1 may return a sequence of suboptimal strategies $\sigma_{\epsilon_0/2^i}$ that results in paths with different lengths. Nonetheless, the cost of each path results from $\sigma_{\epsilon_0/2^i}$ is bounded by the cost $J_{\epsilon_0/2^i}$.

In addition, the strategy considered in the proof of Theorem 6 gives a sequence of inputs which satisfy the property Π from any point in an open neighborhood around the given initial state x_0^*. An open neighborhood exists around each of the control inputs such that the resulting paths satisfy Π. Hence, Algorithm 1 in fact returns a controller that is robust against input uncertainties under the assumption that the original system has such optimal control.

VII. Optimal Control of Finite Transition Systems

This section presents a value iteration scheme for computing the optimal cost and optimal strategy for finite transition systems. Observe that the strategies of the abstract system that are used in the proof of Theorem 6 have a linear structure, that is, there are no paths in the abstract system of length greater than the number of states in the system that conform with the strategy. We call such a strategy a layered strategy. Hence, in this section we present an algorithm for computing an optimal strategy for a finite state transition system that is layered. The algorithm is given in Algorithm 2 which is a modified Bellman-Ford algorithm [27].

The function *ReduceReach* reduces the problem of computing the layered strategy for a property Π to that of reachability. It consists of taking a product of the input transition system $\mathcal{T}_S$ and the transition system $\mathcal{T}_P$ of the property. The final states S_f correspond to those of $\mathcal{T}_P$ is labelled by P_f. Define $\mathcal{T}_S = (\mathcal{S}_S, \mathcal{S}_S^{init}, \mathcal{U}_S, \mathcal{P}, \Delta_S, \mathcal{L}_S, \mathcal{W}_S)$. Represent the property by the automaton $\mathcal{T}_P = (\mathcal{S}_P, \mathcal{S}_P^{init}, \mathcal{U}_P, \mathcal{P}, \Delta_P, \mathcal{L}_P, \mathcal{W}_P)$ and the final states of $\mathcal{T}_P$ by a proposition $P_f \in \mathcal{P}$. Note that the set of propositions $\mathcal{P}$ is the same for both $\mathcal{T}_S$ and $\mathcal{T}_P$. The sets $\mathcal{U}_P$ and $\mathcal{W}_P$ are irrelevant to the problem. The transition system returned by *ReduceReach* is $\mathcal{T} = (\mathcal{S}, \mathcal{S}^{init}, \mathcal{U}, \mathcal{P}, \Delta, \mathcal{L}, \mathcal{W})$, where $\mathcal{S} = \{(s_1, s_2) \in \mathcal{S}_S \times \mathcal{S}_P \,|\, \mathcal{L}_S(s_1) = \mathcal{L}_P(s_2)\} \cup \{s_d\}$ (here s_d is a dead state), $\mathcal{U} = \mathcal{U}_S$, $\Delta = \Delta_1 \cup \Delta_2$, where $\Delta_1 = \{((s_1, s_2), u, (s_1', s_2')) \in \mathcal{S} \times \mathcal{U} \times (\mathcal{S} \backslash \{s_d\}) \,|\, (s_1, u, s_1') \in \Delta_S, (s_2, a, s_2') \in \Delta_P$ for some $a\}$ and Δ_2 consists of edges $((s_1, s_2), u, s_d) \in \mathcal{S} \times U \times \{s_d\}$ such that there exists $(s_1, u, s_1') \in \Delta_S$ for some s_1' and there does not exist a and s_2' such that $(s_2, a, s_2') \in \Delta_P$ and $\mathcal{L}_S(s_1') = \mathcal{L}_P(s_2')$. The set of final states S_f of $\mathcal{T}$ with respect with reachability is solved as $S_f = \{(s_1, s_2) \in (\mathcal{S} \backslash \{s_d\}) \times (\mathcal{S} \backslash \{s_d\}) \,|\, \mathcal{L}_P(s_2) = P_f\}$.

Algorithm 2 *SolveFiniteGame* (Two-Player Games)

Require: Finite state transition system $\mathcal{T}_S$, Property Π specified as $(\mathcal{T}_P, P_f)$

$\mathcal{T}, S_f :=$ *ReduceReach*$(\mathcal{T}_S, \mathcal{T}_P, P_f)$

Set for every $s \in \mathcal{S} - S_f$, $C(s) := 0$ if $s \in S_f$ and ∞ otherwise

for $i = 1, \ldots, |\mathcal{S}|$ **do**

  for $s \in \mathcal{S}$ **do**

$$C^i(s) := \min_{u \in \mathcal{U}} \max_{(s,u,s') \in \Delta} (\mathcal{W}(s, u, s') + C^{i-1}(s'))$$

$$\sigma^i(s) := \arg\min_{u \in \mathcal{U}} \max_{(s,u,s') \in \Delta} (\mathcal{W}(s, u, s') + C^{i-1}(s'))$$

  end for

end for

if $C^{|S|}(s_0) < \infty$ **then**

  Output the strategy $\sigma^{|S|}$ and the cost $C^{|S|}(s_0)$

end if

The algorithm initially assigns a cost of 0 to the states in S_f and ∞ otherwise. The cost C^i in the i-iteration captures the optimal cost of reaching S_f by a strategy in which all paths that conform to it have length at most i, and σ^i stores a corresponding strategy. Hence, $C^{|S|}$ provides a layered strategy if $C^{|S|}(s_0) < \infty$. The algorithm can be improved wherein it terminates earlier than completing the $|S|$ iterations, if the costs $\mathcal{C}$ do not change between iterations.

VIII. Implementation

Algorithm 1 and 2 are implemented in the tool OptCAR in Python 2.7. A Python package, NetworkX, is used to represent the graph structures that arise in solving Algorithm 2, and the Parma Polyhedra Library [28] is used to represent the polyhedral sets that arise in the gridding and to solve the linear program problem that arises in the weight computation. OptCAR is tested on a two-tank system from [29] on a MacBook Pro 8.2, 4 core Intel Core i7 processor with speed 2200 Hz, and 8GB RAM.

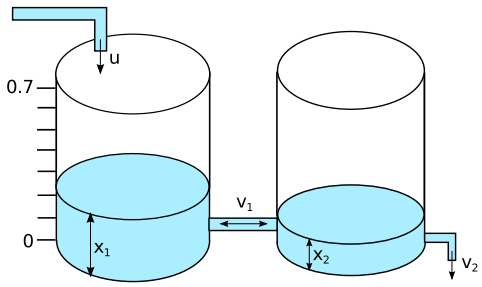


Fig. 2. A schematic of a two-tank system.

The water can flow in between the two tanks through a pipe that connects them (see Figure 2). The pipe is located at level 0.2. Tank 1 (left) has an inflow of water that is managed by a controller, and tank 2 (right) has an outflow of water that is fixed. The controller's goal is to fill up tank 2 to level 0.4 from an initially low water level 0.1 using as small amount of water as possible from the source above tank 1. Formally, the dynamics of this piecewise linear system is

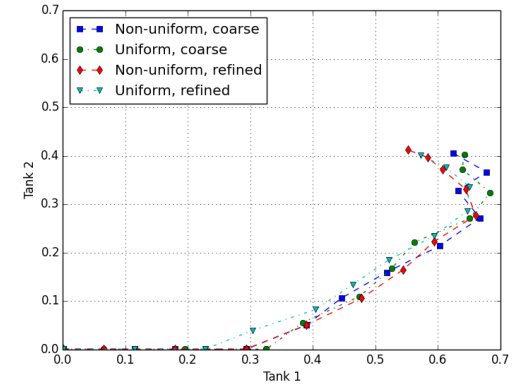
$$x_{t+1} = Ax_t + Bu_t$$

$$A = \begin{cases} A_1 & x \in [0, 0.2]^2 \\ A_2 & \text{otherwise} \end{cases} \qquad B = \begin{bmatrix} 342.6753 \\ 0 \end{bmatrix},$$

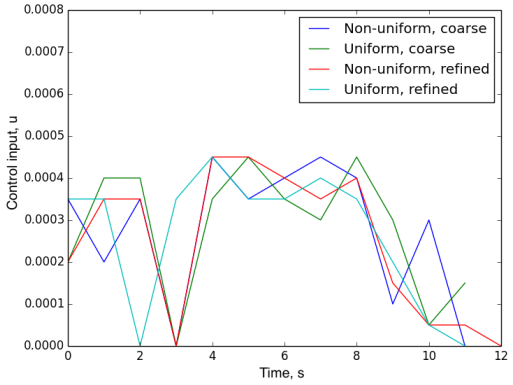
$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0.9635 \end{bmatrix} \quad A_2 = \begin{bmatrix} 0.8281 & 0.1719 \\ 0.1719 & 0.7196 \end{bmatrix},$$

$x_t = (x_t^1, x_t^2) \in [0, 0.7]^2$, and $u_t \in [0, 0.0005]$. The water level in tank 1 at time t is x_t^1, and the water level in tank 2 at time t is x_t^2.

The cost function is chosen to be $\mathcal{J}(\phi(x_0, u)) = \sum_{t \in [k]} \|u_t\|_1$ to represent minimal water inflow, and the goal is to drive the system from partition, $[0, 0.7] \times [0, 0.1]$, to partition $[0, 0.7] \times [0.4, 0.7]$. The algorithm is implemented on two uniform grids on the states - 28×16 and 56×32, and two non-uniform grids - 23×13 and 32×19. The input, u, is partitioned into 10 uniform intervals. In order to reduce computation time, the goal region is combined as one partition for all cases. In addition, for a non-uniform grid, the sizes for the rest of the partitions are not necessary the same. Partitions whereby the states are more likely to take big steps are combined with its neighbors because the states most likely will not end up at the neighboring partitions. Another example of non-uniform grids in the same vein would be to have finer grids near the goal and coarser grids away from the goal. Such modification is feasible if the controller designer has prior information about the system from his/her past experiences. It saves computation time, and also allows for finer grids at places that matter to get a better result.



(a) State trajectory



(b) Control input

Fig. 3. Simulated result of OptCAR on the two-tank system.

Strategies obtained from OptCAR are compared in Table I. Figure 3(a) and 3(b) shows the state and control input trajectory for the four cases. This example shows that choosing a suitable partition can reduce the computation time dramatically while still achieving comparable performance as the performance of a naive uniform grid. Future work includes designing a scheme to partition the domain that can reduce computation time.

IX. Conclusion

In this paper, we consider the problem of synthesizing optimal control strategies for discrete-time piecewise linear system with respect to regular properties. We present an abstraction-refinement approach for constructing arbitrarily

TABLE I

PERFORMANCE OF OPTCAR FOR DIFFERENT CASES. THE FIRST TWO COLUMNS ARE CASES WHEN UNIFORM GRID IS USED. THE LAST TWO COLUMNS ARE CASES WHEN NON-UNIFORM GRID IS USED.

Grid	28×16	56×32	23×13	32×19
Computation time (seconds)	1538	17127	1187	5421
Optimal cost	0.0034	0.0032	0.0035	0.0032
Optimal step	12	13	12	14
Final point	(0.642,0.402)	(0.573,0.401)	(0.625,0.405)	(0.552,0.412)

precise approximations of the optimal cost and the corresponding strategies. The abstraction based approach can be applied to the general class of hybrid systems and for properties over infinite traces, however, the challenge is in computing edges and weights, especially, for non-linear dynamics and in continuous time. Future work will focus on more complex dynamics and continuous-time hybrid systems. To reduce computation time, a more intelligent gridding scheme in the refinement step will be developed. Lastly, the neighborhoods of states and inputs in the abstract system naturally model measurement errors and input uncertainties of the concrete system. Hence, a potential future application of this technique is in synthesizing robust optimal control for a hybrid system.

X. ACKNOWLEDGMENTS

P. Prabhakar was partially supported by EU FP7 Marie Curie Career Integration Grant No. 631622 and NSF CAREER 1552668. This work was partially conducted when Y. P. Leong was an intern at the IMDEA Software Institute.

REFERENCES

[1] A. Church, "Logic, arithmetic, and automata," in *Proc. Internat. Congr. Mathematicians (Stockholm)*. Djursholm: Inst. Mittag-Leffler, 1962, pp. 23–35.

[2] Z. Manna and P. Wolper, "Synthesis of communicating processes from temporal logic specifications," in *Logics of Programs, Workshop, Yorktown Heights, New York*, May 1981, pp. 253–281.

[3] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logics of Programs, Workshop, Yorktown Heights, New York*, May 1981, pp. 52–71.

[4] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan 1987.

[5] E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Hybrid Systems II*, 1994, pp. 1–20.

[6] T. A. Henzinger, B. Horowitz, and R. Majumdar, "Rectangular hybrid games," in *Proceedings of CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands*, August 1999, pp. 320–335.

[7] P. Bouyer, T. Brihaye, and F. Chevalier, "O-minimal hybrid reachability games," *Logical Methods in Computer Science*, vol. 6, no. 1, 2010.

[8] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. E. Dullerud, "Specifications for decidable hybrid games," *Theor. Comput. Sci.*, vol. 412, no. 48, pp. 6770–6785, 2011.

[9] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '95. New York, NY, USA: ACM, 1995, pp. 373–382.

[10] J. Raisch and S. O'Young, "Discrete approximation and supervisory control of continuous systems," *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, vol. 43, no. 4, pp. 569–573, 1998.

[11] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[12] J. Liu, N. Ozay, U. Topcu, and R. M. Murray, "Synthesis of reactive switching protocols from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1771–1785, 2013.

[13] M. Kloetzer and C. Belta, "Temporal logic planning and control of robotic swarms by hierarchical abstractions," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 320–330, 2007.

[14] J. A. DeCastro and H. Kress-Gazit, "Guaranteeing reactive high-level behaviors for robots with complex dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 749–756.

[15] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947–953, 2012.

[16] C. Seatzu, D. Gromov, J. Raisch, D. Corona, and A. Giua, "Optimal control of discrete-time hybrid automata under safety and liveness constraints," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 65, no. 6, pp. 1188 – 1210, 2006.

[17] G. Reissig and M. Rungger, "Abstraction-based solution of optimal stopping problems under uncertainty," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2013, pp. 3190–3196.

[18] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2013, pp. 3197–3204.

[19] E. A. Gol, M. Lazar, and C. Belta, "Temporal logic model predictive control," *Automatica*, vol. 56, pp. 78 – 85, 2015.

[20] A. Bemporad, F. Borrelli, and M. Morari, "Piecewise linear optimal controllers for hybrid systems," in *American Controls Conf. (ACC)*, vol. 2, 2000, pp. 1190–1194 vol.2.

[21] B. Lincoln and B. Bernhardsson, "LQR optimization of linear system switching," *IEEE Trans. Automat. Contr.*, vol. 47, no. 10, pp. 1701–1705, 2002.

[22] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *IEEE Int. Conf. on Decision and Control (CDC)*, 2008, pp. 2117–2122.

[23] M. Mazo and P. Tabuada, "Symbolic approximate time-optimal control," *Systems & Control Letters*, vol. 60, no. 4, pp. 256–263, 2011.

[24] Y. P. Leong and P. Prabhakar, "Optimal control with regular objectives using an abstraction-refinement approach," arXiv:1504.02838.

[25] M. Y. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification (preliminary report)," in *Proceedings of the Symposium on Logic in Computer Science*, Jun 1986, pp. 332–344.

[26] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi, "Alternating refinement relations," in *Proceedings of the Ninth International Conference on Concurrency Theory (CONCUR'98)*, vol. 1466. Springer-Verlag, 1998, pp. 163–178.

[27] R. Bellman, "On a routing problem," DTIC Document, Tech. Rep., 1956.

[28] R. Bagnara, P. M. Hill, and E. Zaffanella, "The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Science of Computer Programming*, vol. 72, no. 1–2, pp. 3–21, 2008.

[29] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1491–1504, June 2012.