Synthesis of Optimal Switching Logic for Hybrid Systems*

Susmit Jha
EECS Dept., UC Berkeley
jha@eecs.berkeley.edu

Sanjit A. Seshia
EECS Dept., UC Berkeley
sseshia@eecs.berkeley.edu

Ashish Tiwari
CSL, SRI International
tiwari@csl.sri.com

ABSTRACT

Given a multi-modal dynamical system, optimal switching logic synthesis involves generating conditions for switching between the system modes such that the resulting hybrid system satisfies a quantitative specification. We formalize and solve the problem of optimal switching logic synthesis for quantitative specifications over long run behavior. Our paper generalizes earlier work on synthesis for safety. We present an approach for specifying quantitative measures using reward and penalty functions, and illustrate its effectiveness using several examples. Each trajectory of the system, and each state of the system, is associated with a cost. Our goal is to synthesize a system that minimizes this cost from each initial state. Our algorithm works in two steps. For a single initial state, we reduce the synthesis problem to an unconstrained numerical optimization problem which can be solved by any off-the-shelf numerical optimization engines. In the next step, optimal switching condition is learnt as a generalization of the optimal switching states discovered for each initial state. We prove the correctness of our technique and demonstrate the effectiveness of this approach with experimental results.

Categories and Subject Descriptors

D.4.4 [**Input/Output and Data Communications**]: Performance Analysis and Design Aids; B.1.2 [**Control Structures and Microprogramming**]: Control Structure Performance Analysis and Design Aids; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Design, Performance

*The UC Berkeley authors were supported in part by NSF grants CNS-0644436 and CNS-0627734, and by an Alfred P. Sloan Research Fellowship. The third author was supported in part by NSF grants CNS-0720721, CSR-0917398 and SHF:CSR-1017483.

EMSOFT'11, October 9–14, 2011, Taipei, Taiwan.

Keywords

Hybrid Systems, Switching Logic Synthesis , Numerical Optimization, Algorithmic Learning

1. INTRODUCTION

One of the holy grails in the design of embedded and hybrid systems is to automatically synthesize models from high-level safety and performance specifications. In general, automated synthesis is difficult to achieve, in part because design often involves human insight and intuition, and in part because of system complexity. Nevertheless, in some contexts, it may be possible for automated tools to *complete partial designs* generated by a human designer, enabling the designer to efficiently explore the design space whilst ensuring that the synthesized system meets its specification.

One such problem is to synthesize the mode switching logic for multi-modal dynamical systems (MDS). An MDS is a physical system (plant) that can operate in different modes. The dynamics of the plant in each mode is known. In order to achieve safe and efficient operation, one needs to design the controller for the plant, typically implemented in software, that switches between the different operating modes. The software component of cyber-physical systems chiefly consists of this control logic. We refer to this problem as *switching logic synthesis*. Designing correct and optimal switching logic can be tricky and tedious for a human designer.

In this paper, we consider the problem of synthesizing the switching logic for an MDS so that the resulting system is *optimal*. Optimality is formalized as minimizing a quantitative cost measure over the long-run behavior of the system. Specifically, we formulate cost as penalty per unit reward motivated by similar cost measure in Economics. For a given initial state, the optimal long-term behavior corresponds to a trajectory of infinite length with infinite number of mode switches which has minimum cost. So, discovering the optimal long-term behavior requires

- discovering this infinite chain of mode switches, and
- the switching states from one mode to another.

Thus, this problem would seem to involve optimization over an infinitely-long trajectory, involving an unbounded set of parameters. However, we reduce this problem to optimization over bounded set of parameters representing the *repetitive long-term behavior*. The key insight is that the long-term cost is essentially the cost of the repetitive part of the behavior. We only require the user to provide a guess of a number of switches which could suffice to reach the repetitive behavior from an initial state. The system stays in repetitive behavior after reaching it and hence, the user can pick any large enough bound. We consider the supersequence of all possible mode sequences with the given number of mode switches and use the times spent in each mode in this supersequence as the parameters for optimization. If the time spent in a particular mode is zero, the mode is removed from the optimum mode sequence. The optimization problem is then formulated as an unconstrained numerical optimization problem which can be solved by off-the-shelf tools. Solving this optimization problem yields the time spent in each mode which in turn gives us the optimum mode switching sequence. So, to summarize, for a given initial state, we obtain a sequence of *switching states* at which mode transitions must occur so as to minimize the long-run cost. The final step involves generalizing from a sample of switching states to a *switching condition*, or *guard*. Given an assumption on the structure of guards, an inductive learning algorithm is used to combine switching states for different initial states to yield the optimum switching logic for the entire hybrid system.

To summarize, the novel contributions of this paper are as follows:

- We formalize the problem of synthesizing optimal switching logic by introducing the notion of long-run cost which needs to be minimized for optimality (Section 2).
- The synthesis problem requires optimization over infinite trajectories and not just a finite time horizon. We show how to reduce optimization over an infinite trajectory to an equivalent optimization over a bounded set of parameters representing the *limit* behavior. (Section 4);
- We present an algorithm to solve this optimization problem for a single initial state based on unconstrained numerical optimization (Section 5). Our algorithm makes *no assumptions on the intra-mode continuous dynamics* other than locally-Lipschitz continuity and relies only on the ability to accurately simulate the dynamics, making it applicable even for nonlinear dynamics;
- An inductive learning algorithm based on randomly sampling initial states is used to generalize from optimal switching states for individual initial states to an optimal switching guard for the set of all initial states. This generated switching logic is guaranteed to be the true optimal switching logic with high probability (Section 6).

Experimental results demonstrate our approach on a range of examples drawn from embedded systems design (Section 7).

2. PROBLEM DEFINITION

2.1 Multimodal and Hybrid Systems

We model a hybrid system as a combination of a multimodal dynamical system and a switching logic.

Definition 1. **Multimodal Dynamical System ($\mathtt{MDS}$).** A *multimodal dynamical system* is a tuple $\langle Q, X, f, \mathtt{Init}\rangle$, where $Q := \{1, \ldots, N\}$ is a set of modes, $X := \{x_1, \ldots, x_n\}$ is a set of continuous variables, $f : Q \times \mathbb{R}^{|X|} \mapsto \mathbb{R}^{|X|}$ defines a vector field for each mode in Q, and $\mathtt{Init} \subseteq Q \times \mathbb{R}^{|X|}$ is a set of initial states. The *state space* of such an MDS is $Q \times \mathbb{R}^{|X|}$. A function $\mathbf{qx} : \mathbb{R}^+ \mapsto (Q \times \mathbb{R}^{|X|})$ is said to be a *trajectory* of this MDS with respect to a sequence $t_1, t_2, \ldots$ of *switching times* if
(i) $\mathbf{qx}(0) \in \mathtt{Init}$ and
(ii) for all i and for all t such that $t_i < t$ and $t < t_{i+1}$, it is the case that $\mathbf{q}(t) = \mathbf{q}(t_i)$ and

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{q}(t), \mathbf{x}(t)), \tag{1}$$

where $\mathbf{q}$ and $\mathbf{x}$ denote the projection of $\mathbf{qx}$ into the mode and continuous state components. The function $\mathbf{x}$ is continuous. The *switching sequence is the sequence of modes* $\mathbf{q}(t_1), \mathbf{q}(t_2), \ldots$.

If $\mathtt{MDS}$ is a multimodal dynamical system, then its semantics, denoted $[\![\mathtt{MDS}]\!]$, is the set of its trajectories with respect to all possible switching time sequences.

Definition 2. **Switching Logic ($\mathtt{SwL}$).** A *switching logic* for a multimodal system $\mathtt{MDS} := \langle Q, X, f, \mathtt{Init}\rangle$ is a tuple $\langle(\mathtt{g}_{q_1 q_2})_{q_1,q_2 \in Q}\rangle$ where $\mathtt{g}_{q_1 q_2} \subseteq \mathbb{R}^X$ is the *guard* defining the switch from mode q_1 to mode q_2.

Given a multimodal system and a switching logic, we can now define a hybrid system by considering only those trajectories of the multimodal system that are consistent with the switching logic.

Definition 3. **Hybrid System ($\mathtt{HS}$).** A *hybrid system* is a tuple $\langle \mathtt{MDS}, \mathtt{SwL}\rangle$ consisting of a multimodal system $\mathtt{MDS} := \langle Q, X, f, \mathtt{Init}\rangle$ and a switching logic $\mathtt{SwL} := \langle(\mathtt{g}_{q_1 q_2})_{q_1,q_2 \in Q}\rangle$. The *state space* of the hybrid system is the same as the state space of $\mathtt{MDS}$. A function $\mathbf{qx} : \mathbb{R}^+ \mapsto (Q \times \mathbb{R}^X)$ is said to be a *trajectory* of this hybrid system if there is a a sequence $t_1, t_2, \ldots$ of *switching times* such that
(a) $\mathbf{qx}$ is a trajectory of $\mathtt{MDS}$ with respect to this switching time sequence and
(b) setting $t_0 = 0$, for all t_i in the switching time sequence with $i \geq 1$, $\mathbf{x}(t_i) \in \mathtt{g}_{\mathbf{q}(t_{i-1})\mathbf{q}(t_i)}$ and for all t such that $t_{i-1} < t < t_i$, $\mathbf{x}(t) \notin \cup_{q \in Q} \mathtt{g}_{\mathbf{q}(t_{i-1})q}$.

Discrete jumps are taken as soon as they are enabled and they do not change the continuous variables. For the notion of a trajectory to be well-defined, guards are required to be closed sets. The semantics of a hybrid system $\mathtt{HS}$, denoted $[\![\mathtt{HS}]\!]$, is the collection of all its trajectories as defined above.

2.2 Quantitative Measures for Hybrid Systems

Our interest is in automatically synthesizing hybrid systems which are *optimal* in the long-run. We define a quantitative measure on a hybrid system $\mathtt{HS}$ by extending $\mathtt{HS}$ with new continuous state variables. The new continuous variables compute "rewards" or "penalties" that are accumulated over the course of a hybrid trajectory. We also allow the new variables to be updated during discrete transitions, which enables us to penalize or reward discrete mode switches.

Definition 4. **Performance Metric.** A *performance metric* for a given multimodal system $\mathtt{MDS} := \langle Q, X, f, \mathtt{Init}\rangle$ is a tuple $\langle \mathtt{PR}, f_{\mathtt{PR}}, \mathtt{update}\rangle$, where $\mathtt{PR} := P \cup R$ is a finite set of continuous variables (disjoint from X), partitioned into penalty variables P and reward variables R, $f_{\mathtt{PR}} : Q \times \mathbb{R}^X \mapsto \mathbb{R}^{\mathtt{PR}}$ defines the vector field that determines the evolution of the variables $\mathtt{PR}$, and $\mathtt{update} : Q \times Q \times \mathbb{R}^{\mathtt{PR}} \mapsto \mathbb{R}^{\mathtt{PR}}$ defines the updates to the variables $\mathtt{PR}$ at mode switches.

Given a trajectory $\mathbf{qx} : \mathbb{R}^+ \mapsto (Q \times \mathbb{R}^X)$ of a multimodal or hybrid system with mode-switching time sequence $t_1, t_2, \ldots$, and given a performance metric, we define the *extended trajectory* $\mathbf{qx}^e : \mathbb{R}^+ \mapsto (Q \times \mathbb{R}^X \times \mathbb{R}^{\mathtt{PR}})$ with respect to the same mode-switching time sequence as any function that satisfies $\mathbf{qx}^e(0) = (\mathbf{q}(0), \mathbf{x}(0), \vec{0})$ and $\mathbf{qx}^e(t) = (\mathbf{q}(t), \mathbf{x}(t), \mathbf{PR}(t))$, where $\mathbf{PR}$ satisfies: $\frac{d\mathbf{PR}(t)}{dt} = f_{\mathtt{PR}}(\mathbf{qx}(t))$ for all $t : t_i < t < t_{i+1}$, and $\mathbf{PR}(t_i) = \mathtt{update}(\mathbf{q}(t_{i-1}), \mathbf{q}(t_i), \lim_{t \to t_i^-} \mathbf{PR}(t))$.

The $\mathtt{cost}$ of a trajectory $\mathbf{qx}$ is defined using its corresponding extended trajectory $\mathbf{qx}^e$ as

$$\mathtt{cost}(\mathbf{qx}) := \lim_{t \to \infty} \sum_{i=1}^{|P|} \frac{\mathbf{P}_i(t)}{\mathbf{R}_i(t)} \tag{2}$$

where $\mathbf{P}_i$ and $\mathbf{R}_i$ are the projection of $\mathbf{qx}^e$ onto the i-th penalty variable and i-th reward variable, and $|P| = |R|$.

We are only interested in trajectories where the above limit exists and is finite. We will further define $\mathtt{cost}$ of a part of a trajectory from time instant t_1 to a time instant t_2 ($t_2 > t_1$) as follows:

$$\mathtt{cost}(\mathbf{qx}, t_1, t_2) := \sum_{i=1}^{|P|} \frac{\mathbf{P}_i(t_2) - \mathbf{P}_i(t_1)}{\mathbf{R}_i(t_2) - \mathbf{R}_i(t_1)} \tag{3}$$

where $\mathbf{P}_i$ and $\mathbf{R}_i$ are components of $\mathbf{PR}$ as before.

As the definition of $\mathtt{cost}$ indicates, we are interested in the *long-run average* (penalty per unit reward) cost rather than (penalty or reward) cost over some bounded/finite time horizon. Examples of cost function can be found in Section 7. More examples are available in the extended version [17].

2.3 Optimal Switching Logic Synthesis

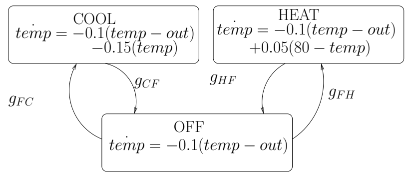
Definition 5. **Optimal Switching Synthesis Problem.** Given a multimodal system $\mathtt{MDS} = \langle Q, X, f, \mathtt{Init} \rangle$, and a performance metric, the *optimal switching logic synthesis problem* seeks to find a switching logic $\mathtt{SwL}^*$ such that the cost of a trajectory from any initial state in the resulting hybrid system $\mathtt{HS}^* := \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL}^*)$ is no more than the cost of the trajectory from the same initial state in an hybrid system $\mathtt{HS} := \mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ obtained using any arbitrary switching logic $\mathtt{SwL}$, that is, $\forall (\mathbf{x}, q) \in \mathtt{Init}\ .\ \mathtt{cost}(\mathbf{qx}^*) \leq \mathtt{cost}(\mathbf{qx})$, *where* $\mathbf{qx}^*(0) = \mathbf{qx}(0) = (\mathbf{x}, q), \mathbf{qx} \in [\![\mathtt{HS}^*]\!], \mathbf{qx} \in [\![\mathtt{HS}]\!]$

We will assume, without loss of any generality, that we are given an over-approximation of the switching logic $\mathtt{SwL}^{over} := \langle (\mathtt{g}^{over}_{qq'})_{q,q' \in Q} \rangle$. In this case, the optimal synthesis problem seeks to find a switching logic $\mathtt{SwL}^* := \langle (\mathtt{g}^*_{qq'})_{q,q' \in Q} \rangle$ that also satisfies the constraint that $\mathtt{g}^*_{qq'} \subseteq \mathtt{g}^{over}_{qq'}$ for all $q, q' \in Q$, which is also written in short as $\mathtt{SwL}^* \subseteq \mathtt{SwL}^{over}$.

The over-approximation $\mathtt{SwL}^{over}$ of the switching set can be used to restrict the search space for switching conditions. The set $\mathtt{g}^{over}_{qq'}$ can be an empty set if switches are disallowed from q to q'. The set $\mathtt{g}^{over}_{qq'}$ can be $\mathbb{R}^X$ if there is no restriction on switching from q to q'.

2.4 Running Example

Let us consider a simple three mode thermostat controller as our running example. The multimode dynamical system describing this system is presented in Figure 1. The thermostat controller is described by the tuple $\langle Q, X, f, \mathtt{Init} \rangle$ where $Q = \{\mathtt{OFF}, \mathtt{HEAT}, \mathtt{COOL}\}$, $X = \{\mathtt{temp}, \mathtt{out}\}$, f is $f_{\mathtt{OFF}}$: $\dot{\mathtt{temp}} = -0.1(\mathtt{temp} - \mathtt{out})$ in mode $\mathtt{OFF}$, $f_{\mathtt{HEAT}}$: $\dot{\mathtt{temp}} = -0.1(\mathtt{temp} - \mathtt{out}) + 0.05(80 - \mathtt{temp})$ in mode $\mathtt{HEAT}$ and $f_{\mathtt{COOL}}$: $\dot{\mathtt{temp}} = -0.1(\mathtt{temp} - \mathtt{out}) - 0.15(\mathtt{temp})$ in mode $\mathtt{COOL}$, and $\mathtt{Init} = \mathtt{OFF} \times [18, 20] \times [12, 26]$. For simplicity, we assume that the outside temperature $\mathtt{out}$ does not change.



$$\dot{\mathtt{discomfort}} = (\mathtt{temp} - 20)^2, \ \dot{\mathtt{fuel}} = (\mathtt{temp} - \mathtt{out})^2$$

$$\dot{\mathtt{swTear}} = 0, \ \dot{\mathtt{time}} = 1$$

$$\mathtt{update}(\mathtt{M}, \mathtt{M}', \mathtt{swTear}) = \mathtt{swTear} + 0.5$$

for any two different modes M, M′ in Q

Figure 1: Thermostat Controller

The performance requirement is to keep the temperature as close as possible to the target temperature 20 and to consume as little fuel as possible in the long run. We also want to minimize the wear and tear of the heater caused by switching. The performance metric is given by the tuple $\langle \mathtt{PR}, f_{\mathtt{PR}}, \mathtt{update} \rangle$, where penalty variables $P = \{\mathtt{discomfort}, \mathtt{fuel}, \mathtt{swTear}\}$ denote the discomfort, fuel and wear-tear due to switching and reward variables $R = \{\mathtt{time}\}$ denote the time spent. The evolution and update functions for the penalty and reward variables is shown in Figure 1. We need to synthesize the guards such that the following cost metric is minimized. Since the reward variable is the time spent, minimizing this metric means minimizing the *average* discomfort, fuel cost and wear-tear of the heater. We give a higher weight (10) to discomfort than fuel cost and wear-tear.

$$\lim_{t \to \infty} \frac{10 \times \mathtt{discomfort}(t) + \mathtt{fuel}(t) + \mathtt{swTear}(t)}{\mathtt{time}(t)}$$

3. RELATED WORK

There is a lot of work on synthesis of controllers for hybrid systems, which can be broadly classified along several different dimensions. First, based on the *property of interest*, synthesis work broadly falls into one of two categories. The first category finds controllers that meet some *liveness* specifications, such as synthesizing a trajectory to drive a hybrid system from an initial state to a desired final state [22, 19], while also minimizing some cost metric [6]. The second category finds controllers that meet some safety specification; see [1] for detailed related work in this category. Our previous work [16] based on combining simulation and algorithmic learning also falls in this caregory. Purely constraint based approaches for solving switching logic synthesis problem have also being used for reachability specifications [30]. While our work does not directly consider only safety or only liveness requirements, both these requirements can be suitably incorporated into the definition of "reward" and "penalty" functions that define the cost that our approach then optimizes. While optimal control problems for hybrid systems have been formulated where cost is defined over some finite trajectory, we are unaware of any work in control of hybrid systems that attempts to formulate and solve the optimal control problem for *long-run* costs.

The second dimension that differentiates work on controller synthesis for hybrid systems is the space of control inputs considered; that is, what is assumed to be controllable. The space of controllable inputs could consist of any combination of continuous control inputs, the mode sequence, and the dwell times within each mode. A recent paper by Gonzales et al. [13] consider all the three control parameters, whereas some other works either assume the mode sequence is not controllable [33, 28] or there are no continuous control inputs [2]. In our work, we assume there are no continuous control inputs and both the mode sequence and the dwell time within each mode are controllable entities.

The third dimension for placing work on controller synthesis of hybrid systems is the approach used for solving the synthesis problem. There are direct approaches for synthesis that compute the controlled reachable states in the style of solving a game [1, 31], and abstraction-based approaches that do the same, but on an abstraction or approximation of the system [25, 10, 29]. Some of these approaches are limited in the kinds of continuous dynamics they can handle. They all require some form of iterative fixpoint computation. The other class of approaches are based on using nonlinear optimization techniques and gradient descent [13, 2]. Axelsson et al. [3] use a bi-level hierarchical optimization algorithm with the higher level used for finding optimal mode sequence employing single mode insertion technique, and the lower level used to find the switching times that minimizes the cost function for fixed mode sequence. Gonzales et al. [13, 12] extended the technique by also considering control inputs apart from mode sequence and switching times. Their approach [12] can handle multiple objectives in the cost function, can be initialized at an infeasible point and can include switching costs.

Notions of long-run cost similar to ours have appeared in other areas. The notion of long-run average cost is used in economics to describe the cost per unit output (reward) in the long-run. In computer science, long-run costs have been studied for graph optimization problems [18]. Long-run average objectives have also been studied for markov decision processes (MDPs) [8, 20]. However, MDPs do not have any continuous dynamics. Another related work is optimal scheduling using priced timed automata [27] in which timed automata are extended by associating a fixed cost to each transition and a fixed cost rate per time unit in a location. We consider multi-modal dynamical systems with possibly non-linear dynamics and our cost rates are functions of the continuous variables. Further, our interest is in long-run cost. There is some recent work on controller synthesis with budget contraints where the budget applies in the long-run [9].

In contrast to existing literature, we present an automated synthesis algorithm to synthesize switching logic $\mathtt{SwL}$ for a given $\mathtt{MDS}$ and performance metric such that all trajectories in the hybrid system $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ have minimum long-term cost with respect to the given performance metric.

4. OPTIMIZATION FORMULATION

In this section, we formulate the problem of finding switching logic for minimum long-run cost from an initial state as an optimization problem. Given a multimodal system $\mathtt{MDS} = \langle Q, X, f, d, Inv, Init\rangle$, an initial state $(q_0, \mathbf{x}_0) \in Init$ and the performance metric tuple $(Y, f_Y, \mathtt{update})$, we need to find the switching times $t_1, t_2, \ldots$ and the mode switching sequence $\mathbf{q}$ such that the corresponding trajectory $\mathbf{qx}$ is of minimum cost.

$$\min_{\mathbf{q}, t_1, t_2, \ldots} \ \mathtt{cost}(\mathbf{qx}) \quad \textit{subject to}$$
$$(1)[\text{Init}] : \mathbf{qx}(0) = (q_0, \mathbf{x}_0) \ \ (2)[\text{Guards}] : \forall i \ \mathbf{x}(t_i) \in \mathbf{g}^{over}_{\mathbf{q}(i)\mathbf{q}(i+1)}$$
$$(3)[\text{Time elapse}] : \forall t \ . \ t_i < t < t_{i+1} \ . \ \mathbf{q}(t) = \mathbf{q}(t_i), i = 1, 2, \ldots ;$$
$$(4)[\text{Flow}] : \forall t \ \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{q}(t), \mathbf{x}(t)) \quad (4)$$

Since the switching sequence $t_1, t_2, \ldots$ could be of infinite length, it is not a-priori evident how to solve the above problem. In the rest of the section, we formulate an equivalent optimization problem with finite number of switching times as variables.

Let *trajectory segment* $\mathbf{qx}_{[tf,te]}$ of a trajectory $\mathbf{qx}$ of length $L = tf - te$ be the restriction of the *trajectory* to $tf \le t \le te$, that is, $\mathbf{qx}_{[tf,te]} : T \mapsto (Q \times \mathbb{R}^X)$ where $T = [tf, te] \subseteq \mathbb{R}^+$ and $\mathbf{qx}_{[tf,te]}(t) = \mathbf{qx}(t)$ for $tf \le t \le te$. The *switching times* of the trajectory segment is a finite subsequence $t_m, t_{m+1}, \ldots t_n$ of the switching times $t_1 \ldots, t_m, \ldots, t_n, \ldots$ of the trajectory $\mathbf{qx}$ and $t_{m-1} < tf \le t_m$ and $t_n \le te < t_{n+1}$. The special case of a trajectory segment is a *trajectory prefix* in which the trace starts at time $ts = 0$.

Our goal is to minimize the lifetime cost. The lifetime cost is dominated by the the cost of the *limit behavior* of the system. We are only interested in the following stable limit behaviors when the lifetime cost is defined by the limit in Equation 2.

- *asymptotic:* for any ϵ, there exists a time t_ϵ after which the trajectory gets asymptotically ϵ-close to some state $(q_T, \mathbf{x}_T)$, $||\mathbf{qx}(t) - (q_T, \mathbf{x}_T)||^2 < \epsilon$ for all $t \ge t_\epsilon$ where $||.,.||$ denotes the Euclidean norm, or
- *converging:* there exists a time t_{conv} after which the trajectory converges, $\mathbf{qx}(t) = \mathbf{qx}(t_{conv})$ for all $t \ge t_{conv}$, or
- *cyclic:* there exists a time t_{cyc} after which the trajectory enters a cycle with period L, $\mathbf{qx}(t) = \mathbf{qx}(t + kP)$ for all $t \ge t_{cyc}$ and $k \ge 1$.

In all these cases, we can reason about the long-run cost by considering some finite, but arbitrarily long, trajectory prefixes. Suppose the trajectory $\mathbf{qx}$ is asymptotic to some hybrid state $\mathbf{qx}^\infty = (q^\infty, \mathbf{x}^\infty)$. In this case, we assume that the penalty and reward variables $\mathtt{PR}$ also asymptotically approach some values $\mathtt{PR}^\infty = (P^\infty, R^\infty)$. Now consider the trajectory prefix $\mathbf{qx}_{[0,te]}$. We have

$$\mathtt{cost}(\mathbf{qx}) = \sum_i \frac{P_i^\infty}{R_i^\infty} \quad \text{and}$$
$$\mathtt{cost}(\mathbf{qx}_{[0,te]}) = \sum_i \frac{P_i(te)}{R_i(te)} < \sum_i \frac{P_i^\infty + \epsilon}{R_i^\infty - \epsilon} < \mathtt{cost}(\mathbf{qx}) + \delta_\epsilon$$

Hence, by choosing te appropriately, we can find a trajectory prefix whose cost is arbitrarily close to the cost of the asymptotic trajectory.

Any repetitive trajectory $\mathbf{qx}$ can be decomposed into a finite prefix $\mathbf{qx}_{pref} = \mathbf{qx}_{[0,tp]}$ followed by a trajectory segment $\mathbf{qx}_{rep} = \mathbf{qx}_{[tp,tP]}$ repeated infinitely. We say

$$\mathbf{qx} = \mathbf{qx}_{pref} \ . \ (\mathbf{qx}_{rep})^\omega$$

when $\forall t \leq tp$. $\mathbf{qx}(t) = \mathbf{qx}_{pref}(t)$ and $\forall t \geq tp$. $\mathbf{qx}(t) = \mathbf{qx}_{rep}(tp + r)$ where $r = (t - tp) \mod L$ and $L = tP - tp$. The case when the trajectory converges to some hybrid state can be treated in the same way as a repetitive trajectory.

In Lemma 1 and Theorem 1, we summarize how cost converges to a limit for trajectories with repetitive limit behavior. The proof of Lemma 1 and Theorem 1 are available in the extended version [17].

LEMMA 1. *For each repetition of the segment $\mathbf{qx}_{rep} = \mathbf{qx}_{[tp,tP]}$, the change in penalty and reward variables is constant, that is, for $P = tP - tp$.*

$$\begin{aligned}\forall k \geq 1 \;.\; P_i(tp + kP) - P_i(tp + (k-1)P) \\ = P_i(tp + P) - P_i(tp) = \Delta P_i \\ \forall k \geq 1 \;.\; R_i(tp + kP) - R_i(tp + (k-1)P) \\ = R_i(tp + P) - R_i(tp) = \Delta R_i\end{aligned}$$

THEOREM 1. *For a trajectory $\mathbf{qx}$ which can be decomposed into $\mathbf{qx}_{pref} \;.\; (\mathbf{qx}_{rep})^\omega$, the cost of the trajectory is equal to the cost of the repetitive segment $\mathbf{qx}_{rep}$, that is, $\texttt{cost}(\mathbf{qx}) = \texttt{cost}(\mathbf{qx}_{rep})$.*

Using Theorem 1, the optimization problem in Equation 4 is equivalent to the following optimization problem. Intuitively, if the repetitive part of the trajectory and the finite prefix before the repetitive part have finite cost, then the long run cost of a trajectory in the limit is the cost of the repetitive part of the trajectory. More generally, to also handle the case when the (optimal) trajectory is asymptotic, we can replace the cyclicity requirement, $\mathbf{qx}(tp) = \mathbf{qx}(tP)$, in the optimization problem by the weaker requirement that the state $\mathbf{qx}(tP)$ at time tP be very "close" to the state $\mathbf{qx}(tp)$ at time tp; see also Section 5.1.

$$\min_{\mathbf{q}, t_1, t_2, \ldots} \texttt{cost}(\mathbf{qx}) \quad \textit{subject to} \tag{5}$$

$$\begin{aligned}&(1)[\text{Init}] : \mathbf{qx}(0) = (q_0, \mathbf{x}_0) \;\; (2)[\text{Guards}] : \forall i \; \mathbf{x}(t_i) \in \mathbf{g}^{over}_{\mathbf{q}(i)\mathbf{q}(i+1)} \\ &(3)[\text{Time elapse}] : \forall t \;.\; t_i < t < t_{i+1} \;.\; \mathbf{q}(t) = \mathbf{q}(t_i), i = 1, 2, \ldots; \\ &(4)[\text{Flow}] : \forall t \; \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{q}(t), \mathbf{x}(t)) \\ &(5)[\text{Repetitive Trajectory}] : \; \mathbf{qx} = \mathbf{qx}_{pref} \;.\; (\mathbf{qx}_{rep})^\omega \\ &(6)[\text{Repetitive Time}] : \; \mathbf{qx}_{pref} = \mathbf{qx}_{[0,tp]}, \; \mathbf{qx}_{rep} = \mathbf{qx}_{[tp,tP]} \\ &\text{where } 0 \leq t_1 \leq \ldots t_n \leq tP, \; 0 \leq tp < tP\end{aligned}$$

5. OPTIMIZATION ALGORITHM

In this section, we present an algorithm to solve the above optimization problem. The key idea is to construct a scalar function $F(\mathbf{q}, t_1, t_2, \ldots, t_n, tp, tP)$ where $\mathbf{q}$ is the switching mode sequence; $t_1, t_2, \ldots, t_n$ are the switching times, and tp, tP are the times denoting repetitive behavior, such that the minimum value of F is attained when the switching mode sequence and switching times correspond to the trajectory $\mathbf{qx}$ with minimum long-run cost, and $\mathbf{qx}_{[tp,tP]}$ is the repetitive part of the trajectory.

Once we have constructed F, we need to minimize F. Apart from $\mathbf{q}$, all arguments of F are real-valued. Suppose we fix $\mathbf{q}$ and let $F_{\mathbf{q}}(t_1, t_2, \ldots, t_n, tp, tP)$ denote the function F with fixed mode sequence $\mathbf{q}$. Now $F_{\mathbf{q}}$ is a function from multiple real variables to a real, and hence (approximate) minimization of F can be performed using *unconstrained nonlinear numerical optimization* techniques [5]. These techniques only require that we are able to evaluate F once its arguments are fixed. This we accomplish using numerical simulation of the multimodal system [1].

5.1 Defining F

The optimization problem in Equation 5 is a constrained optimization problem. The constraint $\mathbf{qx} = \mathbf{qx}_{pref} \;.\; (\mathbf{qx}_{rep})^\omega$ requires identifying a trajectory $\mathbf{qx}$ starting from the given initial state $(q_0, \mathbf{x}_0)$ such that it enters repetitive behavior at time tp, and $q(tp) = q(tP)$ and $\mathbf{x}(tp) = \mathbf{x}(tP)$ where $tp < tP$. We call this constraint the repetition constraint. A standard technique for solving some constrained optimization problems is to translate it into an unconstrained optimization problem by modifying the optimization objective such that optimization automatically enforces the constraint. This is done by quantifying the violation using some metric and then minimizing the sum of the earlier minimization objective and the weighted violation measure. In order to enforce the repetition constraint by suitably modifying the optimization objective, we introduce a distance function between the hybrid states. Let d be the distance function between two hybrid states such that

$$d((q_1, \mathbf{x}_1), (q_2, \mathbf{x}_2)) = ||\mathbf{x}_1 - \mathbf{x}_2||^2 \;\text{ if } q_1 = q_2 \;\text{ and } \infty \; o.w.$$

where $||\mathbf{x}_1 - \mathbf{x}_2||$ is the Euclidean norm. $(Q \times X, d)$ forms a metric space. So, the distance between the hybrid states is 0 if and only if $q_1 = q_2$ and $\mathbf{x}_1 = \mathbf{x}_2$.

Let $F(\mathbf{q}, t_1, \ldots, t_n, tp, tP)$

$$= \begin{cases} \texttt{cost}(\mathbf{qx}_{[tp,tP]}) + M \times d(\mathbf{qx}(tp), \mathbf{qx}(tP)) \\ \texttt{if } (a) \; 0 \leq t_1 \leq \ldots t_n \leq tP, \; tp < tP \text{ and} \\ \quad (b) \; \forall i \; \mathbf{x}(t_i) \in \mathbf{g}^{over}_{\mathbf{q}(i)\mathbf{q}(i+1)} \\ \infty \quad \texttt{otherwise} \end{cases}$$

where M is any positive constant and $\mathbf{qx}$ is a trajectory starting from the given initial state, that is,

$$\mathbf{qx}(0) = (q_0, \mathbf{x}_0); \; \forall t \; t_i < t < t_{i+1} \; \mathbf{q}(t) = \mathbf{q}(t_i) \;, \; i = 1, 2, \ldots \\ \text{and } \forall t \; \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{q}(t), \mathbf{x}(t))$$

It is easy to see that the minimum value of the function F is attained when the hybrid states at time tp and tP are the same, that is, the trajectory segment $\mathbf{qx}_{[tp,tP]}$ is the repetitive part of the trajectory and the cost of this segment is minimum. Using Theorem 1, we conclude that the optimization problem in Equation 5 of Section 4 can be reduced to the following unconstrained multivariate numerical optimization problem

$$\min_{t_1, \ldots, t_n, tp, tP} F(t_1, \ldots, t_n, tp, tP) \tag{6}$$

As remarked above, if the arguments of F are fixed, then F can be evaluated using a numerical simulator. Also, for a fixed $\mathbf{q}$, we can use a numerical nonlinear optimization engine to find the minimum value of the function $F_{\mathbf{q}}$.

Running Example

We illustrate our technique for the running example with a fixed sequence of modes say $\mathbf{q} = \texttt{OFF}, \texttt{HEAT}, \texttt{OFF}$ starting from the initial state $(\texttt{OFF}, \texttt{temp} = 22, \texttt{out} = 16)$. The outside

[1]We rely on simulating continuous behavior described by ODEs in a single mode for a fixed time period and accurate simulation of ODEs is a well-studied problem.

temperature `out` does not change with time and remains the same as the initial state. Only the room temperature `temp` changes with time. The switching time sequence is t_1, t_2. Let tp denote the time when the thermostat enters the repetitive behavior and tP be the time such that $\texttt{temp}(tp) = \texttt{temp}(tP)$. When $t_1 \leq t_2 \leq tp \leq tP$ and $tp < tP$, the function

$$F_{\mathbf{q}}(t_1, t_2, tp, tP) = \texttt{cost}(\mathbf{qx}_{[tp,tP]}) + 1000(\texttt{temp}(tp) - \texttt{temp}(tP))^2$$

and it is set to 2000 otherwise (approximating infinity in the formulation with a very high constant). We use *ode*45 function in MATLAB [24] for numerically simulating the ordinary differential equations representing continuous dynamics in each mode. In order to find the minimum value of $F_{\mathbf{q}}$ and the corresponding arguments that minimize the function, we use the implementation of Nelder-Mead simplex algorithm [26]. The minimum value of $F_{\mathbf{q}}$ is obtained at

	t_0	t_1	t_2	tp	tP
t	0	5.02	5.24	3.54	5.24
temp	22.0	19.6	20.2	20.2	20.2

So, the switch states corresponding to the minimum long-run cost for the given initial state ($\texttt{OFF}, \texttt{temp} = 22, \texttt{out} = 16$) and given switching sequence of modes `OFF`, `HEAT`, `OFF` is $\mathtt{g}_{HF} = \{20.2\}$ and $\mathtt{g}_{FH} = \{19.6\}$.

We repeat the experiments with different initial states but with the same mode switching sequence. Even with different initial states ($\texttt{OFF}, \texttt{temp} = 20.5, \texttt{out} = 16$), ($\texttt{OFF}, \texttt{temp} = 21, \texttt{out} = 16$) and ($\texttt{OFF}, \texttt{temp} = 21.5, \texttt{out} = 16$), we obtain the same switching states in this example: $\mathtt{g}_{HF} = \{20.2\}$ and $\mathtt{g}_{FH} = \{19.6\}$.

When we change the mode switching sequence to `OFF`, `HEAT`, `OFF`, `HEAT`, `OFF`, we discover the optimal switching sequence to be

	t_0	t_1	t_2	t_3	t_4	tp	tP
t	0	5.02	5.24	6.73	6.95	3.54	6.95
temp	22.0	19.6	20.2	19.6	20.2	20.2	20.2

$t_1 = 5.02, t_2 = 5.24, t_3 = 6.73, t_4 = 6.95, tp = 3.54, tP = 6.95$ which again yields the same optimal switching states $\mathtt{g}_{HF} = \{20.2\}$ and $\mathtt{g}_{FH} = \{19.6\}$.

We observe that the optimal behavior with respect to the given cost metric would be to switch from `OFF` mode to `HEAT` mode at $\texttt{temp} = 19.6$ and then switch from `HEAT` to `OFF` mode at $\texttt{temp} = 20.2$ regardless of the initial room temperature as long as the outside temperature $\texttt{out} = 16$. The optimal mode cycle is between `OFF` and `HEAT` modes.

For an initial state with outside temperature higher than the outside room temperature $\texttt{out} > 20$, the optimal cycle would be between `OFF` and `COOL` modes. With the mode sequence `OFF`, `COOL`, `OFF` and the initial state ($\texttt{OFF}, \texttt{temp} = 20.5, \texttt{out} = 26$), we discover the optimal switching states to be $\mathtt{g}_{CF} = \{20\}$ and $\mathtt{g}_{FC} = \{20.3\}$.

5.2 Finding Optimal Mode Sequence

The algorithm above assumed that the switching mode sequence $\mathbf{q}$ was fixed. It can be easily adapted to also automatically discover the optimal switching mode sequence. Any mode sequence starting in mode 1 and with atmost k switches in a system with N modes $Q = \{1, 2, \ldots N\}$ is a subsequence of $1(2 \ \ldots \ N \ 1)^k$, that is, mode 1 followed by $(2 \ \ldots \ N \ 1)$ repeated k times. Let dwell-time of a mode i be the time spent in the mode $t_{i+1} - t_i$. Given the switching times $t_1, t_2, \ldots t_{Nk}$ and tp, tP, we define the NZ function which removes the switch times and modes from the switching sequence with zero dwell-times, that is,

$$NZ(\overline{\mathbf{q}}, t_1, t_2, \ldots, t_{Nk}, tp, tP) = (\mathbf{q}, t_{i_1}, t_{i_2}, \ldots, t_{i_K}, tp, tP)$$
$$\text{where } \mathbf{q} = q_{i_1}, q_{i_2}, \ldots, q_{i_K}, 0 < t_{i_1} < t_{i_2} < \ldots < t_{i_K} < tP$$
$$\text{and } t_m = t_{i_j} \text{ for all } i_j < m < i_{j+1}$$

For example, given the sequence of switching times $5, 6, 6, 11, 12, 12$ and $tp = 6.5, tP = 12.5$ with the switching mode sequence $\overline{\mathbf{q}} = 1, 2, 3, 1, 2, 3, 1$,

$$NZ(\overline{\mathbf{q}}, 5, 6, 6, 11, 12, 12, 6.5, 12.5) = (\mathbf{q}, 5, 6, 11, 12, 6.5, 12.5)$$

where $\mathbf{q} = 1, 2, 1, 2, 1$.

Given a guess on the number of mode switches k such that k or less switches are needed to reach the optimal repetitive behavior, we can use $\overline{\mathbf{q}} = 1(2 \ \ldots \ N \ 1)^k$ as the over-approximate switching mode sequence and then find the optimal switching subsequence corresponding to the minimal long-run cost behavior using the following modified optimization formulation.

$$\min_{t_1, \ldots, t_{Nk}, tp, tP} \quad F(NZ(\overline{\mathbf{q}}, t_1, \ldots, t_{Nk}, tp, tP)) \qquad (7)$$

If the optimal value returned by minimizing the above function is attained with the arguments $t_1^*, \ldots, t_{Nk}^*, tp^*, tP^*$, then the optimal switching sequence $\mathbf{q}$ and the optimal switching time sequence is given by

$$(\mathbf{q}, t_{i_1}, \ldots, t_{i_k}, tp, tP) = NZ(\overline{\mathbf{q}}, t_1^*, \ldots, t_{Nk}^*, tp^*, tP^*)$$

Running Example

We illustrate the above technique on the running example below. Let us guess that reaching the optimal repetitive behavior from the initial state $\texttt{OFF}, \texttt{temp} = 22, \texttt{out} = 16$ takes atmost 2 switches. We consider the mode sequence `OFF`, `HEAT`, `COOL`, `OFF`, `HEAT`, `COOL`, `OFF` which would contain all mode sequences with 2 switches (it also contains some mode sequences with more than 2 switches). We try to minimize the corresponding function $F(NZ(t_1, t_2, \ldots, t_6, tp, tP))$.

The minimum value obtained for the function F with the starting state ($\texttt{OFF}, \texttt{temp} = 22, \texttt{out} = 16$) by our optimization engine corresponds to the following trajectory.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	tp	tP
t	0	5.08	5.32	5.32	6.97	7.23	7.23	4.87	8.66
temp	22.0	19.6	20.2	20.2	19.6	20.2	20.2	19.7	19.7

The optimal mode sequence and the switching times points are obtained as

$$NZ(\overline{\mathbf{q}}, 5.08, 5.32, 5.32, 6.97, 7.23, 7.23, 4.87, 8.66) =$$
$$(\texttt{OFF}, \texttt{HEAT}, \texttt{OFF}, \texttt{HEAT}, \texttt{OFF}, 5.08, 5.32, 6.97, 7.23, 4.87, 8.66)$$

Since $tp = 4.87$ and $tP = 8.66$, the repetitive part of the mode sequence is `HEAT`, `OFF`. The switch from mode `OFF` to `HEAT` occurs at times t_1 and t_4. We observe that $\texttt{temp}(t_1) = \texttt{temp}(t_4) = 19.6$. So, the optimal trajectory switches from `OFF` to `HEAT` at $\texttt{temp} = 19.6$. The switches from `HEAT` to `COOL` and then to `OFF` occur at the same times: $t_2 = t_3$ and $t_5 = t_6$. So, the dwell-time in the mode `COOL` is 0 and it needs to be removed from the optimal switching sequence. The switch into mode `OFF` occurs at times t_3 and t_6 with

$\texttt{temp}(t_3) = \texttt{temp}(t_6) = 20.2$. Thus, the optimal mode sequence is $\texttt{OFF}, (\texttt{HEAT}, \texttt{OFF})^\omega$ and the guards discovered from this trajectory are $\texttt{g}_{FH} = 19.6$ and $\texttt{g}_{HF} = 20.2$. □

Thus, the approach presented so far can be used to synthesize switching conditions for minimum cost long-run behavior for a given initial state. We need a guess on the number of switches k such that the optimal behavior has atmost k switches. We summarize the guarantee of our approach for a single initial state in the following theorem

THEOREM 2. *For a single initial state, our technique discovers the switching states corresponding to the optimal trajectory with minimum long-run cost if numerical optimization engine can discover global minimum of the numerical function F.*

The proof of the above theorem follows from the definition of F. If numerical optimization engines are guaranteed to only find local minima of F, our technique will find trajectories of minimal cost. We employ the Nelder-Mead simplex algorithm as described by Lagarias et al [21, 26] for minimizing F since it is a derivative-free method and it can better handle discontinuities in function F. We use its implementation available as the *fminsearch* [23] function in MATLAB.

6. MULTIPLE INITIAL STATES

The approach presented in Section 5 can find the switching state for each mode switch along the trajectory corresponding to optimal long-run behavior *for a given initial state*. However, since systems are generally designed to operate in more than one initial state, we need to synthesize the *guard condition* for mode switches such that the trajectory from each initial state has optimal long-run cost. In this section, we present a technique for synthesizing guard conditions in a probabilistic setting, where we assume the ability to sample initial states from their (arbitrary) probability distribution. Our technique samples initial states and obtains corresponding optimal switching states for each mode switch. From the individual optimal switching states, we generalize to obtain the guard condition for the mode switch using *inductive learning* (learning from examples). In order to employ learning, we make a *structural hypothesis* on the form of guards and use concept learning algorithms to efficiently learn the guards from sampled switching states.

Structural Hypothesis:
We assume that the guard condition is a halfspace, that is, a linear inequality over the continuous variables X.

In the rest of the section, we discuss how the existing results from algorithmic concept learning can be used efficiently to learn a halfspace representing the guard condition from the discovered switching states for each mode-switch.

Background: We first mention results which prove the efficient learnability of halfspaces and then present an algorithm which can be used to learn halfspaces in the probabilistically approximately correct (PAC) learning framework [32]. In this framework, the learner receives samples marked as positive or negative for points lying inside and outside the concept respectively, and the goal is to select a generalization concept from a certain class of possible concepts such that the selected concept has low generalization error with very high probability. In our case, the concept class is the set of all possible halfspaces in $\mathbb{R}^n$ and the concept to be learnt is the halfspace that is the correct guard in the optimal switching logic. The points in a concept to be learnt are the states in the guard and the points outside the concept are the states outside the guard.

We briefly summarize results from learning theory that establish the efficient learnability of halfspaces in the PAC learning framework [4]. A concept class is said to *shatter* a set of points if for any classification of the points as positive and negative, there is some concept in the class which would correctly classify the points. Any concept class is associated with a combinatorial parameter of the class, namely, the Vapnik-Chervonenkis (VC) dimension defined as the cardinality of the largest set of points (arbitrarily labeled as positive or negative) that can be shattered. For example, consider the concept class to be partitions in $\mathbb{R}^2$ using straight lines, that is, halfspaces in $\mathbb{R}^2$. The straight line should separate positive points in the true concept and negative points outside the concept. There exist sets of 3 points that can indeed be shattered using this model; in fact, any 3 points that are not collinear can be shattered, no matter how one labels them as positive or negative. However, it can be shown using Radon's theorem that no set of 4 points can be shattered [14]. Thus, the VC dimension of straight lines is 3. In general, the VC dimension for halfspaces in $\mathbb{R}^n$ is known to be $n+1$ [4]. The following theorem from Blumer et al [4] establishes the relation between efficient learnability of a concept class in the PAC learning framework and VC dimension of the concept class.

THEOREM 3. *Let $\mathbf{C}$ be a concept class with a finite VC dimension d. Then, any concept in $\mathbf{C}$ can be learnt in the following sense: with probability at least $1-\delta$ a concept C is learnt which incorrectly labels a point with a probability of at most ϵ, where C is generated using a random sample of labeled points of size at least*

$$\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{13}{\epsilon}\right)$$

.

Since the VC dimension for the class of halfspaces in $\mathbb{R}^n$ is $n+1$, a halfspace can be learnt in PAC learning framework using a sample of size at least

$$\max\left(\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8n+8}{\epsilon}\log\frac{13}{\epsilon}\right)$$

This, learning halfspaces in $\mathbb{R}^n$ requires samples polynomial in n, $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ and by increasing the probabilistic accuracy of the learnt halfspace requires polynomial increase in the number of samples. This is critical for efficiently learning guards in our algorithm.

Halfspace learning algorithm: We first discuss an algorithm `HSinfer` which can be used to learn halfspaces in the PAC framework from a given sample and then, describe the switching logic synthesis algorithm. In $\mathbb{R}^n$, a halfspace is given by $\theta \cdot X + \theta_0 \geq 0$ where $\theta \in \mathbb{R}^n, \theta_0 \in \mathbb{R}$ and X is any point in $\mathbb{R}^n$ which satisfied the above inequality if and only if the point is in the concept halfspace to be learnt. For any point X_i, let Y_i be 1 if X_i is in the concept and -1 if it is outside the concept. The algorithm below is the standard *Perceptron Learning* algorithm and is known to converge after k iterations where $k \leq (max_i||X_i||)/(\min_i \frac{Y_i(\theta X_i + \theta_0)}{||\theta||})^2$ [11].

Learning guards: We now describe the algorithm to learn guards for multiple initial states using the technique presented in Section 5 and the halfspace learning algorithm

Input: Set of labelled sample points $\{(X_i, Y_i)\}$
Output: θ, θ_0 such that $\theta X + \theta_0 \geq 0$ is the halfspace
Set $\theta^0 = 0, \theta_0^0 = 0, t = 0$;
for *each i* **do**
  if $\theta^0 X + \theta_0^0 \geq 0$ **then**
    Predicted $y_i = 1$
  else
    Predicted $y_i = -1$
  end
end
while *some i has* $Y_i \neq y_i$ **do**
  pick some i with $Y_i \neq y_i$;
  $\theta^{t+1} := \theta^t + Y_i X_i$, $\theta_0^{t+1} := \theta_0^t + Y_1$, $t := t + 1$;
  for *each i* **do**
    if $\theta^t X + \theta_0^t \geq 0$ **then**
      Predicted $y_i = 1$
    else
      Predicted $y_i = -1$
    end
  end
end
return θ, θ_0

Algorithm 1: Halfspace learning algorithm `HSinfer` [11]

`HSinfer`. The algorithm simply involves finding optimal switching points for each mode-switch and then using halfspace learning to infer the guards. The key idea is to use the optimal switching states as *positive* points for the concept learning problem and the non-optimal states explored during optimization (which preceded the optimal switching states along any trajectory) as *negative* points since these states cannot be in the guard for an optimal switching logic.

Guarantees: Under the structural hypothesis that guards are halfspaces, our PAC learning algorithm algorithm computes guards with probability atleast $1 - \delta$ such that the probability that a guard contains any state which is not a switching-state or misses any switching-state is at most ϵ. Further, the guards inferred by the above algorithm can be made probabilistically more and more accurate by choosing suitable values of ϵ, δ and considering correspondingly larger and larger samples of initial states as given by Theorem 3. For a trajectory to be non-optimal, at least one switching point in the trajectory needs to be classified incorrectly. Thus, the following theorem establishes the probabilistic guarantees of our switching logic synthesis algorithm.

THEOREM 4. *Given a* $\mathtt{MDS}(X, Q)$, *using random sampling from the set of initial states which has a sample size polynomial in* $n, \frac{1}{\epsilon}$ *and* $\frac{1}{\delta}$, *Algorithm 2 synthesizes a switching logic* `SwL` *with probability at least* $1 - \delta$ *such that any trajectory in the synthesized hybrid system* $\mathtt{HS}(\mathtt{MDS}, \mathtt{SwL})$ *is not optimal with probability at most* $m\epsilon$, *where* m *is the number of guards in the switching logic, that is,* $m = |\mathtt{SwL}| \leq |Q|^2$ *and* n *is the number of variables, that is,* $n = |X|$.

Running Example

Given the set of initial states $16 \leq \mathtt{temp} \leq 26$ and $\mathtt{out} \in \{16, 26\}$. The set of initial states is partitioned into subsets where each subset is a 0.1 interval of room temperature `temp` and the outside temperature is 16 or 26. The guards discovered are: $\mathtt{g}_{HF} : \mathtt{temp} \geq 20.2 \wedge \mathtt{out} = 16$, $\mathtt{g}_{FH} : \mathtt{temp} \leq 19.6 \wedge \mathtt{out} = 16$, $\mathtt{g}_{CF} : \mathtt{temp} \leq 20.0 \wedge \mathtt{out} = 26$, $\mathtt{g}_{FC} : \mathtt{temp} \geq 20.3 \wedge \mathtt{out} = 26$.

Input: $MDS(X, Q)$, initial states I, tolerance of generalization error δ and maximum probability of error ϵ
Output: Optimal Switching Logic SL^{opt}
1. Sample initial states from I for provided δ, ϵ.
2. For each initial state, obtain optimal trajectory in $MDS(X, Q)$ and switching states for the mode switches along the trajectory.
3. Label the obtained switching states as positive points.
4. Label the states preceding switching states along any trajectory to be negative points.
5. Using obtained sample of positive and negative states, learn the guard for the mode switch as generalization of these states using `HSinfer`.
6. Output these guards as synthesized optimal switching logic SL^{opt}.

Algorithm 2: Finding optimal switching logic SL^{opt} with single initial state

7. CASE STUDIES

Apart from the running example of Thermostat controller, we applied our technique to two other case studies: (i) an Oil Pump Controller, which is an industrial case study from [7], and (ii) a DC-DC Buck-Boost Converter, motivated by the problem of minimizing voltage fluctuation in a distributed aircraft power system. We employ the implementation of Nelder-Mead simplex algorithm as described by Lagarias et al [21, 26] and available as the *fminsearch* [23] function in MATLAB for numerical optimization. All experiments were performed on a dual-core 1.66GHz processor with 4GB memory.

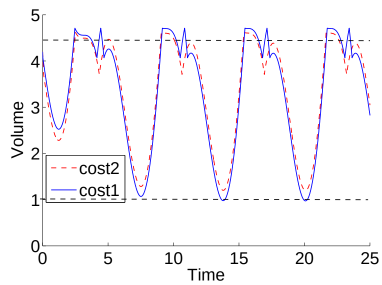
7.1 Thermostat Controller

If we change the cost metric in the thermostat controller to $\lim_{t\to\infty} \frac{\mathtt{discomfort}(t)+\mathtt{fuel}(t)+\mathtt{swTear}(t)}{\mathtt{time}(t)}$ giving equal weight to all the three penalties (instead of $10 : 1 : 1$ weight ratio used earlier) the optimal switching logic discovered with this cost metric are: $\mathtt{g}_{HF} : \mathtt{temp} \geq 20.0 \wedge \mathtt{out} = 16$, $\mathtt{g}_{FH} : \mathtt{temp} \leq 18.8 \wedge \mathtt{out} = 16$, $\mathtt{g}_{CF} : \mathtt{temp} \leq 21.9 \wedge \mathtt{out} = 26$, $\mathtt{g}_{FC} : \mathtt{temp} \geq 22.7 \wedge \mathtt{out} = 26$. We observe that the room temperature oscillates closer to the target temperature when the discomfort penalty is given relatively higher weight in the cost metric. This case study illustrates that a designer can suitably define a cost metric which reflects their priorities and, then, our technique can be used to automatically synthesize switching logic for the given cost metric. The runtime for simulation and numerical optimization is 284 seconds.

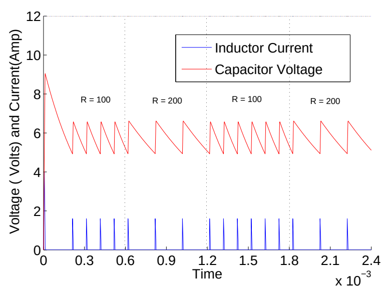
7.2 Oil Pump Controller

Our second case study is an Oil Pump Controller, adapted from the industrial case study in [7]. The example consists of three components - a machine which consumes oil in a periodic manner, a reservoir containing oil, an accumulator containing oil and a fixed amount of gas in order to put the oil under pressure, and a pump. The simplification we make is to use a periodic function to model the machine's oil consumption and we do not model any noise (stochastic variance) in oil consumption.

The state variable is the volume V of oil in the accumulator. The system has two modes: mode `ON` when the pump is ON and mode `OFF` when the pump is OFF. Let



(a) Oil Pump Controller: Volume in accumulator



(b) DC-DC Boost Converter

Figure 2: Case Studies

the rate of consumption of oil by the machine be given by $m = 3 * (cos(t) + 1)$ where t is the time. The rate at which oil gets filled in the accumulator is p. $p = 4$ when the pump is on and $p = 0$ when the pump is off. The change in volume of oil in the accumulator is given by the following equation $\dot{V} = p - m$ where p and m take different values depending on the mode of operation of the pump. For synthesis, we consider two different sets of requirements [7].

In the first set of requirements, the volume of oil in the tank must be within some safe limit, that is, $1 \leq V \leq 8$ and the average volume of oil in the accumulator should be minimized. We model these requirements using our cost definition by defining one penalty variable p_1 and one reward variable r_1. Let the evolution of penalty p_1 be $\dot{p}_1 = V$ if $1 \leq V \leq 8$, M otherwise where M is a very large ($M \geq 10^5 p_1$) constant (10^6 in our experiments) and that of reward r_1 be $\dot{r}_1 = 1$. Minimizing the cost function $cost1 = \lim_{t\to\infty} \frac{p_1(t)}{r_1(t)}$ minimizes the average volume $\lim_{t\to\infty} \frac{\int_0^t V(t)}{t}$ and also enforces the safety requirement $\forall t \,.\, 1 \leq V(t) \leq 8$.

In the second set of requirements, we add an additional requirement to those in the first set. We require that the the oil volume is mostly below some threshold $V_{high} = 4.5$ in the long run. We model this requirement by adding an additional penalty and an additional reward variable p_2 and r_2 with evolution functions: $\dot{p}_2 = 1$ if $V > V_{high}$, 0 otherwise and $\dot{r}_2 = 1$ if $V < V_{high}$, 0 otherwise. The new cost function is $cost2 = \lim_{t\to\infty}(\frac{p_1(t)}{r_1(t)} + \frac{p_2(t)}{r_2(t)})$. Let t_{high} be the total duration when the volume is above V_{high} and t_{low} be the duration that it is below V_{high}. Minimizing $p_2/r_2 = t_{high}/t_{low}$ would ensure that we spend more time with volume less than V_{high} in the accumulator.

The guards: $\mathtt{g}_{FN}$ from `OFF` to `ON` and $\mathtt{g}_{NF}$ from `ON` to `OFF` obtained for the above $cost1$ objective are $\mathtt{g}_{FN} : V \leq 3.71$ $\mathtt{g}_{NF} : V \geq 4.62$ and for $cost2$ objective are $\mathtt{g}_{FN} : V \leq 4.07$ $\mathtt{g}_{NF} : V \geq 4.71$. The runtimes are 438 seconds and 479 seconds respectively.

We simulate from an initial state $V = 4$ and the behavior for both objectives is presented in Figure 2(a). In both cases, the behavior satisfies the safety property that the volume is within 1 and 8. Since, we minimize oil volume, the volume is close to the lower limit of 1. We also observe that using the second cost metric causes decrease in duration of time when oil volume is higher than the 4.5 but the average volume of oil increases. This illustrates how designers can use different cost metrics to better reflect their requirements.

7.3 DC-DC Buck-Boost Converter

In this case study, we synthesize switching logic for controlling DC-DC buck-boost converter circuits described in [15]. The goal is to maintain the output voltage V_R across a varying load R at some target voltage V_d. The converter can be modeled as a hybrid system with three modes of operation. The state space of the system is $X = \langle i_L \; u_C \rangle$ where i_L is the current through the inductor and u_C is the load voltage. The dynamics in the three modes are given by the state space equation $\dot{X} = A_k X + B_k E$ where $k = 1, 2, 3$ is the mode and E is the input voltage. The coefficients of the equations are

$$A_1 = \begin{bmatrix} \frac{-rL-rs}{L} & 0 \\ 0 & \frac{-1}{C*(R+rC)} \end{bmatrix}, \quad B_1 = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} \frac{-rL-rd}{L} & \frac{-1}{L} \\ \frac{R}{R+rC} * (\frac{1}{C} - \frac{rC*(rL+rd)}{L}) & \frac{-R}{R+rC} * (\frac{rC}{L} + \frac{1}{R*C}) \end{bmatrix},$$

$$B_2 = \begin{bmatrix} \frac{1}{L} \\ \frac{R}{(R+rC)*\frac{rC}{L}} \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 0 \\ 0 & \frac{-1}{(R+rC)*C} \end{bmatrix}, \quad B_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We mention two key performance requirements of the DC-DC Boost Converter described in [15]. The first requirement is that the converter be resilient to load variations. The second requirement is to minimize the variance of the voltage across the load V_R from the target voltage. This variance is called the ripple voltage. We define penalty variable p_1 with the following evolution functions: $\dot{p}_1 = (V_R - V_d)^2$. We want to minimize the average deviation from the target voltage. So, we define the reward variable r_1 with $\dot{r}_1 = 1$. The cost function is $\mathtt{cost} = \lim_{t\to\infty} \frac{p_1(t)}{r_1(t)}$. This minimizes the average variance of V_R from the target voltage V_d. This corresponds to minimizing the ripple voltage. Since the load R also changes periodically, it also minimizes the transient variance in voltage.

Given the dynamics in each of the three modes and the cost function, the synthesis problem is to automatically synthesize the guards $\mathtt{g}_{12}, \mathtt{g}_{23}, \mathtt{g}_{31}$ which minimizes the cost. We are given the over-approximation of the guard $\mathtt{g}_{23}^{over} : i_l = 0$. The guards obtained are as follows: $\mathtt{g}_{12} : i_L > 1.9$, $\mathtt{g}_{23} : i_L = 0$ and $\mathtt{g}_{23} : v_C > 4.6$. The runtime is 394 seconds. The system remains in the first mode until the inductor current reaches the reference current I_{ref}. The system remains in the second mode until the inductor current becomes 0. Then, the system switches to the third mode where it remains as long as the capacitor voltage remains over the reference voltage V_{ref}. We simulate the synthesized system and the behavior is shown in Figure 2(b).

8. CONCLUSION

In this paper, we present an algorithm for automated synthesis of switching logic in order to achieve minimum long-run cost. Our algorithm is based on reducing the switching logic synthesis problem to an unconstrained numerical optimization problem which can then be solved by existing optimization techniques. We also give a learning-based approach to generalize from a sample of switching states to a switching condition, where the learnt condition is optimal with high probability.

9. REFERENCES

[1] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proc. of the IEEE*, 88(7):1011–1025, 2000.

[2] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest. Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. *J. Optimization Theory and Applications*, 136(2):167–186, 2008.

[3] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest. Gradient descent approach to optimal mode scheduling in hybrid dynamical systems. *Journal of Optimization Theory and Applications*, 136:167–186, 2008.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[5] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization – Theoretical and Practical Aspects, Second Edition*. 2006.

[6] M. Branicky, V. Borkar, and S. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Trans. on Aut. Cntrl.*, 43(1):31–45, 1998.

[7] F. Cassez, J. J. Jessen, K. G. Larsen, J.-F. Raskin, and P.-A. Reynier. Automatic synthesis of robust and optimal controllers - an industrial case study. In *HSCC*, pages 90–104, 2009.

[8] K. Chatterjee. Markov decision processes with multiple long-run average objectives. In *FSTTCS*, pages 473–484, 2007.

[9] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Controller synthesis with budget constraints. In *HSCC*, pages 72–86, 2008.

[10] J. Cury, B. Brogh, and T. Niinomi. Supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Aut. Cntrl.*, 43:564–568, 1998.

[11] Y. Freund and R. E. Schapire. Large Margin Classification Using the Perceptron Algorithm. In *Machine Learning*, pages 277–296, 1998.

[12] H. González, R. Vasudevan, M. Kamgarpour, S. Sastry, R. Bajcsy, and C. Tomlin. A numerical method for the optimal control of switched systems. In *CDC*, pages 7519–7526, 2010.

[13] H. Gonzalez, R. Vasudevan, M. Kamgarpour, S. S. Sastry, R. Bajcsy, and C. J. Tomlin. A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In *HSCC*, pages 51–60, 2010.

[14] P. M. Gruber and J. M. Wills, editors. *Handbook of Convex Geometry : Two-Volume Set*. North Holland, 1993.

[15] P. Gupta and A. Patra. Super-stable energy based switching control scheme for dc-dc buck converter circuits. In *ISCAS (4)*, pages 3063–3066, 2005.

[16] S. Jha, S. Gulwani, S. Seshia, and A. Tiwari. Synthesizing switching logic for safety and dwell-time requirements. In *ICCPS*, pages 22–31, 2010.

[17] S. Jha, S. A. Seshia, and A. Tiwari. Full version: Synthesizing switching logic to minimize long-run cost. `http://eecs.berkeley.edu/~jha/optSLsynth.pdf`.

[18] R. Karp. A characterization of the minimum cycle mean in a digraph. *Dis. Math.*, pages 309–311, 1978.

[19] T. Koo and S. Sastry. Mode switching synthesis for reachability specification. In *Proc. HSCC 2001*, LNCS 2034, pages 333–346, 2001.

[20] A. Kucera and O. Strazovsky. On the controller synthesis for finite-state markov decision processes. *Fundamenta Informaticae*, 82(1-2):141–153, 2008.

[21] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[22] P. Manon and C. Valentin-Roubinet. Controller synthesis for hybrid systems with linear vector fields. In *Proc. IEEE Intl. Symp. on Intelligent Control*, pages 17–22, 1999.

[23] MathWorks. Minimum of unconstrained multivariable function using derivative-free method. `http://www.mathworks.com/help/techdoc/ref/fminsearch.html`.

[24] MathWorks. Solve initial value problems for ordinary differential equations. `http://www.mathworks.com/help/techdoc/ref/ode23.html`.

[25] T. Moor and J. Raisch. Discrete control of switched linear systems. In *ECC'99*, 1999.

[26] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, January 1965.

[27] J. I. Rasmussen, K. G. Larsen, and K. Subramani. On using priced timed automata to achieve optimal scheduling. *FMSD*, 29:97–114, July 2006.

[28] M. Shaikh and P. Caines. On the optimal control of hybrid systems: Optimization of trajectories, switching times, and location schedules. In *HSCC*, 2003.

[29] P. Tabuada. Controller synthesis for bisimulation equivalence. *Systems and Control Letters*, 57(6):443–452, 2008.

[30] A. Taly and A. Tiwari. Switching logic synthesis for reachability. In *EMSOFT*, pages 19–28, 2010.

[31] C. Tomlin, L. Lygeros, and S. Sastry. A game-theoretic approach to controller design for hybrid systems. *Proc. of the IEEE*, 88(7):949–970, 2000.

[32] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[33] X. Xu and P. Antsaklis. Optimal control of switched systems via nonlinear optimization based on direct differentiation of value functions. *Intl. journal of control*, 75(16):1406–1426, 2002.