2016 IEEE First International Conference on Internet-of-Things Design and Implementation

Control Improvisation with Probabilistic Temporal Specifications

Ilge Akkaya, Daniel J. Fremont, Rafael Valle, Alexandre Donzé, Edward A. Lee, and Sanjit A. Seshia
Department of Electrical Engineering and Computer Science
UC Berkeley
Berkeley, CA, USA
{*ilgea, dfremont, rafaelvalle, donze, eal, sseshia*}*@berkeley.edu*

***Abstract*—We consider the problem of generating randomized control sequences for complex networked systems typically actuated by human agents. Our approach leverages a concept known as control improvisation, which is based on a combination of data-driven learning and controller synthesis from formal specifications. We learn from existing data a generative model (for instance, an explicit-duration hidden Markov model, or EDHMM) and then supervise this model in order to guarantee that the generated sequences satisfy some desirable specifications given in Probabilistic Computation Tree Logic (PCTL). We present an implementation of our approach and apply it to the problem of mimicking the use of lighting appliances in a residential unit, with potential applications to home security and resource management. We present experimental results showing that our approach produces realistic control sequences, similar to recorded data based on human actuation, while satisfying suitable formal requirements.**

***Keywords*-control improvisation; randomized control; data-driven modeling; automated lighting**

I. Introduction

The promise of the emerging Internet of Things (IoT) is to leverage the programmability of connected devices to enable applications such as connected smart vehicles, occupancy-based automated HVAC control, autonomous robotic surveillance, and much more. However, this promise cannot be realized without better tools for the automated programming and control of a network of devices — computational platforms, sensors, and actuators. Traditionally, this problem has been approached from two different angles. The first approach is to be *data-driven*, leveraging the ability of devices and sensors to collect vast amounts of data about their operation and environments, and using learning algorithms to adjust the working of these devices to optimize desired objectives. This approach, exemplified by devices such as smart learning thermostats, can be very effective in many settings, but typically cannot give any guarantees of correct operation. The second approach is to be *model-driven*, where accurate models of the devices and their operating environment are used to define a suitable control problem. A controller is then synthesized to guarantee correct operation under specified environment conditions. However, such an approach is difficult in settings where such accurate models are hard to come by. Moreover, strong correctness guarantees may not be needed in all cases.

Consider, for instance, the application domain of home automation. More specifically, consider a scenario where one is designing the controller for a home security system that controls the lighting (and possibly other appliances) in a home when the occupants are away. One might want to program the system to mimic typical human behavior in terms of turning lights on and off. As human behavior is somewhat random, varying day to day, one might want the controller to exhibit random behavior. However, completely random control may be undesirable, since the system must obey certain time-of-day behavioral patterns, and respect correlations between devices. For these requirements, a data-driven approach where one learns a randomized controller mimicking human behavior seems like a good fit. It is important to note, though, that such an application may also have constraints for which provable guarantees are needed, such as limits on energy consumption being obeyed with high probability, or that multiple appliances never be turned on simultaneously. A model-based approach is desirable for these. Thus, the overall need is to *blend data and models* to synthesize a control strategy that obeys certain *hard constraints* (that must always be satisfied), certain *soft constraints* (that must be "mostly satisfied") and certain *randomness requirements* on system behavior.

This setting has important differences from typical control problems. For example, in traditional supervisory control, the goal is typically to synthesize a control strategy ensuring that certain mathematically-specified (formal) requirements hold on the entity being controlled (the "plant"). Moreover, the generated sequence of control inputs is typically completely determined by the state of the plant. Predictability and correctness guarantees are important concerns. In contrast, in the home automation application sketched above, predictability is not that important. Indeed, the system's behavior must be random, within constraints. Moreover, the source of randomness (behavior of human occupants) differs from home to home, and so this cannot be pre-programmed.

This form of randomized control is suitable for human-in-the-loop systems or applications where randomness is desirable for reasons of security, privacy, or diversity. Application domains other than the home automation setting described above include microgrid scheduling [1], [2] and robotic art [3]. In the former, randomness can provide some diversity of

978-1-4673-9948-7/16 $31.00 © 2016 IEEE
DOI 10.1109/IoTDI.2015.33

load behavior, hence making the grid more efficient in terms of peak power shaving and more resilient to correlated faults or attacks on deterministic behavior. For the latter case, there is growing interest in augmenting human performances with computational aids, such as in automatic synthesis and improvisation of music [4]. All these applications share the property of requiring some randomness while maintaining behavior within specified constraints. Additionally, the human-in-the-loop applications can benefit from data-driven methods. Streams of time-stamped data from devices can be used to learn semantic models capturing behavioral correlations amongst them for further use in programming and control.

In this paper, we show how a recently-proposed formalism termed *control improvisation* [5] can be suitably adapted to address the problem of randomized control for IoT systems. We consider the specific setting of a system whose components can be controlled either by humans or automatically. Human control of devices generates data comprising streams of time-stamped events. From such data, we show how one can learn a nominal randomized controller respecting certain constraints present in the data including correlations between behavior of interacting components. We also show how additional constraints can be enforced on the output of the controller using temporal logic specifications and verification methods based on model checking [6], [7]. We apply our approach to the problem of randomized control of home appliances described above. We present simulated experimental results for the problem of lighting control based on data from the UK Domestic Appliance-Level Electricity (UK-DALE) dataset [8]. Our approach produces realistic control sequences, similar to recorded data based on human actuation, while also satisfying suitable formal requirements.

II. Background

We introduce relevant background material that the present paper builds upon and establish notation for use in the rest of the paper.

A. Discrete-Event Systems with Hidden States

Our work focuses on control of systems whose behavior can be described by a sequence of timestamped *events*. An event e is a tuple $\langle \tau, v \rangle \in T \times V$, where T is a totally ordered set of time stamps and V is a finite set of values. We define a *signal* to be a set of events, where T imposes an ordering relation on the events occurring within the signal [9].

We define the *state* of such a system to take values from a finite set of distinct states, where *events* are emitted by state transitions. In many systems, the underlying events and states are hidden, and all that can be observed is some function of the state. We term this the *observation*. This function can be time-dependent and probabilistic, so that a single state can produce many different observations.

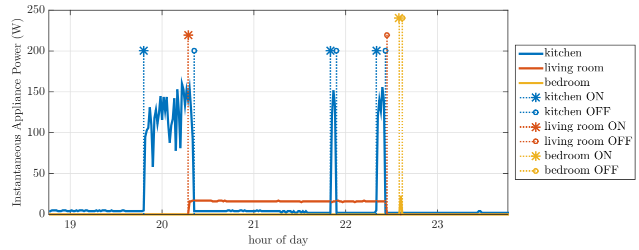

Figure 1: Sample appliance power trace

We assume that the number of possible observation values is finite (this can be enforced in continuous systems by discretization), and that observations are made at discrete time steps. A sequence of observations over time generated by a behavior of the system is called a *trace*.

An example of such a trace that captures the power consumption of three appliances is given in Figure 1. The events related to each appliance, which can either be an "ON" or an "OFF" event in this case, are annotated on the sub-traces. Each state change of the system triggers an event. Consider, for example, that the hidden state in this scenario captures the current status of a set of physical appliances and that all appliances are initially turned off. The kitchen appliance being turned on at 19:50 pm causes an "ON" event to be emitted, and triggers a state change in the system, where in the new state, the kitchen appliance is on, and the other two appliances are off. The system stays in this state until any appliance triggers a state transition. In such a scenario, it may be that the only information available from the system are traces of the instantaneous appliance power consumptions. Given these traces, one can *infer* the state of the system and which events may have happened at particular times.

B. Control Improvisation

The *control improvisation problem*, defined formally in [5], can be seen as the problem of generating a random sequence of control actions subject to *hard* and *soft* constraints, while satisfying a *randomness requirement*. The hard constraints may, for example, encode safety requirements on the system that must always be obeyed. The soft constraints can encode requirements that may occasionally be violated. The randomness requirement ensures that no control sequence occurs with too high probability.

This problem is a natural fit to the applications of interest in this paper, as our end goal is to randomize the control of discrete-event systems subject to both constraints enforcing the presence of certain learned behaviors (hard constraints), and probabilistic requirements upper bounding the observations (soft constraints). In the lighting control scenario we consider later, for example, we effectively learn a hard constraint preventing multiple appliances from being toggled at exactly the same time, since this never occurs

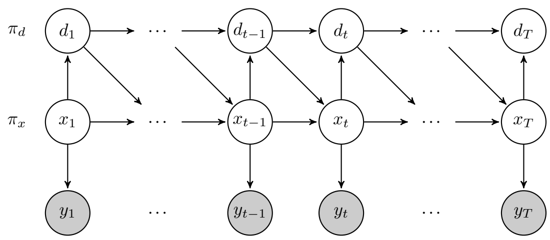

Figure 2: Graphical model representation of an EDHMM

in the training data. We also use soft constraints to limit the probability that the hourly power consumption exceeds desired bounds.

More formally, the control improvisation problem is defined as follows. This is generalized from the definition in [5] to allow multiple soft constraints with different probabilities.

Definition 1: An instance of the control improvisation (CI) problem is composed of (i) a language I of *improvisations* that are strings over a finite alphabet Σ, i.e., $I \subseteq \Sigma^*$, and (ii) finitely many subsets $\mathcal{A}_i \subseteq I$ for $i \in \{1, \dots, n\}$. These sets can be presented, for example, as finite automata, but for our purposes in this paper the details are unimportant (see [5] for a thorough discussion).

Given error probability bounds $\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)$ with $\epsilon_i \in [0, 1]$, and a probability bound $\rho \in (0, 1]$, a distribution $D : \Sigma^* \to [0, 1]$ with support set S is an $(\boldsymbol{\epsilon}, \rho)$-*improvising distribution* if

(a) $S \subseteq I$ (hard constraints),

(b) $\forall w \in S,\ D(w) \leq \rho$ (randomness),

(c) $\forall i, P[w \in \mathcal{A}_i \mid w \leftarrow D] \geq 1 - \epsilon_i$ (soft constraints),

where $w \leftarrow D$ indicates that w is drawn from the distribution D. An $(\boldsymbol{\epsilon}, \rho)$-*improviser*, or simply an *improviser*, is a probabilistic algorithm generating strings in Σ^* whose output distribution is an $(\boldsymbol{\epsilon}, \rho)$-improvising distribution. For example, this algorithm could be a Markov chain generating random strings in Σ^*. The control improvisation problem is, given the tuple $(I, \{\mathcal{A}_i\}, \boldsymbol{\epsilon}, \rho)$, to generate such an improviser.

C. Explicit-Duration Hidden Markov Models

In data-driven controller synthesis, it is essential that the learning model captures relevant properties of the underlying system based on observed data. For probabilistic inference in dynamical systems whose state is only observable via state-dependent data, Hidden Markov Models (HMM) have been a widely used tool. An HMM is characterized by a hidden state variable subject to Markov dynamics, observable via state-dependent noisy observations. However, in many applications, including those of interest to this paper, the Markov assumption on the hidden state space is insufficient, since part of the underlying problem structure lies in the *durations* of events. In such scenarios, the model quality can be improved significantly by the use of semi-Markov models. In this study, we will specifically consider the Explicit-Duration Hidden Markov Model (EDHMM) [10]. These models are an extension of HMMs that, in addition to modeling the hidden state space as a Markov chain, also introduces the duration spent within each state as another hidden variable of the Bayesian network. The graphical model representation of a general EDHMM is shown in Figure 2.

The standard definition of an EDHMM models hidden state and its duration to be discrete hidden variables. The state dependent observations can be drawn from either a discrete or continuous distribution, often referred to as an *emission* distribution. In this paper, we assume that the possible observations are quantized as necessary so that the emission distributions are discrete.

An EDHMM with discrete emissions observed for T time steps is characterized by a partially observed set of variables $(\mathbf{x}, \mathbf{d}, \mathbf{y}) = (x_1, \dots, x_T,\ d_1, \dots, d_T,\ y_1, \dots, y_T)$. Each x_i indicates the hidden state of the model at time i from a finite state space $\mathcal{X}$, which for notational convenience we assume to be the set $\{1, \dots, N\}$. The value $d_i \in \{1, \dots, D\}$ denotes the remaining duration in the hidden state, where D is the maximum possible state duration. Finally, each y_i is an observation drawn from a discrete alphabet $\Sigma = \{v_1, \dots, v_M\}$. The joint probability distribution imposed by the EDHMM over these variables can be written as

$$\begin{aligned} P(\mathbf{x}, \mathbf{d}, \mathbf{y}) &= p(x_1)p(d_1)\prod_{t=2}^{T} p(x_t|x_{t-1}, d_{t-1})p(d_t|d_{t-1}, x_t)\prod_{t=1}^{T} p(y_t|x_t) \\ &= \pi_x \pi_d \prod_{t=2}^{T} p(x_t|x_{t-1}, d_{t-1})p(d_t|d_{t-1}, x_t)\prod_{t=1}^{T} p(y_t|x_t), \end{aligned}$$

where $p(x_1) \triangleq \pi_x$ and $p(d_1) \triangleq \pi_d$ are the priors on the hidden state and duration distributions, respectively. The conditional state and duration dynamics are given by

$$p(x_t|x_{t-1}, d_{t-1}) \triangleq \begin{cases} p(x_t|x_{t-1}) & \text{if } d_{t-1} = 1 \\ \delta(x_t, x_{t-1}) & otherwise \end{cases} \tag{1}$$

$$p(d_t|d_{t-1}, x_t) \triangleq \begin{cases} p(d_t|x_t) & \text{if } d_{t-1} = 1 \\ \delta(d_t, d_{t-1} - 1) & otherwise \end{cases}, \tag{2}$$

where $\delta(\cdot, \cdot)$ is the Kronecker delta function. Equations (1) and (2) specify the current state x_t and the remaining duration d_t for that state as a function of the previous state and its remaining duration. Unless the remaining duration at the previous state is equal to 1, the state will remain unchanged across time steps, while at each step within the state, the remaining duration is decremented by 1. When the remaining duration is 1, the next state is sampled from a transition probability distribution $p(x_t|x_{t-1})$, while the remaining duration at x_t is sampled from a state-dependent duration distribution $p(d_t|x_t)$. All self-transition probabilities are set to zero: $p(x_t|x_{t-1}) = 0$ if $x_t = x_{t-1}$. For compactness of notation, for all $x_t, x_{t-1} \in \{1, \dots, N\}$, $d_t \in \{1, \dots, D\}$, and $y_t \in \{v_1, \dots, v_M\}$ we define probabilities a_{x_{t-1},x_t}, b_{x_t,y_t}, and c_{x_t,d_t} so that

$$p(x_t|x_{t-1}) = \begin{cases} a_{x_{t-1},x_t} & \text{if } x_t \neq x_{t-1} \\ 0 & \text{otherwise} \end{cases},$$
$$p(y_t|x_t) = b_{x_t,y_t},$$
$$p(d_t|x_t) = c_{x_t,d_t}.$$

We consolidate these probabilities into an $N \times N$ state transition matrix $A \triangleq (a_{ij})$, an $N \times M$ emission probability matrix $B \triangleq (b_{ij})$, and an $N \times D$ duration probability matrix $C \triangleq (c_{ij})$.

The procedure to obtain the EDHMM parameter set $\lambda = \{\pi_x, \pi_d, A, B, C\}$, given the observed sequence $\boldsymbol{y}$, is often referred to as the *parameter estimation* problem, which in the general Bayesian inference setting seeks to assign the parameters of a model so that it best explains given training data. More precisely, given a trace $(y_1, \dots, y_T)$, parameter estimation approximates the optimal parameter set λ^* such that

$$\lambda^* = \arg\max_{\lambda} p(y_1, \dots, y_T \mid \lambda).$$

This procedure can be extended to estimate parameters from multiple traces, provided that the traces are aligned so that the first observation of each trace corresponds to the same initial state. This ensures that the state prior will be correctly captured [10]. In the case of the EDHMM, parameter estimation can be done with a variant of the well-known Expectation-Maximization (EM) algorithm for HMM. The detailed formulation is presented in [11].

1) EDHMM with Non-homogeneous Hidden Dynamics: The general definition of an EDHMM is useful in modeling hidden state dynamics encoded with explicit duration information. However, in many applications where the state dynamics model behaviors that exhibit seasonality, it can be useful to train separate state transition and duration distributions for different time intervals. As an example, we consider the case where the dynamics exhibit a dependence on the hour of the day, so that for each hour $h \in \{1, \dots, 24\}$ we have different probability matrices A_h and C_h.

Estimating the parameters of an EDHMM with hourly dynamics requires an additional input sequence $\{h_1, \dots, h_T\}$, where each $h_i \in \{1, ..., 24\}$ labels at which hour of the day the observation y_i was collected. Given the observation and hour label streams, training follows the same EM-based estimation procedure as in [11], with the difference that parameters A_h and C_h are estimated using the training data subsequences collected within hour h.

The EDHMM with hourly dynamics will be given by a parameter set $\lambda = \{\pi_x, \pi_d, \{A_h\}, B, \{C_h\}\}$, where $\{A_h\}$ and $\{C_h\}$ are the transition and duration distribution matrices valid for hour $h \in \{1, ..., 24\}$ such that

$$a^l_{i,j} \triangleq p(x_t = j|x_{t-1} = i, d_{t-1} = 1, h_{t-1} = l)$$
$$c^l_{i,d} \triangleq p(d_t = d|x_t = i, d_{t-1} = 1, h_t = l)$$

where $A_l = (a^l_{ij})$ and $C_l = (c^l_{i,d})$ are the hourly transition and duration probability matrices for hour l.

D. Probabilistic Model Checking

Our approach relies on the use of a verification method known as probabilistic model checking, which determines if a probabilistic model (such as a Markov chain or Markov decision process) satisfies a formal specification expressed in a probabilistic temporal logic. We give here a high-level overview of the relevant concepts for this paper. The reader is referred to the book by Baier and Katoen [6] for further details. For our application, we employ probabilistic computation tree logic (PCTL). The syntax of this logic is as follows:

$$\phi ::= True \mid \omega \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid P_{\bowtie p}[\psi] \quad \text{state formulas}$$
$$\psi ::= \mathcal{X}\phi \mid \phi_1\, \mathcal{U}^{\leq k}\phi_2 \mid \phi_1\, \mathcal{U}\phi_2 \quad \text{path formulas}$$

where $\omega \in \Omega$ is an atomic proposition, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0, 1]$ and $k \in \mathbb{N}$. State formulas are interpreted at states of a probabilistic model; if not specified explicitly, this is assumed to be the initial state of the model. Path formulas ψ use the *Next* $(\mathcal{X})$, *Bounded Until* $\left(\mathcal{U}^{\leq k}\right)$ and *Unbounded Until* $(\mathcal{U})$ operators. These formulas are evaluated over computations (paths) and are only allowed as parameters to the $P_{\bowtie p}[\psi]$ operator. Additional temporal operators, $\mathcal{G}$, denoting "globally", and $\mathcal{F}$ denoting "finally", are defined as follows: $\mathcal{F}\phi \triangleq True\,\mathcal{U}\,\phi$ and $\mathcal{G}\phi \triangleq \neg\,\mathcal{F}\,\neg\phi$.

We describe the semantics informally; the formal details are available in [6]. A path formula of the form $\mathcal{X}\phi$ holds on a path if state formula ϕ holds on the second state of that path. A path formula of the form $\phi_1\, \mathcal{U}^{\leq k}\phi_2$ holds on a path if the state formula ϕ_2 holds eventually at some state on that path within k steps of the first state, with ϕ_1 holding at every preceding state. The semantics of $\phi_1\, \mathcal{U}\phi_2$ is similar without the "within k steps" requirement. The semantics of state formulas is standard for all propositional formulas. The only case worth elaborating involves the probabilistic operator: $P_{\bowtie p}[\psi]$ holds at a state s if the probability q that path formula ψ holds for any execution beginning at s satisfies the relation $q \bowtie p$.

A probabilistic model checker, such as PRISM [7], can check whether a probabilistic model satisfies a specification in PCTL. Moreover, it can also compute the probability that a temporal logic formula holds in a model, as well as synthesize missing model parameters so as to satisfy a specification. We show in Sec. III-E how an EDHMM can be encoded as a Markov chain and thereby as a suitable input model to PRISM.

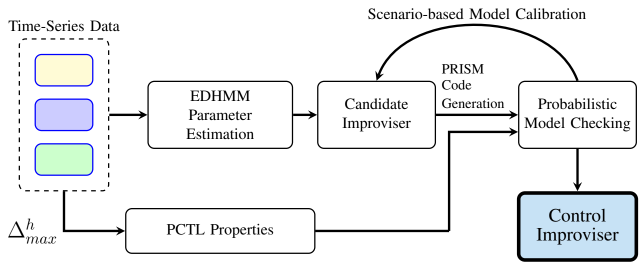

Figure 3: Algorithmic Workflow

III. Control Improvisation with Probabilistic Temporal Specifications

Now we define the problem tackled in this paper, and describe the approach we take to solve it.

A. Problem Definition and Solution Approach

We begin with a set of traces of a discrete-event system whose set of events is known, but whose dynamics are not. Our goal is to randomly generate new traces with similar characteristics to the given ones. Furthermore, we want to be able to enforce two kinds of constraints:

- *Hard* constraints that the traces must always satisfy, forbidding transitions between states that never occur in the input traces. For example, if no part of the input traces can be explained as a particular state transition t, then we want to assume that t is impossible and not generate any string that is only possible using it.
- *Soft* constraints that need only be satisfied with some given probability. We focus on systems whose observations are *costs*, for example power consumption, and assume soft constraints which put upper bounds on the cost at a particular time, or accumulated over a time period.

In the next section, we will formalize this problem as an instance of control improvisation. First, however, we summarize our solution approach, which consists of three main steps:

1) *Data-Driven Modeling:* From the given traces, learn an EDHMM representing the time-dependent dynamics of the underlying system. The EDHMM effectively applies hard constraints on our generation procedure by eliminating all strings assigned zero probability.
2) *Probabilistic Model Checking:* Using a probabilistic model checker, we compute the probability that a behavior of the candidate improviser obtained in the previous step will satisfy the soft constraints. If this is sufficiently high, we return the EDHMM as our generative model.
3) *Scenario-Based Model Calibration:* Otherwise, we apply heuristics that increase the probability by modifying the EDHMM parameters, and return to step (2).

A high level algorithmic workflow is given by Figure 3. We elaborate on each of the steps in subsequent sections.

B. Formalization as a Control Improvisation Problem

We can formalize the intuitive description above as an instance of the control improvisation (CI) problem described in Section II-B. To do so, we need to specify the alphabet Σ, languages $\mathcal{I}$ and $\mathcal{A}_i$, and parameters ϵ_i and ρ that make up a CI instance.

Σ — Since we are learning from and want to generate traces, which are sequences of observations, we let Σ be the set of all observations (i.e. those occurring anywhere in the input traces).

I — We let $\mathcal{I}$ consist of all traces that are assigned nonzero probability by the EDHMM[1]. Since the CI problem requires any improviser to output only strings in I, this will ensure the hard constraints are always satisfied.

$\mathcal{A}_i, \epsilon_i$ — We let $\mathcal{A}_i$ consist of all traces that satisfy the i-th soft constraint. For instance in the lighting example, $\mathcal{A}_i$ could only contain traces whose total power consumption within hour $i \in \{1, \ldots, 24\}$ of the day never exceeds a given bound. Then in the CI problem, ϵ_i is the greatest probability we are willing to tolerate of the improviser generating a trace violating the bound.

ρ — We can ensure that many different traces can be generated, and that no trace is generated too frequently, by picking a small value for ρ: the CI problem requires that no improvisation be generated with probability greater than ρ, and so that at least $1/\rho$ improvisations can be generated.

This CI problem captures the informal requirements we described earlier. Now we need to show that our generation procedure is actually an improviser solving this problem according to the three conditions given in Definition 1. We consider each in turn.

1) Hard Constraints: By definition, any string that we generate has nonzero probability according to the EDHMM and so is in $\mathcal{I}$.

2) Randomness Requirement: As long as the EDHMM is ergodic (when converted to an ordinary Markov chain; see Section III-E), the probability of generating any particular string $w \in \Sigma^*$ goes to zero as its length goes to infinity. So for any $\rho \in (0, 1]$, we can satisfy the randomness requirement by generating sufficiently long strings. We can efficiently detect when the EDHMM is not ergodic using

[1] The definition of the CI problem given in [5] requires that I be described by a finite automaton. It is straightforward to build a nondeterministic finite automaton that accepts precisely those strings assigned nonzero probability by the EDHMM, but we will not describe the construction here since it is not needed for the technique used in this paper.

standard graph algorithms, but this is unlikely to be necessary in practice for applications as lighting control.

3) Soft Constraints: Our procedure checks whether this requirement is satisfied using probabilistic model checking. This requires encoding the sets $\mathcal{A}_i$ as PCTL formulas, and the EDHMM as a Markov chain (explained in Sections III-D and III-E respectively). Once this has been done, the model checker computes the probability that a string generated by the EDHMM will be in $\mathcal{A}_i$. If this probability is at least $1-\epsilon_i$, then the EDHMM satisfies the soft constraint, and if this is true for each i, it is a valid improviser. Otherwise, our procedure applies heuristics to modify the EDHMM, detailed in Section III-F. As shown in that section, the heuristics decrease the expected accumulated cost, so that after sufficiently many applications the EDHMM will satisfy the soft constraints[2].

Therefore, our generation procedure yields a valid improviser solving the CI problem we defined above. We note that our technique has some further useful properties not captured by the CI problem. In particular, we can easily disable particular transitions between hidden states by setting their probabilities to zero and normalizing remaining transition probabilities appropriately. This could be useful, for example, when controlling an IoT system with unreliable components: if a component drops off the network or becomes otherwise unusable, we can disable all transitions to states in which that component is active.

C. Learning an EDHMM from Traces

The first step in our procedure is to learn an EDHMM from the input traces. Since as explained in Section II-C we use an EDHMM with different transition matrices for each hour, every input trace $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T\}$ is augmented with a corresponding stream of labels $\{\tau_1, \dots, \tau_T\}$ indicating the hour of the day each observation was recorded. Note that the observations need not be scalar costs, but could be vectors: for example, in our lighting experiments each observation was a K-tuple $\mathbf{y}_i = [y_{i,1}, \dots, y_{i,K}]^T$ containing instantaneous power readings from each of K different appliances.

Given this training data, we perform EDHMM parameter estimation as described in Section II-C. This yields a parameter set $\lambda = (\{A_h\}, \{C_h\}, B, \pi)$ where the matrices $\{A_h\}$ and $\{C_h\}$ give state transition and duration probabilities respectively for each hour $h \in \{1, \dots, 24\}$. The distribution of observations for each state is given by B, and π is the prior on the state space. In this work we use categorical distributions for B and $\{C_h\}$, although in other applications it may be appropriate to use parametric distributions.

Note that the parameter estimation process based on the EM algorithm is an iterative method; thus obtaining a reasonable parameter set depends on model convergence, which in turn requires sufficient training data. In the case of an EDHMM with hourly transition matrices, if few events happen at certain hours it may not be possible to estimate some of the state transition and duration probabilities for those hours. Many application-specific heuristics exist for handling such scenarios, as outlined in [10]. The particular technique we used in our experiments is detailed in Section IV-A.

D. Encoding Soft Constraints as PCTL Formulas

As mentioned earlier, we consider soft constraints which put upper bounds on the cost observed at a particular time or accumulated over a time period. We illustrate how to encode upper bounds on the hourly cost — other time periods are handled analogously.

Recall that our traces take the form $\{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T\}$ where each $\mathbf{y}_i$ is an observation, generally a vector $[y_{i,1}, \dots, y_{i,K}]^\mathrm{T}$ of costs. Define $Y_i \triangleq \sum_{k=1}^{K} y_{i,k}$, the total cost at time step i. Considering that the data is sampled at the rate of N_s samples per hour, the total hourly cost accumulated up to time step t is

$$\Delta = \sum_{N_s(\lceil t/N_s \rceil - 1)+1 \le i \le t} Y_i .$$

In the next section, we show how a simple monitor added to the encoding of the EDHMM can maintain the value Δ.

In order to be able to impose a different upper bound Δ^h_{max} on Δ for each hour h of the day, we need to compute the current hour of the day as a function of the time step:

$$h(t) = mod(\lceil t/N_s \rceil - 1, 24) + 1 \,,$$

which holds if $t = 1$ corresponds to the time step of the first sample collected within hour 1. Then we can write the soft constraint for hour h as the following PCTL formula:

$$P_{\ge 1-\epsilon_h} \, \mathcal{G} \left[(h(t) = h) \Rightarrow (\Delta \le \Delta^h_{max}) \right] . \tag{3}$$

This simply asserts that with probability at least $1-\epsilon_h$, at every time step during hour h the corresponding upper bound on Δ holds. In practice we can omit the quantifier $P_{\ge 1-\epsilon_h}$ and ask the probabilistic model checker to compute the probability that the rest of the formula holds, instead of having to specify a particular ϵ_h ahead of time.

E. Encoding the EDHMM as a Markov Chain

In this section, we discuss how the EDHMM can be represented as a Markov chain, so that the soft constraints can be verified using probabilistic model checking.

Ignoring the soft constraints for now, the interpretation of the EDHMM as a Markov chain follows the outline in Section II-C: we expand the state space with a new state variable $d \in \{1, \dots, D\}$ which keeps track of the remaining duration in the current hidden state $x \in \{1, \dots, N\}$. When $d > 1$, we stay in x for another time step, decrementing d.

[2] Obviously, some soft constraints cannot be satisfied, for example one requiring that the cost at the first time step be less than the smallest possible cost of any state. See Section III-F for a precise statement.

Only when $d = 1$ do we transition to a new hidden state, picking the new value of d from the corresponding duration distribution.

Since we use an extension of the EDHMM where state transition and duration probabilities depend on the current hour, we need to expand the state space further to keep track of time. The state variable $t \in \{0, \ldots, T\}$ indicates the current time step, with $t = 0$ being an initialization step in which the state is sampled from a state prior π. Note that the domain of t need not grow unboundedly with T: in our example where we use different transition probabilities for each of the 24 hours, we only need to track the time within a single day.

Finally, in order to detect when the soft constraints are violated, we need to monitor the total hourly cost Δ defined in the previous section. We add the state variable $\Delta \in \{0, \ldots, \Delta_{max} + 1\}$, where Δ_{max} is the largest of the hourly upper bounds Δ^h_{max} imposed by the soft constraints. This range of values for Δ is clearly sufficient to detect when the total cost exceeds any of these bounds. Maintaining the correct value of Δ is simple: at each time step we increase it by a cost sampled from the appropriate emission distribution, except when a new hour is starting, in which case we first reset it to zero.

Putting this all together, we obtain a Markov chain whose states are 4-tuples (x, d, t, Δ) with the state variables as described above. The initial state is $(0, 1, 0, 0)$. Given the current state, the next state (x', d', t', Δ') is determined as follows:

`EDHMM:`

$$\begin{aligned}
(t = 0) \rightarrow \quad & x' \sim \pi_x \wedge \\
& d' \sim C_{h(t)}(x') \wedge \\
& t' = t + 1 \\
(t > 0) \wedge (d > 1) \rightarrow \quad & x' = x \wedge \\
& d' = d - 1 \wedge \\
& t' = t + 1 \\
(t > 0) \wedge (d = 1) \rightarrow \quad & x' \sim A_{h(t)}(x) \wedge \\
& d' \sim C_{h(t)}(x') \wedge \\
& t' = t + 1
\end{aligned}$$

`Cost Monitor:`

$$\begin{aligned}
(t = 0) \rightarrow \quad & \Delta' = 0 \\
(t > 0) \wedge (h(t') = h(t)) \rightarrow \quad & \Delta' = \Delta + \sum_{i=1}^{K} p_i, \ \mathbf{p} \sim B(x) \\
(t > 0) \wedge (h(t') \neq h(t)) \rightarrow \quad & \Delta' = \sum_{i=1}^{K} p_i, \ \mathbf{p} \sim B(x)\,,
\end{aligned}$$

where $h(t) = mod(\lceil t/N_s \rceil - 1, 24) + 1$.

F. Scenario-Based Model Calibration

The procedure described so far provides a way to obtain a generative model that captures the probabilistic nature of events and their duration characteristics in a physical system, and to verify that the model satisfies desired soft constraints. However, the model may not satisfy these constraints with sufficiently high probability, particularly if the constraints are not always satisfied by the training data. In terms of control improvisation, the error probability of our improviser for some soft constraint i is greater than the desired ϵ_i. We now describe two general heuristics for calibrating the EDHMM to decrease the error probability while preserving the faithfulness of the improviser to the original data. In particular, these heuristics do not introduce new behaviors: any trace that can be generated by the calibrated improviser could already be generated before calibration. Since the soft constraints we consider place upper bounds on the observed costs, both heuristics seek to decrease the costs of some behavior of the improviser.

1) Duration Calibration: The duration distributions of the trained EDHMM, $\{C_h\}$, assume a maximum state duration D that is enforced during the training process. One simple way to decrease cost is to further restrict the duration distributions by truncating them beyond some threshold for some or all states. An effective strategy in practice is to eliminate outliers in the duration distributions of states with high expected cost.

This heuristic has the advantage of leaving the transition probabilities of the model completely unchanged, and so is a relatively minor modification. On the other hand, it cannot reduce the duration of a state below 1 time step. So although it can eliminate some high-cost behaviors from the model, it is not guaranteed to eventually yield an improviser satisfying the soft constraints.

2) Transition Calibration: A different approach is to modify the state transition probabilities, making the model less likely to transition to a high cost state during certain hours of the day. Specifically, we can limit the probability of transitioning from any state i to a particular state x_r during hour h_r to be at most some value p^i_r. We shift the removed probability mass to the transition leading to the state x_{min} with least expected cost, which we assume is strictly less than that of x_r. Writing the original transition probability matrix A_{h_r} as (a_{ij}), we replace it in the EDHMM with a new matrix $\tilde{A}_{h_r} = (\tilde{a}_{ij})$ defined by

$$\tilde{a}_{ij} = \begin{cases} \min(p^i_r, a_{ij}) & \text{if } j = x_r \\ a_{ij} + (a_{ix_r} - \min(p^i_r, a_{ij})) & \text{if } j = x_{min} \\ a_{ij} & \text{otherwise.} \end{cases}$$

Note that the second case ensures that the transition probabilities from any state $i \in \mathcal{X}$ are properly normalized. Provided that the limits p^i_r are chosen such that $\tilde{a}_{ix_r} < a_{ix_r}$ for some $i \in \mathcal{X}$, the heuristic will decrease the expected cost of a behavior generated by the improviser.

Applying the heuristic iteratively for every choice of $x_r \neq x_{min}$ and hour $h_r \in \{1, \ldots, 24\}$ will eventually result in an improviser that remains at the x_{min} state for all time steps (assuming x_{min} is the starting state). Thus for any soft constraints which are true for behaviors that only stay at x_{min}, our procedure will eventually terminate and yield a valid improviser. This over-simplified improviser is unlikely to model the original data well, but it is only attained as the limit of this heuristic: in practice, judicious choices of the state x_r and limits p_r^i can improve the error probability significantly in a few iterations without drastically changing the model.

IV. Experimental Results and Analysis

A. Experimental Setup

To demonstrate the control improvisation approach we have described in Sec. III, we use the UK Domestic Appliance-Level Electricity (UK-DALE) dataset [8], which contains disaggregated time series data representing instantaneous power consumptions of residential appliances from 5 homes over a period of 3 years.

We consider a lighting improvisation scenario over the three most-used lighting appliances in a single residence, each from a separate room of the house. The data is presented as a vector-valued power consumption sequence $\mathbf{y}$ with a corresponding sequence of time stamps $\boldsymbol{\tau}$. The input stream $\mathbf{y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ consists of 3-tuples

$$\mathbf{y}_i = \begin{bmatrix} y_{i,1} \\ y_{i,2} \\ y_{i,3} \end{bmatrix}, \; i = 1, \ldots, T \,,$$

where the values $y_{i,1}$, $y_{i,2}$, and $y_{i,3}$ are instantaneous power readings with time stamp τ_i from the main kitchen light, a dimmable living room light, and the bedroom light respectively. The power readings were sampled with a period of 1 minute and are measured in watts.

In our experiments, we synthesized three improvisers from this data: one using an unmodified EDHMM, and two that were calibrated using the different kinds of heuristics described in Section III-F to enforce soft constraints on hourly power consumption. Below, we describe the specific choices that were made when implementing each of the three main steps of our procedure.

1) Data-Driven Modeling: We assume there are three sources of hidden events, corresponding to each of the three appliances being turned on or off. This yields a hidden state space $\mathcal{X}$ with 8 states, one for each combination of active appliances. Based on inspection of the dataset, we chose the maximum state duration to be 720 time steps (12 hours, sufficient to allow long periods when all appliances are off). Since we used disaggregated data, our observations are 3-tuples of power consumptions (quantized to integer values as part of the dataset), which we assume fall in the alphabet $\Sigma = \Sigma_1 \times \Sigma_2 \times \Sigma_3$ where $\Sigma_1 = \{0, 1, \ldots, 350\}$, $\Sigma_2 = \{0, 1, \ldots, 20\}$, and $\Sigma_3 = \{0, 1, \ldots, 30\}$ (the maximum consumptions for each appliance were again obtained by inspecting the dataset). Having fixed these parameters (summarized in Table I), an EDHMM was trained from a 100-day subset of the data from one residence. Several portions of this training data (for one appliance) are shown at the top of Figure 9.

Note that for the specific case of lighting improvisation, since the power emission distributions of each appliance are independent, $B \triangleq p(\mathbf{y_t}|x_t)$, the learned emission probability matrix over vector-valued observations, can be written as

$$B = p(\mathbf{y_t}|x_t) = \prod_{k=1}^{K} p(y_{t,k}|x_t)\,.$$

It should also be noted that following the training process, some of the state transition probabilities $\{A_h\}$ may remain unlearned, i.e., we may have

$$\sum_{j=1}^{N} a_{i,j}^h = 0$$

for some state $i \in \{1, ..., N\}$. This can occur, for example, when no state transitions from state i happen during the hour h in any of the input traces. Since it is key to capture the observed appliance behavior, we treat these incomplete distributions that are unobserved in the training data as behaviors that should also be absent from the set of improvised behaviors. Consequently, we use a completion strategy that forces transitions to the state x_{min} with the least expected cost (i.e. the state with all appliances off) in this scenario:

$$\forall a_{i,j}^h\,,\ \text{where} \sum_{j=1}^{N} a_{i,j}^h = 0,$$
$$i, j \in \{1, ..., N\},\ h \in \{1, ..., 24\},$$
$$\tilde{a}_{i,j}^h = \begin{cases} 1 & \text{if } j = x_{min} \\ 0 & \text{otherwise} \end{cases}$$

where $\tilde{a}_{i,j}^h$ is the adjusted state transition probability of switching from state i to j in hour h. Note that in this case study, such incomplete parameter estimates arose only for early morning hours in which few state transitions were recorded (typically hours $h \in \{1, \ldots, 5\}$). Having completed the transition probability matrices in this way, we obtain a fully specified EDHMM.

2) Probabilistic Model Checking: We experimented with soft constraints upper bounding the total power consumed during each hour. Figure 4 depicts the hourly energy consumptions of each appliance, as well as the aggregated consumption, averaged across each day in the training data. The maximum hourly consumptions occurring in the training data are not ideal bounds to use as soft constraints, since they tend to be trivially satisfied by the improviser. Instead,

Parameter ID	Value
Data Source	UK DALE Dataset
House ID	house_1
Appliance IDs	kitchen_lights livingroom_s_lamp bedroom_ds_lamp
Training Duration	100 days
Training Start Date	30 Jul 2013 19:07:56 GMT
Sampling Period (T_s)	60 s
Training Sequence Length (T)	144000
Maximum State Duration (D)	720
Appliance 1 Costs (Σ_1)	$\{0, 1, \ldots, 350\}$
Appliance 2 Costs (Σ_2)	$\{0, 1, \ldots, 20\}$
Appliance 3 Costs (Σ_3)	$\{0, 1, \ldots, 30\}$
State Labels	OFF: All appliances off K: Kitchen on L: Living room on B: Bedroom on KL: Kitchen and living room on KB: Kitchen and bedroom on LB: Living room and bedroom on KLB: All appliances on

Table I: Parameters of the training dataset for EDHMM learning

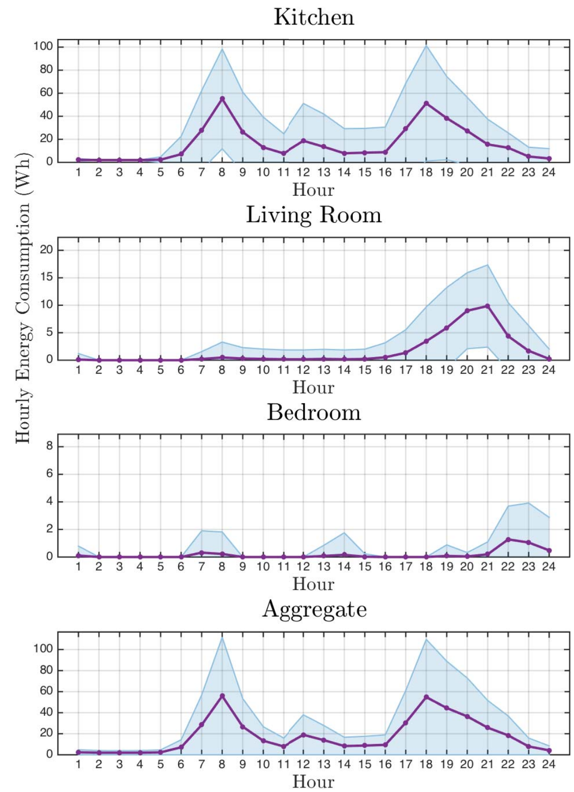

Figure 4: Hourly usage patterns of main lighting appliances. Solid curve represents average consumption and shaded area represents one standard deviation above mean

for each hour h we imposed a tighter bound Δ^h_{max} on the aggregate power consumption during that hour, where Δ^h_{max} was one standard deviation above the mean consumption in hour h in the training data. Note that 89.2% of the training samples were within this bound. The values Δ^h_{max} are plotted as the shaded curve at the bottom of Figure 4.

To compute the probability of satisfying these constraints, we used the PRISM model checker [7]. As detailed in Section III-E, the EDHMM and a monitor tracking hourly power consumption can be written as a discrete-time Markov chain. This description translates more or less directly into the PRISM modeling language. Having done this, the soft constraints can be put directly into PRISM using the PCTL formulation explained in Section III-D to obtain the hourly satisfaction probabilities $1 - \epsilon_i,\ i = 1, \ldots, 24$.

3) Scenario-Based Model Calibration: As mentioned above, we tested three types of improvisers:

- **Scenario I: Uncalibrated Improviser.** This improviser uses the learned EDHMM with no model calibration.
- **Scenario II: Duration-Calibrated Improviser.** This improviser uses the duration calibration heuristic described in Section III-F. From the aggregate power profile given in Figure 4, we identified peak power consumption as occurring during hours 7, 8, 9, 17, 18, 19, 20, and 21. For these hours, the probabilities of event durations greater than 60 minutes were set to zero and the distributions re-normalized. Figure 5 shows a sample set of original and calibrated event duration distributions for the 19th hour of the day.
- **Scenario III: Transition-Calibrated Improviser.** This improviser extends the previous one by also applying the transition calibration heuristic described in Section III-F. The set of hours for which transition probabilities were calibrated includes the peak hours considered in the previous section, with the addition of hours 4 and 5, for which very few events were recorded in the training data. As Figure 4 indicates, the significant sources of power consumption are the kitchen and the living room lighting appliances. Therefore, we choose x_r to include states K, L, KL, and KLB (see Table I for label descriptions).
Figure 6 depicts some hourly transition probability matrices before and after calibration. Each circle indicates a nonzero transition probability from state x_t to x_{t+1}, where its area is proportional to the probability. The blue circles show the original learned probabilities, and the green circles show the probabilities decreased by calibration. For clarity, we do not show the corresponding increases in the probabilities of transitioning to the OFF state.

B. Experimental Results

Our focus in this section is to evaluate the performance of synthesized improvisers using probabilistic model checking and to compare them based on their fidelity to soft constraints. It is also of interest to study the power profile

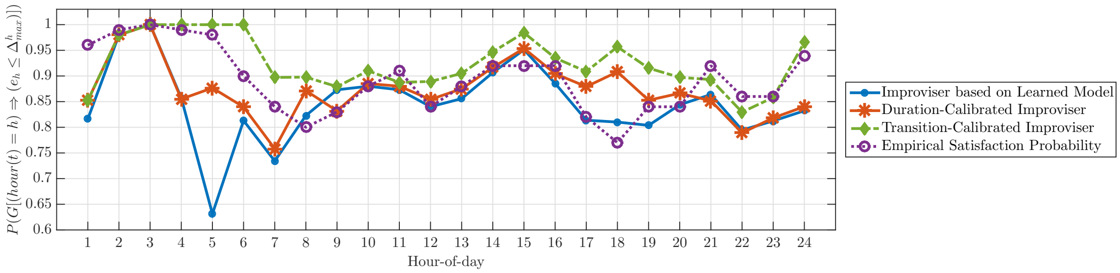

Figure 7: Model checking results on the satisfaction probabilities of hourly soft constraints

characteristics of improvised traces to ensure scenario-based calibrations do not impact the similarity of improvisations to recorded data based on human actuation.

Figure 7 summarizes model checking results for the original EDHMM and for the two calibrated models with constrained power consumption properties. We additionally provide the empirical probability of soft constraints being satisfied by training data, mainly for visual comparison. Model checking results suggest that the improviser based on the learned EDHMM behaves comparably to the empirical satisfaction probabilities, however, since the soft constraints are not explicitly enforced by the EDHMM, some hourly probabilities significantly deviate from empirical values. When we investigate model checking results for the two calibrated improvisers, which aim to improve the probability of satisfying soft constraints, we observe that the transition-calibrated improviser yields highest satisfaction probabilities on the soft constraints for all hours of the day. The duration-constrained improviser performs better than the learned model, for all hours except for hours 9, 21 and 22. As explained in Section III-F, the duration heuristic does not guarantee an improvement in the probability of satisfying the soft constraints. This can be explained in this particular case by the phenomenon that at these particular hours, the state transition matrix tends to make transitions to high-consumption states more probable, and skewing the duration distribution towards zero causes more state transitions to be made during peak hours.

Figure 8 compares the aggregate hourly power consumption profiles obtained from the training data, with ones obtained from 100 20-day long improvisations generated by a particular lighting improviser. For all three improviser profiles, the hourly mean power trend matches that of the original data. Moreover, for calibrated improvisers, the one standard deviation curve above mean mostly remains within the same bound for the original data. Even though the duration-calibrated improviser has eliminated most of the

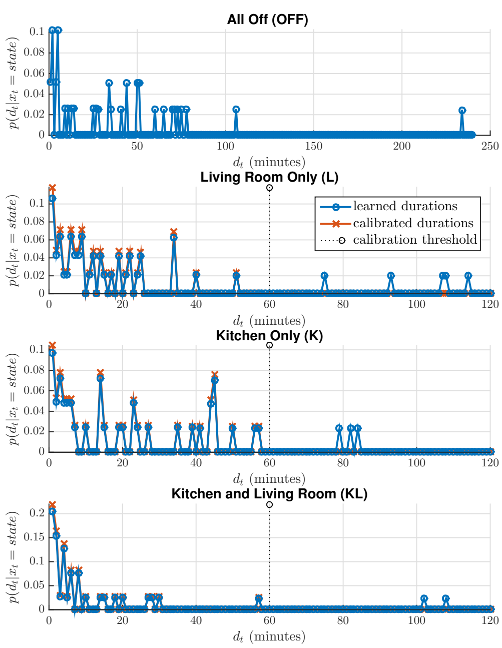

Figure 5: Sample learned and calibrated duration distributions for h=19

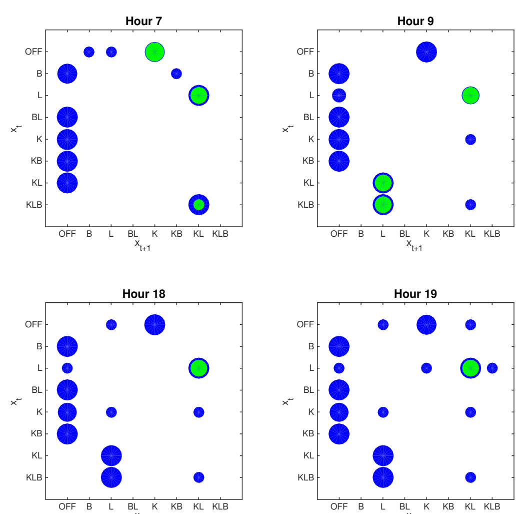

Figure 6: Learned vs. calibrated state transition probabilities for selected hours. (Blue: Learned A_h, Green: Probabilities adjusted by Scenario III, See Table I for label descriptions)

highly variable power consumption trend exhibited by the uncalibrated improviser, it still demonstrates high variability in power for the hour range $h = 9, \ldots, 12$ compared to the training data. This behavior is successfully mitigated by the transition-calibrated improviser, which is shown to satisfy the one sigma power constraint more strictly than the duration-calibrated improviser as expected.

Finally, in Figure 9, we show several day-long traces from the three improvisers together with time-aligned excerpts from the training data. Note that the uncalibrated improvisations are visually quite similar to the training data, illustrating the quality of the EDHMM as a model. The calibrated improvisations are also qualitatively similar to the training data, but somewhat sparser as we would expect from enforcing constraints on power consumption. This demonstrates how our model calibration techniques are effective at enforcing soft constraints without drastically changing system behavior.

Overall, experimental results suggest that, given a suitable learning model, it is possible to synthesize a control improviser which produces randomized control sequences that are faithful to observed system behavior. More importantly, scenario-based model calibration methods can be applied to systematically constrain the nature of randomness, which is quantifiable via probabilistic model checking. Our experiments have shown significant improvements on the satisfaction probabilities of soft constraints after applying heuristic calibrations, while preserving desired qualitative characteristics in improvised control sequences.

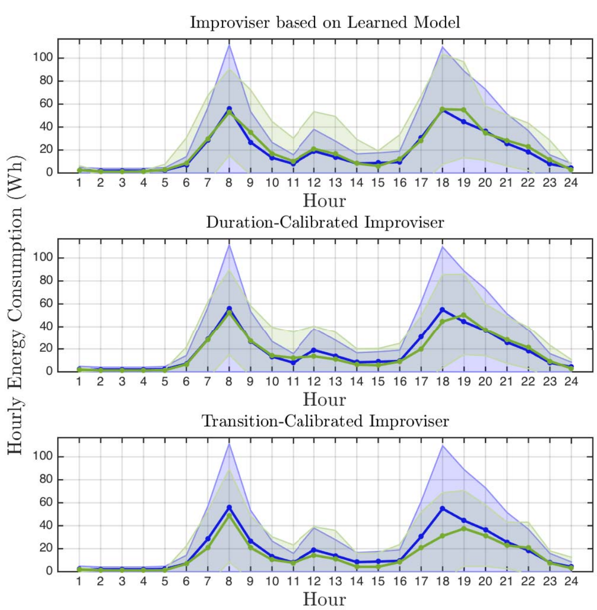

Figure 8: Comparison of aggregate hourly energy profiles (Blue: Training data, Green: Improvisations. Solid curves represent mean energy, shaded region represents one standard deviation from mean)

V. Related Work

Control improvisation is an automata-theoretic problem that was formally defined and analyzed in [5]. CI was applied to machine improvisation of music in [4], where a symbolic reference melody was used to synthesize an automaton that was composed with a specification automaton (capturing user-specified musical properties) to produce a control improviser. In this work, we consider a case study in the field of home automation, which enables us to learn a more general Bayesian model. We represent training data by modeling temporal progression and the stochastic characteristics of underlying events given noisy sensor data. Moreover, as an extension of our previous work, we learn specifications from user-generated data directly, and perform scenario-based calibrations on the learned model to enforce formal statistical properties.

Appliance modeling in residential settings has several proposed benefits, including reduced power consumption, automated actuation of smart appliances subject to energy pricing, microgrid load balancing, and home security [12]. Additionally, personalized advisory tools have gained popularity to provide adaptive demand-response prediction [13], [14]. Bayesian modeling techniques for home appliance load modeling has been an emerging topic of interest [15], and EDHMM-based models have previously been proposed for load disaggregation [16]. Markov modeling of uncertainties in demand and energy pricing has been studied in [17], which presents a reinforcement learning based approach to optimal load scheduling.

The related subjects of data-driven occupancy prediction [18] and user behavior modeling for energy demand predictions have also been studied in recent years. In [19], a stochastic model to predict time-dependent user activity was presented, while in [20], a data-driven approach was adapted for learning residential power profiles based on user-specific factors. Integration of suitable occupancy and user prediction techniques with ours is a clear direction for future work.

VI. Conclusion

In this paper, we address the problem of randomized control for IoT systems, with a particular focus on systems whose components can be controlled either by humans or automatically. From streams of time-stamped system events, we learn models that are assumed to vary as a function of an underlying state space governed by events with durations. We leverage the recently-proposed technique of control improvisation [5], [21] to generate randomized control sequences, which are *similar* to an observed set of behaviors, and moreover, always satisfy some desired *hard* constraints and *mostly* satisfy *soft* constraints, while exhibiting *variability*. We presented an implementation of the end-to-end control improvisation workflow using the PRISM tool to enforce soft constraints on the improviser. We evaluated our technique in the domain of home appliance control

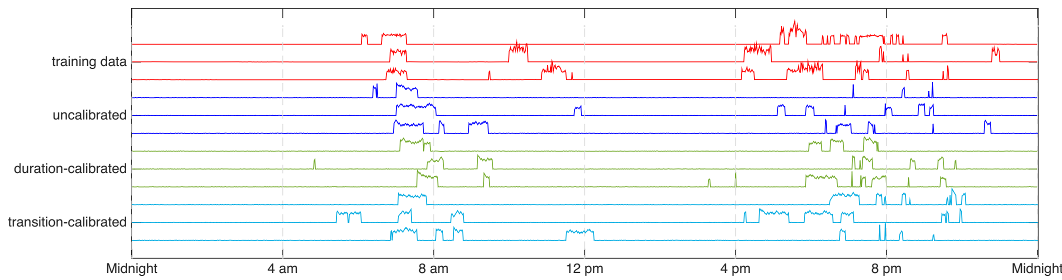

Figure 9: Appliance 1 activation patterns over day-long intervals

by synthesizing improvisers to control a group of lighting appliances based on learned usage patterns and subject to probabilistic constraints on power consumption. The results of our experiments showed that our methods can effectively enforce soft constraints while largely maintaining qualitative and quantitative properties of the original system's behavior.

For future work, we plan to investigate new applications of this framework in the IoT space. We also plan to investigate techniques to improve the efficiency of our scheme, as well as its implementation on real hardware.

ACKNOWLEDGMENT

This work is supported in part by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1106400 and by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

REFERENCES

[1] M. Strelec, K. Macek, and A. Abate, "Modeling and simulation of a microgrid as a stochastic hybrid system," in *Innovative Smart Grid Technologies (ISGT Europe), 2012 3rd IEEE PES International Conference and Exhibition on*. IEEE, 2012, pp. 1–9.

[2] C. Gamarra and J. M. Guerrero, "Computational optimization techniques applied to microgrids planning: a review," *Renewable and Sustainable Energy Reviews*, vol. 48, pp. 413–424, 2015.

[3] G. Assayag and S. Dubnov, "Using factor oracles for machine improvisation," *Soft Computing*, vol. 8, no. 9, pp. 604–610, 2004.

[4] A. Donzé, R. Valle, I. Akkaya, S. Libkind, S. A. Seshia, and D. Wessel, "Machine improvisation with formal specifications," in *Proceedings of the 40th International Computer Music Conference (ICMC)*, 2014.

[5] D. J. Fremont, A. Donzé, S. A. Seshia, and D. Wessel, "Control Improvisation," in *35th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2015)*, 2015, pp. 463–474.

[6] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[7] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, 2011.

[8] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," vol. 2, no. 150007, 2015.

[9] E. A. Lee and A. Sangiovanni-Vincentelli, "A framework for comparing models of computation," *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, vol. 17, no. 12, pp. 1217–1229, 1998.

[10] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[11] S.-Z. Yu, "Hidden semi-markov models," *Artificial Intelligence*, vol. 174, no. 2, pp. 215–243, 2010.

[12] L. G. Swan and V. I. Ugursal, "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques," *Renewable and sustainable energy reviews*, vol. 13, no. 8, pp. 1819–1835, 2009.

[13] DR-Advisor: Data-Driven Demand Response Recommender System. Accessed: 2015-10-15. [Online]. Available: http://mlab.seas.upenn.edu/dr-advisor/

[14] Y. Li, B. L. Ng, M. Trayer, and L. Liu, "Automated residential demand response: Algorithmic implications of pricing models," *Smart Grid, IEEE Transactions on*, vol. 3, no. 4, pp. 1712–1721, 2012.

[15] Z. Kang, M. Jin, and C. Spanos, "Modeling of end-use energy profile: An appliance-data-driven stochastic approach," in *Industrial Electronics Society, IECON 2014 - 40th Annual Conference of the IEEE*, Oct 2014, pp. 5382–5388.

[16] Z. Guo, Z. J. Wang, and A. Kashani, "Home appliance load modeling from aggregated smart meter data," *Power Systems, IEEE Transactions on*, vol. 30, no. 1, pp. 254–262, 2015.

[17] D. O. Neill, M. Levorato, A. Goldsmith, and U. Mitra, "Residential demand response using reinforcement learning," in *Smart Grid Communications, 2010 First IEEE International Conference on*. IEEE, 2010, pp. 409–414.

[18] D. Aerts, J. Minnen, I. Glorieux, I. Wouters, and F. Descamps, "A method for the identification and modelling of realistic domestic occupancy sequences for building energy demand simulations and peer comparison," *Building and environment*, vol. 75, pp. 67–78, 2014.

[19] U. Wilke, F. Haldi, J.-L. Scartezzini, and D. Robinson, "A bottom-up stochastic model to predict building occupants' time-dependent activities," *Building and Environment*, vol. 60, pp. 254–264, 2013.

[20] B. Aksanli, A. S. Akyurek, and T. S. Rosing, "User behavior modeling for estimating residential energy consumption," in *Proc. EAI International Conference on Smart Grids for Smart Cities*. Springer, 2015.

[21] D. J. Fremont, A. Donzé, S. A. Seshia, and D. Wessel, "Control improvisation," *arXiv preprint arXiv:1411.0698*, 2014.