Data-Driven Model Predictive Control with Regression Trees—An Application to Building Energy Management

ACHIN JAIN, University of Pennsylvania
FRANCESCO SMARRA, Università degli Studi dell'Aquila
MADHUR BEHL, University of Virginia
RAHUL MANGHARAM, University of Pennsylvania

Model Predictive Control (MPC) plays an important role in optimizing operations of complex cyber-physical systems because of its ability to forecast system's behavior and act under system level constraints. However, MPC requires reasonably accurate underlying models of the system. In many applications, such as building control for energy management, Demand Response, or peak power reduction, obtaining a high-fidelity physics-based model is cost and time prohibitive, thus limiting the widespread adoption of MPC. To this end, we propose a data-driven control algorithm for MPC that relies only on the historical data. We use multi-output regression trees to represent the system's dynamics over multiple future time steps and formulate a finite receding horizon control problem that can be solved in real-time in closed-loop with the physical plant. We apply this algorithm to peak power reduction in buildings to optimally trade-off peak power reduction against thermal comfort without having to learn white/grey box models of the systems dynamics.

CCS Concepts: • **Computing methodologies → Classification and regression trees**;

Additional Key Words and Phrases: Machine learning, predictive control, cyber-physical systems, demand response, peak power reduction

ACM Reference format:
Achin Jain, Francesco Smarra, Madhur Behl, Rahul Mangharam. 2018. Data-Driven Model Predictive Control with Regression Trees—An Application to Building Energy Management. *ACM Trans. Cyber-Phys. Syst.* 2, 1, Article 4 (January 2018), 21 pages.
https://doi.org/10.1145/3127023

4

1 INTRODUCTION

The year 2016 was the hottest on record since the beginning of weather recording in 1880 [12]. Heat waves in summer and polar vortexes in winter are growing longer and pose increasing challenges to an already over-stressed electric grid. Furthermore, with the increasing penetration of

This work was partially supported by the Italian Government under Cipe resolution n.135 (Dec. 21, 2012), project *INnovating City Planning through Information and Communication Technologies* (INCIPICT).
Authors' addresses: A. Jain and R. Mangharam, Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104, USA; emails: {ajain, rahum}@seas.upenn.edu; F. Smarra, Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Via Vetoio, Coppito, 67100 L'Aquila, Italy and Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104, USA; emails: francesco.smarra@univaq.it, fsmarra@seas.upenn.edu; M. Behl, Room 423, Rice Hall, 85 Engineers Way, University of Virginia, Charlottesville, VA 22903, USA; email: mb2kg@virginia.edu.

https://doi.org/10.1145/3127023

renewable generation, the electricity grid is also experiencing a shift from predictable and dispatchable electricity generation to variable and non-dispatchable generation. This adds another level of uncertainty and volatility to the electricity grid. The volatility due to the mismatch between electricity generation and supply further leads to volatility in the wholesale price of electricity. For example, the polar vortex triggered extreme weather events in the U.S. in January 2014, which caused many electricity customers to experience increased costs. Parts of the U.S. northeastern electricity grid experienced an 86-fold increase in the price of electricity from \$31/MW h to \$2,680/MW h in a matter of a few minutes [22]. Such events show how unforeseen and uncontrollable circumstances can greatly affect electricity prices that impact grid operator and customers. Electricity price volatility is becoming the new norm rather than the exception.

Across the United States, utilities and grid operators are devoting increasing attention and resources to Demand Response (DR). It is considered as a reliable means of mitigating the uncertainty and volatility of renewable generation and extreme weather conditions and improving the grid's efficiency and reliability. The resource contribution from all U.S. DR programs is estimated to be nearly 72,000MW, or about 9.2% of U.S. peak demand [10], making DR the largest virtual generator in the U.S. national grid. The annual revenue to end-users from DR markets with PJM ISO alone is more than \$700 million [26]. Global DR revenue is expected to reach nearly \$40 billion from 2014 through 2023 [23].

The volatility in real-time electricity prices poses the biggest operational and financial risk for large-scale end-users of electricity, such as large commercial buildings, industrial and institutional customers (often referred to as *C/I/I* consumers). To shield themselves from the volatility and risk of high prices, such consumers must be more flexible in their electricity demand. Consequently, large *C/I/I* customers are increasingly looking to demand response programs to help manage their electricity costs. DR programs involve a voluntary response of a building to a price signal or a load curtailment request from the utility or the Curtailment Service Provider (CSP). Upon successfully meeting the required curtailment level the end-users are financially rewarded, but may also incur penalties for under-performing and not meeting a required level of load curtailment. In practice, one of the biggest challenges with end-user demand response is the following: *Upon receiving the notification for a DR event, what actions must the end-user take to achieve an adequate and a sustained DR curtailment?*

Unfortunately, most of the Demand Response today is done in an ad hoc manner. The decisions are rule-based or they depend upon a building operator's experience, which has several disadvantages:

- *Limitations of rule-based DR.* The building's operating conditions, internal thermal disturbances, and environmental conditions must all be taken into account to make appropriate DR control decisions, which is not possible with using rule-based and pre-determined DR strategies, since they do not account for the state of the building but are instead based on best practices and rules of thumb. Rule-based DR strategies have the advantage of being simple, but they do not account for the state of the building and weather conditions during a DR event.
- *Control complexity and scalability.* Upon receiving a notification for a DR event, the building's facilities manager must determine an appropriate DR strategy to achieve the required load curtailment. These control strategies can include adjusting zone temperature set-points, supply air temperature and chilled water temperature set-point, dimming or turning off lights, decreasing duct static pressure set-points and restricting the supply fan operation, and so on. In a large building, it is difficult to assess the effect of one control action on other sub-systems and on the building's overall power consumption, because the building sub-systems are tightly coupled.

These drawbacks are addressed by predictive control techniques like Model Predictive Control (MPC). MPC allows us to optimize desired performance while guaranteeing system's constraints. For example, it can be used to provide the optimal actions for an energy curtailment during a DR event while guaranteeing thermal comfort for occupants. So, MPC-based solutions fit into the DR problem. However, the quality of the MPC solutions depends on the accuracy of the predictive models describing the system. This limits the widespread of MPC for complex systems such as buildings. Thus, in context of modeling of large buildings, applying MPC poses further challenges:

- *Modeling complexity and heterogeneity.* Unlike the automobile or the aircraft industry, each building is designed and used in a different way, and therefore it must be uniquely modeled. Learning predictive models of a building's dynamics using first-principles-based approaches (e.g., with EnergyPlus [7]) is very cost and time prohibitive and requires retrofitting the building with several sensors [29]. The user expertise, time, and costs required to develop a model of a single building are very high.

1.1 Main Contribution

To address the aforementioned challenges, we use machine-learning algorithms to generate data-driven models for cyber-physical systems where we preserve predictive capability of model-based control but without the expense of first principle or grey-box model development. Although machine-learning algorithms are a popular choice for predicting systems' behavior, they do not provide models that are suitable for control. In our previous work [1, 2], we overcome this issue by using regression tree-based algorithms to provide models of system dynamics to be used in an optimal control problem setup. We proposed an algorithm for model-based control using regression trees (mbCRT) and apply it to synthesize real-time DR strategy. The mbCRT algorithm can be used to setup optimal control problems with one-step lookahead prediction, for example, to trade off thermal comfort inside a building against the amount of load curtailment. However, mbCRT cannot be used to setup a receding horizon control problem with an arbitrary length of the horizon as in MPC. This drawback limits the closed-loop control performance, since it does not take into account the very accurate long-term state prediction that regression trees can provide. In this article, we overcome this issue providing the following contributions:

- *Multi-variate regression trees:* We introduce a methodology to construct a multi-variate or multi-output predictive model using regression trees, where each output corresponds to the prediction on a future step of the horizon. More precisely, we setup a least-square problem that minimizes the prediction error over the control horizon. This is done by modifying the variable selection and splitting criteria at the nodes of the standard regression tree algorithm CART [4].
- *Predictive control:* We extend the mbCRT algorithm proposing a Data-driven Predictive Control strategy using Regression Trees (DPCRT), which implements receding horizon control using data-based models learned from the available historical data.
- *Application:* We apply this control strategy to peak power curtailment on a large office building and compare with the previous mbCRT algorithm. Our results show an improvement of 8.6% in terms of peak curtailment is obtained using DPCRT with respect to mbCRT.

In summary, the methodology proposed in this article bypasses the cost- and time-prohibitive process of building high-fidelity models (using grey and white box approaches) while still being suitable for receding horizon control. The contributions of the article are graphically shown in Figure 1.

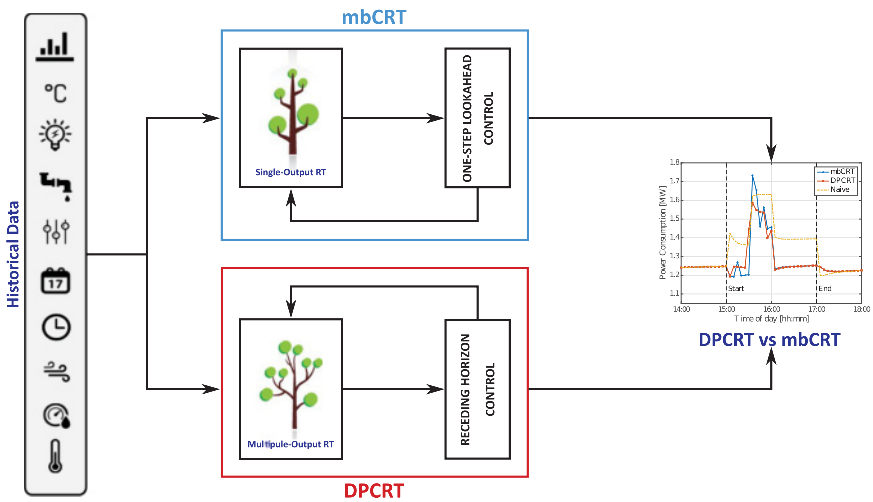


Fig. 1. Historical data are used to train models for mbCRT in Section 3 and DPCRT in Section 4. These models are then used in the optimization in a closed-loop with the physical system. Finally, the two approaches are compared in Section 5.

1.2 Related Work

In recent years, data-driven optimal control has drawn a huge interest in the research community. Several different approaches have been considered. For example, in one approach, models are trained on optimal solutions obtained from MPC. The resulting models can then be used for explicit MPC, as in Reference [3]. This approach has been applied to problems of stabilization [5] and freeway traffic systems using regression trees [25]. In applications like buildings, where obtaining a model for MPC is hard, such techniques cannot be used. Another class of methods solve the optimization directly on the trained models to do predictive control [6, 11, 16]. However, none of them addresses the problem of including data-driven state models into the optimal predictive control loop. Hence, we cannot solve an MPC-like optimal control problem. In Reference [2], we proposed MPC-like control problem considering regression tree-based algorithms for the DR problem. In Reference [21], neural networks are used instead to setup an MPC-like control for the wind turbine problem using evolutionary algorithms. Nevertheless, their approaches considered only one-step lookahead predictive model, and hence does not allow predictive control over an arbitrary horizon.

There is a vast literature more specific to the Demand Response that look at the problem of determining DR strategies [24, 31]. The majority of the approaches are either rule-based techniques for curtailment or white/grey box model-based control. These usually assume that the model of the system is well known whereas the task is much more complicated and time consuming in the case of a real building and sometimes, it can be even more complex and involved than the design of the controller itself. After several years of work on using first-principles-based models for demand response, multiple authors [29, 33] have concluded that the biggest hurdle to mass adoption of intelligent building control is the cost and effort required to capture accurate dynamical models of the buildings. There are ongoing efforts to make tuning and identifying white box models of buildings more autonomous [27]. OpenADR standard and protocol [15] describes the formats for information exchange to facilitate DR but modeling, prediction and control strategies are out of scope. Several machine-learning approaches [9, 30, 32] have been utilized before for forecasting electricity load including some that use regression trees. However, there are two significant shortcomings of the work in this area: (a) these approaches are coarse grained and are not aimed at solving demand response problems but are restricted to long-term load forecasting with applications in evaluating building retrofits savings and building energy ratings, and (b) there is no focus on control synthesis or addressing the suitability of the model to be used in control design. There exist several different approaches to balance the power consumption in buildings and avoid peaks, e.g. by load shifting and load shedding [14, 20]. However, they operate on coarse grained time scales and do not guarantee any thermal comfort.

The DPCRT algorithm is first of its kind that does finite receding horizon control with regression trees. It is computationally efficient, because the optimization problem is convex and the number of constraints scales linearly with the number of control variables. A preliminary version of this article can be found in the conference article [17].

1.3 Paper Organization

This article is organized as follows. We describe our problem formulation in Section 2. Section 3 provides an overview of the data-driven control strategy (mbCRT) proposed in Reference [2] that will be useful to introduce our new methodology. In Section 4, we present our main contribution—predictive control using multi-variate regression trees. In particular, we extend the standard training algorithm for regression trees in Section 4.1, so we can use it to build predictive models over a horizon of arbitrary length. In Section 4.2, we use the DPCRT algorithm to formally build system models and setup a receding horizon optimal control problem that can be used for the closed-loop control. Finally, in Section 5, we apply DPCRT to a peak power reduction problem and compare it with mbCRT.

2 PROBLEM DEFINITION

Our goal is to learn data-driven models for cyber-physical systems starting from the available historical data. We would like to build models that relate the value of the response variables $\mathrm{y}_1, \ldots, \mathrm{y}_p$ with the values of the predictor variables or features $\mathrm{x}_1, \ldots, \mathrm{x}_s$, and that can be used to setup an optimization predictive control problem. For example, in the case of buildings, the response variables can be represented by the power consumption or the room temperatures, while the features can be represented by weather data, set-points information, and building schedules.

When the dataset has several features, as in the case of large buildings, which interact in complicated nonlinear ways, assembling a single global model, such as linear or polynomial regression, can be difficult and lead to poor response predictions. An approach to non-linear regression is to partition the data space into smaller regions R_i, where the interactions are more manageable. A regression tree is an example of an algorithm that belongs to the class of recursive partitioning algorithms. The seminal algorithm for learning regression trees is CART as described in Reference [4]. Regression trees-based approaches are our choice to construct data-driven models for cyber-physical systems. The primary reason for this modeling choice is that regression trees are highly interpretable by design. Interpretability is a fundamental desirable quality in any predictive model. Complex predictive models, such as neural networks, support vector regression, and so on, go through a long calculation routine and involve too many factors. It is not easy for a human engineer to judge if the operation/decision is correct or not or how it was generated in the first place. For example, building operators are used to operating a system with fixed logic and rules. They tend to prefer models that are more transparent, where it is clear exactly which factors were used to make a particular prediction. At each node of a regression tree, a simple *if this, then that* human readable plain text rule is applied to generate a prediction at the leafs, which anyone can easily understand and interpret. Making machine-learning algorithms more interpretable is an active area of research, one that is essential for incorporating human centric models in cyber-physical systems.

We want to use such regression trees-based models to setup optimal predictive control strategies as in the classical MPC formulation. The main challenge when dealing with machine-learning algorithms is that they do not provide mathematical models that are directly usable for control. To overcome this issue, we propose a solution that is based on the *separation of variables.* In particular, given a dataset $(\mathcal{X}, \mathcal{Y})$, where $\mathcal{X} = \{\mathrm{x}_1, \ldots, \mathrm{x}_s\}$ is the set of predictor variables (or features) and $\mathcal{Y} = \{\mathrm{y}_1, \ldots, \mathrm{y}_p\}$ is of response variables, we want to use regression trees to learn a model f_i to predict the response y_i as

$$\mathrm{y}_i = f_i(\mathrm{x}_1, \ldots, \mathrm{x}_s). \tag{1}$$

Given a forecast of the predictor variables $\hat{\mathrm{x}}_i$, we can predict the system response using Equation (1). However, when some of the features are control variables, such a model is not suitable for control. We denote the set of system inputs by $\mathcal{X}_c$. These are the variables we can manipulate and form a subset of the features set; i.e., $\mathcal{X}_c = \{\mathrm{u}_1, \ldots, \mathrm{u}_m\} \subset \mathcal{X}$. Let $u(t) = [\mathrm{u}_1(t), \ldots, \mathrm{u}_m(t)]$ be the vector of inputs applied to the system at time t to be optimized, $x(t) = [\mathrm{x}_1(t), \ldots, \mathrm{x}_{s-m}(t)]$ be the vector of system variables measured at time t, and $y(t) = [\mathrm{y}_1(t), \ldots, \mathrm{y}_p(t)]$ be the vector of system response variables at time t. We want to setup a receding horizon control problem to find the actions that optimize a desired cost function. The model Equation (1) is not suitable to solve the following optimal predictive control problem:

$$\begin{aligned} \underset{u_{t+k} \in \mathcal{X}_c}{\text{minimize}} \quad & \sum_{k=0}^{N} J(y_{t+k}, u_{t+k}), \\ \text{subject to} \quad & y_{t+k} = f(x_{t+k}, u_{t+k}), \\ & u_{t+k} \in \bar{\mathcal{U}}, \\ & y_{t+k} \in \bar{\mathcal{Y}}, \\ & x_t = x(t), \\ & k = 0, \ldots, N, \end{aligned} \tag{2}$$

since the inputs u are the decision variables in the optimization problem, and they are not known a priori to be used as features in Equation (1).

We use *separation of variables* to partition the set of features into two disjoint sets of inputs (or control variables) and disturbances, i.e., the sets of variables we can and cannot manipulate, respectively. More precisely, $\mathcal{X} = \mathcal{X}_c \uplus \mathcal{X}_d$, where $\uplus$ represents the disjoint union. The regression tree is trained only using the disturbance variables in $\mathcal{X}_d$, so that we can associate to each leaf a model that depends only on the input data. In this way, we can use the forecast of disturbances to predict the system behavior using Equation (1). This procedure provides a modeling framework that allows to solve optimization problem Equation (2). In Sections 3 and 4, we show how this is achieved. In particular, in the next section, we review the mbCRT algorithm, which uses separation of variables to create a model based on regression trees that can be used to solve optimization problem Equation (2) for the one-step look ahead case, i.e., only with $N = 0$. Then, we present a new algorithm based on multi-output regression trees so the optimization problem Equation (1) can be solved for an arbitrary N. We finally compare in Section 5 the performance of the two approaches to show how the long-term prediction can improve the system performance.

3 CONTROL WITH REGRESSION TREES: BACKGROUND

In this section, we briefly recall the model-based control with regression trees algorithm (mbCRT) proposed in Reference [2] to synthesize optimal control strategies. We provide a general

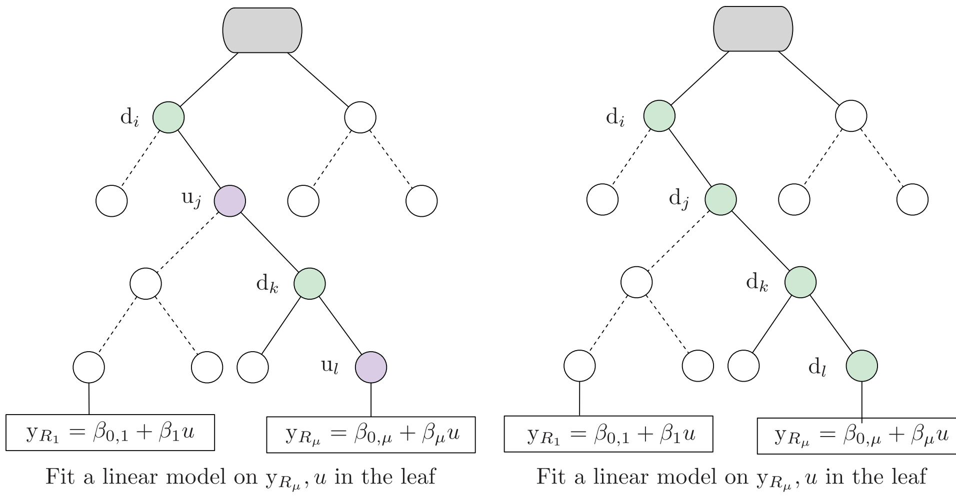


Fig. 2. Example of a regression tree not suitable for control due to the mixed order of $\mathcal{X}_c$ and $\mathcal{X}_d$ (left). Example of a tree structure obtained for the mbCRT algorithm. The separation of variables allows using the linear model in the leaf to depend only on the control variables (right).

formulation that is suitable for any cyber-physical system. This will be useful in the next section to introduce our new algorithm.

3.1 Data-driven One-step Look Ahead Predictive Model

The *model-based Control with Regression Trees (mbCRT)* uses the separation of variables introduced in Section 2. The model construction is illustrated in Figure 2—a regression tree is trained only on disturbance features in $\mathcal{X}_d = \{\mathrm{d}_1, \ldots, \mathrm{d}_{\mathrm{s-m}}\}$ to predict the output variable y. Without any loss of generality, we consider only a single response variable. Multiple trees can be considered for multiple response variables as we do for the case study in Section 5. After learning a regression tree, a linear regression model,

$$\mathrm{y}_{R_\mu} = \beta_{0,\mu} + \beta_\mu u, \tag{3}$$

is fitted using the subset of samples present in every leaf of the tree, where y_{R_μ} is the predicted response in region R_μ of the tree using all the features in $\mathcal{X}_d$, and $\beta_{0,\mu} \in \mathbb{R}$ and $\beta_\mu \in \mathbb{R}^{1\times m}$. Separation of variables allows to use the forecast of the disturbances in $\mathcal{X}_d$ to navigate to the appropriate region R_μ and use the linear regression model Equation (3) with only the control variables in it as the valid prediction model for that time-step. The linear model approximation in the leaves is validated in Reference [2]. The left part of Figure 2 shows the case where all the features in $\mathcal{X}$ are used to train the trees. In such a tree, control variables are used as the splitting variables at several nodes. As a consequence, as already explained in Section 2, this does not allow to set up an MPC-like optimization problem, since inputs values are not available a priori to go through the tree and determine the correct region R_μ.

3.2 Data-driven Optimal Control

Given p response variables and s training features, of which there are m input and $s - m$ disturbance variables, p regression trees are built to create models as in Equation (3) for every response

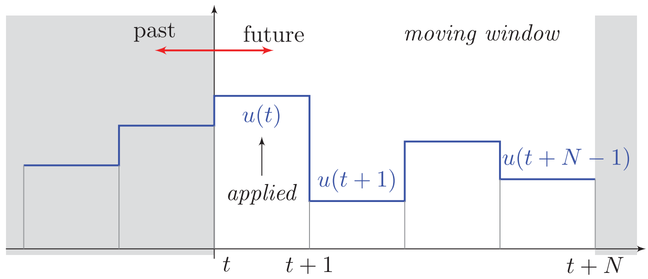


Fig. 3. Finite-horizon moving window of MPC: at time t, the MPC optimization problem is solved for a finite-length window of N steps and the first control input $u(t)$ is applied; the window then recedes one step forward and the process is repeated at time $t + 1$.

variable. Then, at each time-step t the following quadratic optimal control problem is solved:

$$\begin{aligned} \underset{u_t \in \mathcal{X}_c}{\text{minimize}} \quad & y_t^\top \mathrm{Q}\, y_t + u_t^\top \mathrm{R}\, u_t, \\ \text{subject to} \quad & \mathrm{y}_{1,t} = \beta_{0,i_1} + \beta_{i_1} u_t, \\ & \mathrm{y}_{2,t} = \beta_{0,i_2} + \beta_{i_2} u_t, \\ & \vdots \\ & \mathrm{y}_{p,t} = \beta_{0,i_p} + \beta_{i_p} u_t, \\ & u_t \in \bar{\mathcal{U}}. \end{aligned} \tag{4}$$

The optimal control input $u(t) = u_t^*$ is applied to the system, and the optimization is repeated at $t + 1$. Q and R are the design parameters to trade-off the quadratic objective function components. Different objective functions can be chosen, for example, linear or non-linear, as well as different models in the leaves instead of Equation (3), decreasing or increasing the complexity of the problem.

4 DPCRT: DATA-DRIVEN PREDICTIVE CONTROL USING REGRESSION TREES

Although the mbCRT algorithm enables control with regression trees-based models, it suffers from two significant limitations:

(1) It is based on uni-variate output regression tree models and is unable to make multi-variate predictions.
(2) It is a "one-step look ahead" algorithm and can account for an unexpected disturbance only one time-step before it occurs, thus making it sub-par as compared to receding horizon control algorithms.

The finite receding horizon control approach involves optimizing a cost function subject to the dynamics of the system and the constraints, over a finite horizon of time. After an optimal sequence of control inputs is computed, only the first input is applied to the system, then at the next step the optimization is solved again considering the new measurements, as shown in Figure 3. Our main goal is to construct data-driven predictive models for cyber-physical systems that relates the value of the response variable with the values of the predictor variables or features over an horizon with an arbitrary length N, and that can also be used to setup a receding horizon control problem. In this way, we substantially improve the simple mbCRT modeling framework.

Methods based on regression trees are *predominantly* uni-variate output, i.e., defined only for single-output variable. We introduce a different splitting criteria for trees that enables us to predict multiple outputs. If we consider these new outputs as the future states of the single-output system, then the multi-output tree enables us to implement receding horizon control as the prediction can be made for multiple steps. For example, in a building automation case study, one can consider a training dataset with information about the building states like zone temperatures, control set-points, and weather conditions, while the output could be represented by the power consumption of the building. With a single-output model, one could estimate the power consumption of the building only at time-step t. The approach we propose allows to predict the power consumption of the same building at multiple time-steps, i.e., considering a tree with $N+1$ outputs, one could estimate power consumption of the building at $t, t+1, \ldots, t+N$. This is termed as look ahead capability of a multi-variate output tree. A related case study will be considered in Section 5. In this section, we first explain how the regression trees are learned using the CART algorithm, and then modify it into a multi-variate output algorithm to create models that are suitable for finite horizon prediction. Finally, we setup a receding horizon control problem based on such models.

4.1 Predictive Modeling with Multi-output Regression Trees

We use the following notation. We consider a dataset with n observations, where each observation has s features and model has N outputs as

$$\begin{aligned} x^i &:= \left[\mathrm{x}_1^i, \ldots, \mathrm{x}_s^i\right] \in \mathbb{R}^s, \\ y^i &:= \left[\mathrm{y}_1^i, \ldots, \mathrm{y}_N^i\right] \in \mathbb{R}^N, \\ i &\in \{1, 2, \ldots, n\}. \end{aligned} \tag{5}$$

Splitting of nodes is shown in Figure 4(a). At ith node, CART splits the data set into 2 subsets. The left branch R_L contains the data corresponding to $\mathrm{x}_j \leq t_j$ and the right branch R_R corresponding to $\mathrm{x}_j > t_j$. The optimal split at each node is then determined by minimizing the sum of mean square error in both the branches:

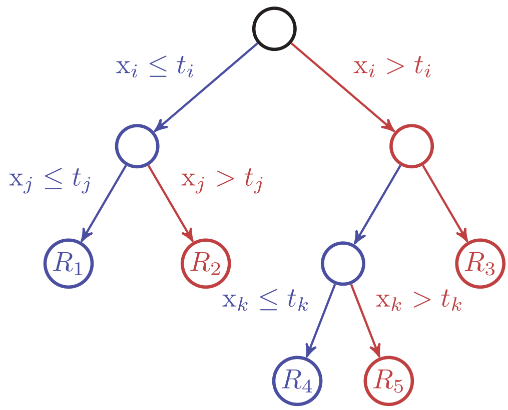
$$(\mathrm{x}_k, t_k) = \arg\min \sum_{\{i|\mathrm{x}^i \in R_L\}} \left(\mathrm{y}_1^i - \bar{y}_L\right)^2 + \sum_{\{i|\mathrm{x}^i \in R_R\}} \left(\mathrm{y}_1^i - \bar{y}_R\right)^2, \tag{6}$$

where $\bar{y}_L$ and $\bar{y}_R$ are the mean outputs of all the data points in R_L and R_R, respectively. The tree is grown in this fashion till the number of data points in the terminal nodes (leaves) exceeds the minimum number of observations in a leaf $minLeaf$, which is often a tuning parameter. Typically, a tree is grown till $minLeaf$ size is achieved, and then cost-complexity pruning is employed by collapsing the weak splits [13].

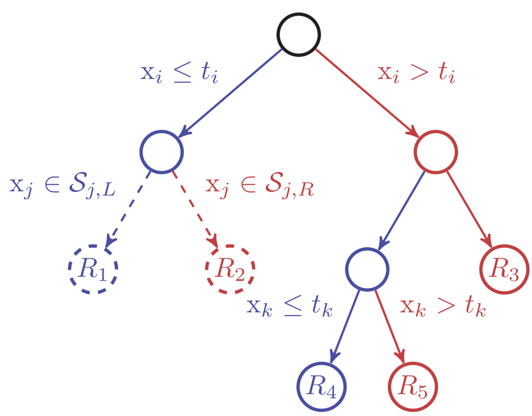
We extend the same approach to deal with the multi-output data. To determine node splits, we are again interested in calculating the splitting variable x_k and the splitting value t_k, but this time we account for errors in all N outputs. Appropriately, we modify Equation (6) as follows:

$$(\mathrm{x_k}, \mathrm{t_k}) = \arg\min \sum_{\{\mathrm{i}|\mathrm{x^i} \in \mathrm{R_L}\}} \left\|\mathrm{y^i} - \bar{\mathrm{y}}_\mathrm{L}\right\|_1^2 + \sum_{\{\mathrm{i}|\mathrm{x^i} \in \mathrm{R_R}\}} \left\|\mathrm{y^i} - \bar{\mathrm{y}}_\mathrm{R}\right\|_1^2, \tag{7}$$

where $\bar{y}_L, \bar{y}_R \in \mathbb{R}^N$, and represent the mean $\forall y^i \in R_L$ and $\forall y^i \in R_R$, respectively. Norm in this optimization criteria can be chosen to l^1 norm if we want to minimize the largest absolute error



(a) First split occurs with input x_i at t_i, second split with input x_j at t_j and so on, resulting in 5 regions in this case $R_1, \ldots, R_5$.



(b) First split occurs with continuous input x_i at t_i, second split with categorical input x_j at t_j^r such that $\mathcal{S}_{j,L} = \{t_j^1, \ldots, t_j^r\}$ and $\mathcal{S}_{j,R} = \{t_j^{r+1}, \ldots, t_j^q\}$.

Fig. 4. Tree structures for (a) only continuous and (b) mix of discrete and continuous variables.

in the outputs or l^2 norm, which will minimize the sum of squares across all the outputs. We can also introduce weights matrix $Q \in \mathbb{R}^{N \times N}$ as another tuning parameter and choose a quadratic optimization objective:

$$(\mathrm{x_k}, \mathrm{t_k}) = \arg\min \sum_{\{i | x^i \in R_L\}} (y^i - \bar{y}_L)^T Q (y^i - \bar{y}_L) + \sum_{\{i | x^i \in R_R\}} (y^i - \bar{y}_R)^T Q (y^i - \bar{y}_R). \tag{8}$$

Both Equations (7) and (8) can be solved numerically by discretizing the search space of t_k between $max(\mathrm{x}_k)$ and $min(\mathrm{x}_k)$ calculated across n data points. The finer the resolution, the better the accuracy of splits. The terminating condition for growing the tree remains unchanged in Equations (7) and (8).

So far, we covered how a tree is built when all the features/variables are continuous. It is often the case that some of the features in the data set are categorical, i.e., they can only take discrete values. The problem of partitioning a set of discrete values in two subsets is a combinatorial problem. Consider a categorical input feature x_c, which can take q different values belonging to the set $\mathcal{S}_c = \{t_c^1, \ldots, t_c^q\}$. The number of ways to partition $\mathcal{S}_c$ into two non-empty subsets are $2^{q-1} - 1$. Note that the different possible partitions scale exponentially with q, unlike in the continuous case where it grows linearly with resolution. Hence, when q is large, the exact search is not computationally easy to solve. We use a near-optimal approach to narrow down this search over all possible partitions. The approach is similar to the one described in Reference [28] for single-output system. We first find out all y^i corresponding to each element in $\mathcal{S}_c$ and then order the set $\mathcal{S}_c$ according to an increasing mean:

$$\bar{y}_q = \frac{\sum_{\{i|\mathrm{x}_c = t_c^q\}} ||y^i||_l}{N_q}, \tag{9}$$

where N_q is the number of data points for which $\mathrm{x}_c = t_q$. Once $\mathcal{S}_c = \{t_c^1, \ldots, t_c^q\}$ is ordered such that $\bar{y}_1 < \cdots < \bar{y}_q$, we split the variable as if it is a continuous variable using Equation (7) or (8), depending upon the chosen type of formulation. If the cost is minimized for $\mathrm{x}_c \leq t_c^r$, then the left branch contains $\mathrm{x}_c \in \mathcal{S}_{c,L} = \{t_c^1, \ldots, t_c^r\}$ and the right branch contains $\mathrm{x}_c \in \mathcal{S}_{c,R} = \{t_c^{r+1}, \ldots, t_c^q\}$. A tree with a mix of continuous and categorical variables is shown in Figure 4(b).

In summary, when the dataset contains both types of variables, i.e., continuous and categorical, first the range of all the categorical variables is sorted. Then, the optimal cost of splitting is determined for each input feature. Finally, the input feature for which this cost is minimum is taken as the splitting variable. Following this approach, we obtain a tree model in the form of

$$[\mathrm{y}_1, \ldots, \mathrm{y}_N] = f(\mathrm{x}_1, \ldots, \mathrm{x}_s). \tag{10}$$

4.2 Predictive Control with Multi-output Regression Trees

In this section, we setup a receding horizon control problem using models Equation (10). This algorithm is called Data-driven Predictive Control with Regression Trees (DPCRT). The central idea behind DPCRT is to build tree models that can also predict future states of the system. We still use separation of variables as in mbCRT, but train a regression tree with multiple response variables. Thus, the difference lies in the number of output variables in each leaf.

Consider p response or output variables $\mathrm{y}_1, \ldots, \mathrm{y}_p$. We build p multi-output regression trees. Each tree provides the prediction of the corresponding output variable over an horizon of length N. More precisely, using separation of variables, we first use the methodology proposed in Section 4.1 to train p multi-output regression trees with features $[\mathrm{d}_1, \ldots, \mathrm{d}_{s-m}] \in \mathcal{X}_d$, and then, we fit linear models in the leaves using input variables $u = [\mathrm{u}_1, \ldots, \mathrm{u}_m] \in \mathcal{X}_c$, obtaining the following predictive models,

$$y_j := [\mathrm{y}_{j,t}, \ldots, \mathrm{y}_{j,t+N}]^\top = \alpha_{0,i_\mu} + \alpha_{i_\mu}[u(t), \ldots, u(t+N)]^\top, \tag{11}$$

where $\alpha_{0,i_j} \in \mathbb{R}^{(N+1)\times 1}$ and $\alpha_{i_j} \in \mathbb{R}^{(N+1)\times m}$ are model coefficients associated with the region/leaf R_μ of the jth output variable. Note that the training procedure can also fit a more complex, e.g., non-linear, model of the input variables instead of Equation (11), but this worsens the computational complexity. We show in Section 5 that the linear models are accurate. Once the model is trained, we can setup the following receding horizon control problem

$$
\begin{aligned}
\underset{u_{t+k} \in \mathcal{X}_c}{\text{minimize}} \quad & \sum_{k=0}^{N} \mathrm{y}_{t+k}^{\top} \mathrm{Q}\, \mathrm{y}_{t+k} + u_{t+k}^{\top} \mathrm{R}\, u_{t+k}, \\
\text{subject to} \quad & \mathrm{y}_{1,t+k} = \alpha_{0,i_1} + \alpha_{i_1}[u_t, \dots, u_{t+k}]^{\top}, \\
& \mathrm{y}_{2,t+k} = \alpha_{0,i_2} + \alpha_{i_2}[u_t, \dots, u_{t+k}]^{\top}, \\
& \vdots \\
& \mathrm{y}_{p,t+k} = \alpha_{0,i_p} + \alpha_{i_p}[u_t, \dots, u_{t+k}]^{\top}, \\
& u_{t+k} \in \bar{\mathcal{U}}, \\
& k = 0, \dots, N.
\end{aligned} \tag{12}
$$

We solve this optimization as in the classical MPC formulation, i.e., we apply only the first optimal control input $u(t) = u_t^*$ and proceed to the next time step, where we run the algorithm again with the new measurements.

If the number of control variables is large, then the optimization problem Equation (11) may require many data points in the leaves, or in other words a large $minLeaf$, which can affect the accuracy of the regression tree. Therefore, we also introduce a variant of this algorithm that will ease the selection of a lower $minLeaf$. To do this, we define $h_\nu(y_1, \dots, y_p),\ \nu = 1, 2, \dots,$ as some desired functions that relate variables $y_j,\ j = 1, \dots, p$, and setup the following problem:

$$
\begin{aligned}
\underset{u_{t+k} \in \mathcal{X}_c}{\text{minimize}} \quad & \sum_{k=0}^{N} g(h_1, \dots, h_m) + u_{t+k}^{\top} \mathrm{R}\, u_{t+k}, \\
\text{subject to} \quad & h_1(y_1, \dots, y_p) = \bar{\alpha}_{0,1} + \bar{\alpha}_1[\mathrm{u}_{1,t}, \dots, \mathrm{u}_{1,t+k}]^{\top}, \\
& h_2(y_1, \dots, y_p) = \bar{\alpha}_{0,2} + \bar{\alpha}_2[\mathrm{u}_{2,t}, \dots, \mathrm{u}_{2,t+k}]^{\top}, \\
& \vdots \\
& h_m(y_1, \dots, y_p) = \bar{\alpha}_{0,m} + \bar{\alpha}_m[\mathrm{u}_{m,t}, \dots, \mathrm{u}_{m,t+k}]^{\top}, \\
& \mathrm{u}_{j,t+k} \in \bar{\mathcal{U}},\ j = 1, \dots, m, \\
& k = 0, \dots, N,
\end{aligned} \tag{13}
$$

where $\bar{\alpha}_{0,\nu} \in \mathbb{R}$ and $\bar{\alpha}_\nu \in \mathbb{R}^{1 \times k}$. For example, h_ν can simply be a linear combination of variables y_j and depends only on the control variable u_ν. With a suitable choice of g, both RHC problem Equations (12) and (13) are convex.

Our algorithm for DPC with regression trees in summarized in Algorithm 1 and a schematic is shown in Figure 5. During the training process, the tree is learned only on disturbance variables with linear models in the leaves, which are a function only of control variables. During the control step, at time t, the disturbance features $\mathcal{X}_d$ are known for that instant and thus the leaf R_μ is known. The optimal control problem is solved to determine the optimal control variables $[u_t^*, \dots, u_{t+N}^*]$. Only the first input $u(t) = u_t^*$ is applied to the system. The resulting outputs $\mathrm{y}_j(t)$, which form the features for the next time-step, are used to determine the next R_μ. In the context of a building model, we show the efficacy of DPCRT in Section 5.

5 CASE STUDY: PEAK POWER REDUCTION

In this section, we compare performance of DPCRT against mbCRT with a case study on peak power reduction. To this aim, we use the DoE Commercial Reference Building (Doe CRB) simulated

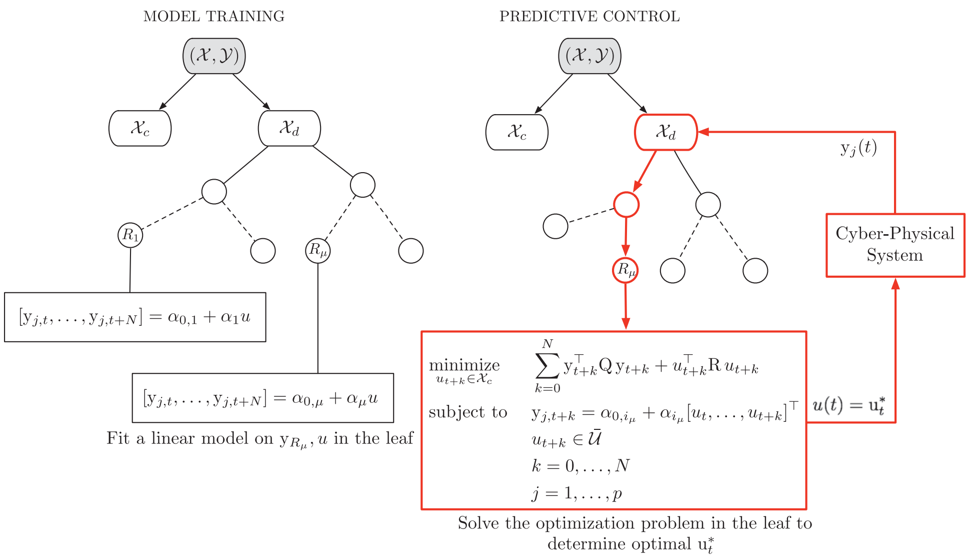


Fig. 5. Data Predictive Control with Regression Trees with Model Training Process (left) and Receding Horizon Control (right). During the control step, the optimal control sequence $[u_t^*, \dots, u_{t+N}^*]^\top$ is determined. The first input $u(t) = u_t^*$ is applied to the system. The resulting output y_t, which is a feature for the next time step, is fed back to determine to determine R_i at $t + 1$.

ALGORITHM 1: DPCRT: Data-driven Predictive Control with Regression Trees

---

DESIGN TIME
procedure MODEL TRAINING USING SEPARATION OF VARIABLES
  Set $\mathcal{X}_c \leftarrow$ control variables
  Set $\mathcal{X}_d \leftarrow$ disturbance variables
  Build predictive trees using dataset $(\mathcal{Y}, \mathcal{X}_d)$ using (10)
  for all regions R_μ at the leaves ofeach tree **do**
    Fit linear model as in (11)
  end for
end procedure
RUN TIME
procedure PREDICTIVE CONTROL
  while $t < t_{\text{stop}}$ **do**
    Determine the leaf and region R_μ using current measurements of $\mathcal{X}_d$
    Obtain the linear model at R_μ
    Solve optimization in (12) or (13) to determine optimal control sequence $[u_t^*, \dots, u_{t+N}^*]^\top$
    Apply the first input $u(t) = u_t^*$
  end while
end procedure

---

in EnergyPlus [8] as the virtual test-bed building. We first describe our test-bed and introduce the data we use, then validate the quality of the DPCRT modeling on the power consumption prediction of DoE CRB, and finally compare DPCRT and mbCRT on the peak power reduction problem.

5.1 Test-bed Description

DoE CRB is a large 12-story office building consisting of 19 zones with a total area of 500,000 sq.ft. There are 2,397 people in the building during peak occupancy. During peak load conditions the building can consume up to 1.6 MW of power. For the simulation of the DoE CRB building, we use actual meteorological year data from Chicago for the years 2012 and 2013. The dataset we use for training the trees can be divided into four different categories:

(1) **Weather Data:** $w = [\mathrm{w}_1, \ldots, \mathrm{w}_d]$. These data include measurements of the outside dry-bulb and wet-bulb air temperature, relative humidity, and wind characteristics. Since we are interested in predicting the power consumption for a finite horizon, we include the weather forecast of the complete horizon in the training features.

(2) **Schedule Data:** $s = [\mathrm{s}_1, \ldots, \mathrm{s}_r]$. We create proxy variables that correlate with repeated patterns of electricity consumption, e.g. due to occupancy or equipment schedules. For example, Day of Week is a categorical predictor that takes values from 1 to 7 depending on the day of the week. This variable can capture any power consumption patterns that occur on specific days of the week. Likewise, Time of Day is quite an important predictor of power consumption as it can adequately capture daily patterns of occupancy, lighting and appliance use without directly measuring any one of them. Besides using proxy schedule predictors, actual building equipment schedules can also be used as training data for building the trees.

(3) **Building Data:** $b = [\mathrm{b}_1, \ldots, \mathrm{b}_m]$. These data include the states of the building, such as Chilled Water Supply Temperature, Hot Water Supply Temperature, Zone Air Temperature, Supply Air Temperature, and Lighting levels.

(4) **Power Consumption:** P. This is the response variable, in addition to zone temperatures.

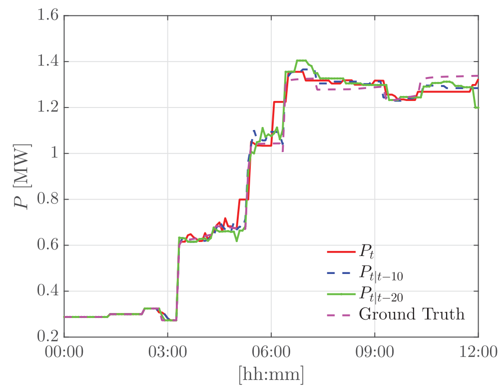
To predict the power consumption of the building for the entire length of the horizon, we use the notion of auto-regressive trees. An auto-regressive tree is a regular regression tree except that the lagged values of the response variable are also predictor variables for the regression tree, i.e., the tree structure is learned to provide the following model as in Equation (1),

$$P_t = f\left(w(t), s(t), b(t), P(t-1), \ldots, P(t-\delta)\right), \tag{14}$$

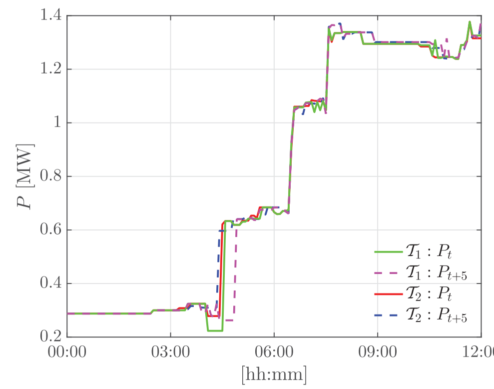
where δ is the order of the auto-regression and $P(t-j)$ is the value of the power measured at time $t-j$. This allows us to make finite horizon predictions of power consumption for the building.

5.2 DPCRT Modeling Validation

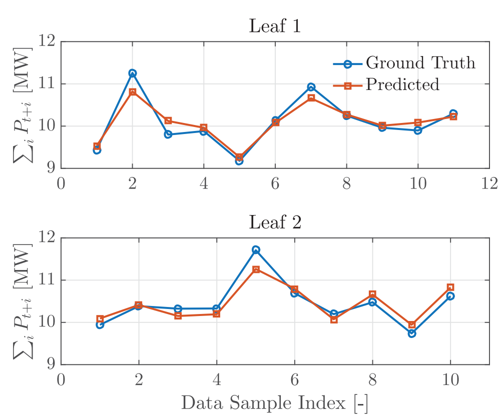
Applying Equation (8), we build multi-variate regression trees using a training dataset from July 2012. For a tree with order of auto-regression $\delta = 6$, a prediction horizon $N = 20$, and Q equal to the identity matrix, the results on the test dataset are shown in Figure 6(a). The test set shows a day from July 2013. In particular, we compare the building power consumption P_t predicted at time t, with the actual power consumption of the building from the test dataset. Since we can predict the power for multiple steps of the horizon, we add to the comparison the power prediction at time t computed at $t-10$ and $t-20$, i.e., $P_{t|t-10}$ and $P_{t|t-20}$, respectively. It can be seen that even with a relatively long horizon, the multi-variate tree model captures the rapid changes in the response variable very accurately. DPCRT uses only a subset of features to train the tree, while the inputs are used to train models in the leaves. Thus, the performance of DPCRT depend on two key assumptions



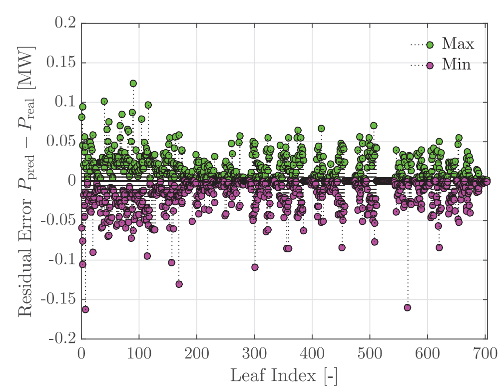
(a) Building power consumption at time t predicted at time t is denoted by $P(t)$, predicted 10 steps ahead at time $t-10$ is denoted by $P_{t|t-10}$, and predicted 20 steps ahead at time $t-20$ is denoted by $P_{t|t-20}$.



(b) A comparison of power consumption of the building for two different regression trees. $\mathcal{T}_1$ is trained on all the features while $\mathcal{T}_2$ is trained only on non-manipulated features using separation of variables.



(c) Model validation for linear regression at the leaves of the tree. The predicted and the actual power consumption are very close.



(d) The tree has 703 leaves. For each leaf, a maximum and a minimum difference in prediction of average power consumption over the control horizon $P_{\text{pred}} - P_{\text{real}}$ is calculated from the data points that end up in that leaf.

Fig. 6. Model Validation of DPCRT.

- the separation of variables does not introduce significant errors while training the tree, and
- the linear regression at the leaves is a valid assumption.

We verify the validity of these assumptions in terms of their effect on model accuracy considering the following two cases:

(1) We train two kinds of regression trees: $\mathcal{T}_1$, that is trained using all the features described above, and $\mathcal{T}_2$, that was learned from disturbance variables only, with a linear model on the control variables at the leaves as in Equation (11). The predicted power consumption of the building at time t and $t+5$, i.e., P_t and P_{t+5}, for both trees is shown in Figure 6(b). The normalized root mean square error (NRMSE) for these two outputs on the test dataset is shown in Table 1. We notice a small loss in model accuracy with $\mathcal{T}_2$ due to the separation of variables. This is the cost we have to pay for integrating control synthesis with the tree,

Table 1. NRMSE for Regression Trees with and without Control Features

	P_t	P_{t+1}	P_{t+2}	P_{t+3}	P_{t+4}	P_{t+5}
$\mathcal{T}_1$	0.1037	0.1036	0.1116	0.1124	0.1140	0.1164
$\mathcal{T}_2$	0.1156	0.1182	0.1270	0.1268	0.1324	0.1308

since otherwise the control features would have been a part of the splitting criteria rather than a linear model in the leaves of the tree.

(2) For the tree $\mathcal{T}_2$, we fit a linear model on the sum of all outputs, i.e., the sum of power consumption over the complete control horizon. For each leaf i, we have

$$P_t + \cdots + P_{t+N} = \beta_{0,i} + \beta_i^\top u. \tag{15}$$

For two randomly selected leaves, the fit of the linear model against the actual power consumption is shown in Figure 6(c). The error observed in the predicted and actual power consumption for all leaves is shown in Figure 6(d). It can be seen that for a small number of samples in the leaves of the tree the linear model assumption is valid.

5.3 DPCRT for Peak Power Reduction

The key to answering the question of what actions to take to achieve a significant DR curtailment upon receiving a notification lies in making accurate predictions about the power consumption response of the building. In this context, we evaluate the performance of DPCRT for peak power reduction in buildings, and we show the advantages of receding horizon control with regression trees (DPCRT) with respect to the one-step lookahead control (mbCRT).

As already described, the data samples consist of four types of features, namely weather data, schedule data, building data, and autoregressive terms of building power consumption. We use the following three features from the building data as control variables: zone temperature cooling set point u_c [°C], chilled water temperature set point u_h [°C], and lighting set point u_ℓ (ranges from 0 to 1). Since the power consumption model also depends on the zone temperatures, to have a prediction of the power over the horizon, we first need to predict zone temperatures. To this aim, we built two kinds of tree, with output as power and zone temperature (one for each zone). In the leaf of each tree, we further have three kinds of models as described below.

- The power tree is learned using all the features except for the inputs. In the leaf of the tree, following linear models are trained:
  - P_j^c, power consumption at time j due to u_c. Only u_c have been used to fit the linear model in the leaves.
  - P_j^h, power consumption at time j due to u_h. Only u_h have been used to fit the linear model in the leaves.
  - P_j^ℓ, power consumption at time j due to u_ℓ. Only u_ℓ have been used to fit the linear model in the leaves.
- The temperature tree for the ith zone is learned using all the features except for the inputs and except for the room temperatures r, $\forall r \neq i$.
  - $T_{i,j}^c$, temperature of the ith zone at time j due to u_c. Only u_c have been used to fit the linear model in the leaves.
  - $T_{i,j}^h$, temperature of the ith zone at time j due to u_h. Only u_h have been used to fit the linear model in the leaves.

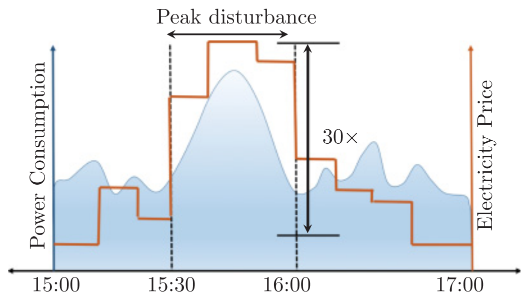


Fig. 7. Peak disturbance between 1530 and 1600h. The test scenario used in the DPCRT case study simulations has a 1.5× peak disturbance between 15:30 and 16:00.

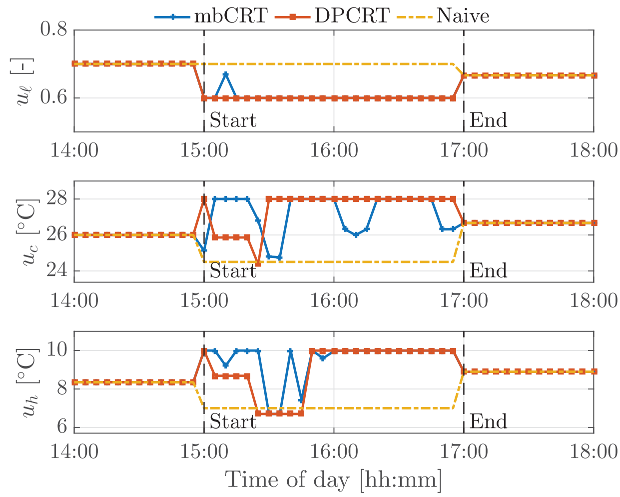
— $T^{\ell}_{i,j}$, temperature of the ith zone at time j due to u_ℓ. Only u_ℓ have been used to fit the linear model in the leaves.

With u defined as $u = [\mathrm{u}_c, \mathrm{u}_h, \mathrm{u}_\ell]^\top$, we can setup the following DPCRT problem to minimize the peak power consumption,

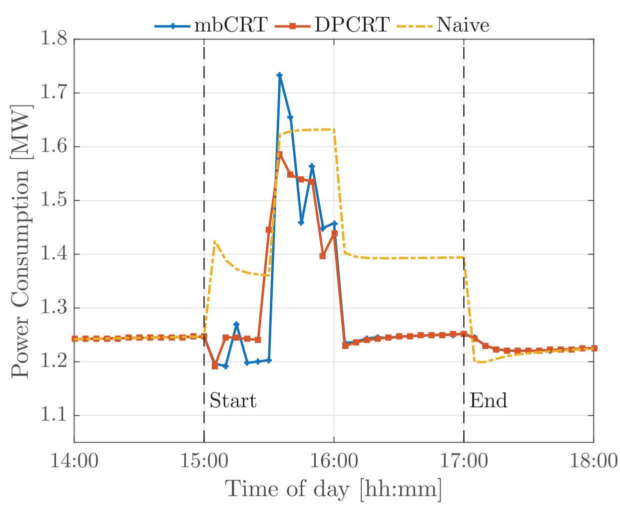
$$\begin{aligned}
\underset{u_{t+k}}{\text{minimize}} \quad & \sum_{k=0}^{N} \frac{P^c_{t+k} + P^h_{t+k} + P^\ell_{t+k}}{3} + \sum_{k=0}^{N}\sum_{i=1}^{19} \lambda_i \left(\frac{T^c_{i,t+k} + T^h_{i,t+k} + T^\ell_{i,t+k}}{3} - T_{ref} \right)^2, \\
\text{subject to} \quad & T^c_{i,t+k} = \alpha^c_{0,i} + \alpha^c_i [\mathrm{u}_{c,t}, \ldots, \mathrm{u}_{c,t+k}]^\top, i = 1, \ldots, 19, \\
& T^h_{i,t+k} = \alpha^h_{0,i} + \alpha^h_i [\mathrm{u}_{h,t}, \ldots, \mathrm{u}_{h,t+k}]^\top, i = 1, \ldots, 19, \\
& T^\ell_{i,t+k} = \alpha^\ell_{0,i} + \alpha^\ell_i [\mathrm{u}_{\ell,t}, \ldots, \mathrm{u}_{\ell,t+k}]^\top, i = 1, \ldots, 19, \\
& \sum_{k=0}^{N} P^c_{t+k} = \gamma^c_0 + \gamma^c_1 [\mathrm{u}_{c,t}, \ldots, \mathrm{u}_{c,t+k}]^\top, \\
& \sum_{k=0}^{N} P^h_{t+k} = \gamma^h_0 + \gamma^h_1 [\mathrm{u}_{h,t}, \ldots, \mathrm{u}_{h,t+k}]^\top, \\
& \sum_{k=0}^{N} P^\ell_{t+k} = \gamma^\ell_0 + \gamma^\ell_1 [\mathrm{u}_{\ell,t}, \ldots, \mathrm{u}_{\ell,t+k}]^\top, \\
& u_{t+k} \in [u_{\min}, u_{\max}], \\
& k = 0, \ldots, N.
\end{aligned} \tag{16}$$

At time t, we solve DPCRT Equation (16) and apply only the first input, i.e., $\mathrm{u}_c(t) = \mathrm{u}^*_{c,t}$, $\mathrm{u}_h(t) = \mathrm{u}^*_{h,t}$ and $\mathrm{u}_\ell(t) = \mathrm{u}^*_{\ell,t}$. At $t+1$, we repeat the algorithm with the updated measurements.

We compare DPCRT and mbCRT when simulated in closed-loop with the EnergyPlus model of the building. To compare the performance of DPCRT, we consider a scenario in which there is a significant disturbance that is only anticipated 30min in advance and leads to a sudden increase in zone temperatures in the building. This may be due to a sudden spike in the occupancy or equipment being switched ON at a brief notice. Under this scenario, it is important to react to the disturbance in a predictive manner to minimize the peak power consumption. This scenario is shown in Figure 7. Between 15:30 and 16:00, an enormous spike in the power consumption (1.5×) is expected because of a scheduled operation. The control strategies are tested over a 2h duration between 15:00 and 17:00. Figure 8(a) shows three control strategies: mbCRT, DPCRT, and a naïve



(a) Control strategies.



(b) Power consumption.

Fig. 8. A comparison between mbCRT and DPCRT along with a naïve rule-based strategy.

load reduction strategy. The naïve strategy is equivalent to not responding to the disturbance at all. It maintains the desired zone temperature set point u_c, chiller water temperature set point u_h, and lighting level u_ℓ throughout the test period. In Figure 8(b), it can be seen that DPCRT reacts to the disturbance much before mbCRT, which waits until the last time-step before the disturbance to react. This leads to a significantly lower peak power consumption that mbCRT. In the case of DPCRT, the control horizon is 6. At 15:00, DPCRT strategy is same as the greedy one. At 15:05, the downstream disturbance is visible to DPCRT algorithm and it starts to pre-cool the building by decreasing both cooling and chiller water set points. At 15:25, the u_c and the u_h are reduced to minimum so that in the period of extreme disturbance an optimal trade-off between power consumption and thermal comfort is maintained. Thus, DPCRT algorithm foresees the disturbance and takes a preemptive action against it. On the other hand, the mbCRT algorithm considers the

Table 2. Quantitative Comparison for Energy Consumption, Peak Power, and % Reduction in Peak Power of DPCRT Compared to Naïve Approach and mbCRT

	Energy [kWh]	Peak Power [MW]	Peak Reduction
Naïve	5,358	1.63	3.1%
mbCRT	5,097	1.73	**8.6%**
DPCRT	5,102	1.58	—

power consumption and the zone temperatures of only one time step. Therefore, it does not know of an upcoming disturbance. At every time step, it chooses u_c, u_h, and u_ℓ, which optimizes the cost for that time step. Naturally, this leads to a jaggy behavior in the control strategy. We can see a similar behavior for the power consumption in Figure 8(b). The DPCRT algorithm gradually increases the power consumption, because it can see the disturbance before it actually reaches 15:30, while in the case of mbCRT, the power consumption overshoots by a big margin, because the controller deals with the disturbance in a single step. DPCRT maintains zone temperature much closer to the reference temperature of 24°C while both mbCRT and the naïve strategy have large deviations from the desired temperature.

The quantitative comparison is presented in Table 2. Between 15:00 and 17:00, DPCRT and mbCRT result in similar energy usage, 5,102 and 5,097kWh, respectively, both outperforming the naïve strategy that incurs 5,358kWh. Peak power in the case of DPCRT is 1.58MW, which is lower than both mbCRT (1.73MW) and the naïve strategy (1.63MW), although naïve outperforms mbCRT. The peak power with DPCRT is 8.6% less than mbCRT and 3.1% less than the naïve. While both DPCRT and mbCRT account for thermal comfort, DPCRT deviates less from the desired temperature. The naïve strategy does not trade-off on the thermal comfort. Thus, DPCRT outperforms mbCRT both in terms of a reduced peak power consumption and better thermal comfort.

6 CONCLUSION

We present a data-driven algorithm for control-oriented modeling and receding horizon control – data-driven predictive control with regression trees (DPCRT). DPCRT enables optimal control designs for complex problems that otherwise are dependent on expensive first principles or physics-based models. We apply DPCRT to the problem of peak power reduction in buildings in context of Demand Response. The performance of DPCRT are evaluated on a DoE commercial reference virtual test-bed. The results show that it provides lower power consumption while maintaining a desired thermal comfort level for every zone. In the considered example, DPCRT leads to 8.6% decrease in the peak power consumption of the building when compared to the mbCRT algorithm (one-step look ahead) and 3.1% decrease when compared to a naïve rule-based reduction strategy. These advantages make DPCRT an alluring tool for evaluating and planning DR curtailment responses for large scale cyber-physical energy systems. The approach we propose in this article modifies the CART algorithm considering the error minimization of the system evolution over a predictive horizon. This results in a multi-output regression tree modeling that is extremely simple from the computational complexity point of view. To improve the accuracy of our algorithms, new methods based on Random Forests are currently under investigation. Preliminary results can be found in References [18, 19].

REFERENCES

[1] Madhur Behl, Achin Jain, and Rahul Mangharam. 2016. Data-driven modeling, control and tools for cyber-physical energy systems. In *Proceedings of the 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS'16).* IEEE, 1–10.

[2] Madhur Behl, Francesco Smarra, and Rahul Mangharam. 2016. DR-advisor: A data-driven demand response recommender system. *Appl. Energy* 170 (2016), 30–46. DOI: http://dx.doi.org/10.1016/j.apenergy.2016.02.090

[3] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* 38, 1 (2002), 3–20.

[4] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. 1984. *Classification and Regression Trees.* CRC Press.

[5] L. Cavagnari, Lalo Magni, and Riccardo Scattolini. 1999. Neural network implementation of nonlinear receding-horizon control. *Neural Comput. Appl.* 8, 1 (1999), 86–92.

[6] G. T. Costanzo, S. Iacovella, F. Ruelens, T. Leurs, and B. J. Claessens. 2016. Experimental analysis of data-driven control for a building heating system. *Sustain. Energy Grids Netw.* 6 (2016), 81–90.

[7] Drury B. Crawley, Linda K. Lawrie, et al. 2001. EnergyPlus: Creating a new-generation building energy simulation program. *Energy and Buildings* 33, 4 (2001), 319–331.

[8] Michael Deru, Kristin Field, Daniel Studer, Kyle Benne, Brent Griffith, Paul Torcellini, Bing Liu, Mark Halverson, Dave Winiarski, Michael Rosenberg, et al. 2011. U.S. Department of Energy commercial reference building models of the national building stock.

[9] Richard E. Edwards, Joshua New, and Lynne E. Parker. 2012. Predicting future hourly residential electrical consumption: A machine-learning case study. *Energy Build.* 49 (2012), 591–603.

[10] Federal Energy Regulatory Commission et al. 2012. Assessment of demand response and advanced metering.

[11] P. M. Ferreira, A. E. Ruano, S. Silva, and E. Z. E. Conceicao. 2012. Neural networks-based predictive control for thermal comfort and energy savings in public buildings. *Energy Build.* 55 (2012), 238–251.

[12] NOAA National Centers for Environmental Information. State of the Climate: Global Analysis for August 2015. Retrieved from http://www.ncdc.noaa.gov/sotc/global/201508.

[13] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. 2005. The elements of statistical learning: Data mining, inference and prediction. *Math. Intell.* 27, 2 (2005), 83–85.

[14] Kyoung ho Lee and James E. Braun. 2008. Development of methods for determining demand-limiting setpoint trajectories in buildings using short-term measurements. *Build. Environ.* 43, 10 (2008).

[15] David G. Holmberg, Girish Ghatikar, Edward Koch, and Jim Boch. 2012. OpenADR advances. *ASHRAE J.* 54 (11/2012 2012).

[16] Z.-S. Hou and Z. Wang. 2013. From model-based control to data-driven control: Survey, classification and perspective. *Info. Sci.* 235 (2013), 3–35.

[17] Achin Jain, Madhur Behl, and Rahul Mangharam. 2016. Data predictive control for peak power reduction. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys).*

[18] Achin Jain, Madhur Behl, and Rahul Mangharam. 2017. Data predictive control for building energy management. In *Proceedings of the 2017 American Control Conference.* IEEE.

[19] Achin Jain, Francesco Smarra, and Rahul Mangharam. 2017. Data predictive control using regression trees and ensemble learning. In *Proceedings of the 2017 Conference on Decision and Control.* IEEE.

[20] S. Kiliccote, M. A. Piette, and D. Hansen. 2006. Advanced controls and communications for demand response and energy efficiency in commercial buildings. In *Proceedings of the 2nd Carnegie Mellon Conference in Electronic Power Systems.*

[21] Andrew Kusiak, Zhe Song, and Haiyang Zheng. 2009. Anticipatory control of wind turbines with data-driven predictive models. *IEEE Trans. Energy Conv.* 24, 3 (2009), 766–774.

[22] PJM Interconnection and Michael J. Kormos. 2014. PJM response to consumer reports on 2014 winter pricing. Retrieved from http://www.pjm.com/-/media/documents/reports/20140919-pjm-response-to-consumer-reports-on-2014-winter-pricing.ashx?la=en.

[23] Navigant Research. 2015. Demand response for commercial & industrial markets market players and dynamics, key technologies, competitive overview, and global market forecasts. Retrieved from https://www.prnewswire.com/news-releases/demand-response-for-commercial--industrial-markets--market-players-and-dynamics-key-technologies-competitive-overview-and-global-market-forecasts---reportlinker-review-300151285.html.

[24] F. Oldewurtel et al. 2013. Towards a standardized building assessment for demand response. In *Proceedings of the IEEE Conference on Decision and Control.*

[25] Alberto Nai Oleari, Jose Ramon D. Frejo, Eduardo F. Camacho, and Antonella Ferrara. 2015. A model predictive control scheme for freeway traffic systems based on the classification and regression trees methodology. In *Proceedings of the, 2015 European Control Conference (ECC'15).* IEEE, 3459–3464.

[26] PJM Interconnection. 2014. 2014 demand response operations markets activity report. Retrieved from https://so-ups.ru/fileadmin/files/company/markets/dr/dr_pjm/dr_pjm_otchet_2014.pdf.

[27] Joshua New, Jibonananda Sanyal, Mahabir Bhandari, and Som Shrestha. 2012. Autotune e+ building energy models. *Proceedings of the 5th National SimBuild of IBPSA-USA*.

[28] Brian D. Ripley. 2007. *Pattern Recognition and Neural Networks*. Cambridge University Press.

[29] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith. 2015. Model predictive climate control of a swiss office building: Implementation, results, and cost-benefit analysis. *IEEE Trans. Control Syst. Technol.* PP, 99 (2015), 1–1. DOI: http://dx.doi.org/10.1109/TCST.2015.2415411

[30] A. Vaghefi, M. A. Jafari, E. Bisse, Y. Lu, and J. Brouwer. 2014. Modeling and forecasting of cooling and electricity load demand. *Appl. Energy* 136 (2014), 186–196.

[31] Peng Xu, Philip Haves, Mary Ann Piette, and James Braun. 2004. Peak demand reduction from pre-cooling with zone temperature reset in an office building. *Berkeley National Laboratory* (2004).

[32] Wei Yin, Yogesh Simmhan, and Viktor K. Prasanna. 2012. Scalable regression tree learning on hadoop using OpenPlanet. In *Proceedings of 3rd International Workshop on MapReduce and Its Applications Date*. ACM, 57–64.

[33] Eva Zavcekova, Zdenvek Vavna, and Jivri Cigler. 2014. Towards the real-life implementation of MPC for an office building-identification issues. *Appl. Energy* 135 (2014), 53–62.

Received July 2016; revised April 2017; accepted July 2017