Data Predictive Control for Peak Power Reduction

Achin Jain and Rahul Mangharam
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104, USA.
[achinj,rahulm]@seas.upenn.edu

Madhur Behl
Computer Science
University of Virginia
Charlottesville, VA 22903, USA.
mb2kg@virginia.edu

ABSTRACT

Decisions on how best to optimize today's energy systems operations are becoming ever so complex and conflicting such that model-based predictive control algorithms must play a key role. However, learning dynamical models of energy consuming systems such as buildings, using grey/white box approaches is very cost and time prohibitive due to its complexity. This paper presents data-driven methods for making control-oriented model for peak power reduction in buildings. Specifically, a data predictive control with regression trees (DPCRT) algorithm, is presented. DPCRT is a finite receding horizon method, using which the building operator can optimally trade off peak power reduction against thermal comfort without having to learn white/grey box models of the systems dynamics. We evaluate the performance of our method using a DoE commercial reference virtual test-bed and show how it can be used for learning predictive models with 90% accuracy, and for achieving 8.6% reduction in peak power and costs.

CCS Concepts

•Computing methodologies → Classification and regression trees;

Keywords

Machine learning; Predictive control; Building control; Peak power reduction.

1. INTRODUCTION

The organized electricity markets in the United States all use some variant of real-time or time-of-use (TOU) pricing for wholesale electricity. For e.g. PJM's real-time market is a spot market where electricity prices are calculated at five-minute intervals based on the grid operating conditions. 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, industries and institutions [1]; often referred to as $C/I/I$ consumers. 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 PJM electricity grid experienced a 86 fold increase in the price of electricity from \$31/MW h to \$2,680/MW h in a matter of a few minutes [19]. Similarly, the summer price spiked 32 fold from an average of \$25/MW h to \$800/MW h in July of 2015.

Buildings consume more than one third of the world's total primary energy [15]. They account for 40% of all energy use in the U.S and for 72% of total U.S. electricity consumption [7]. Large $C/I/I$ buildings, are the biggest consumers of electricity and a significant contributor to peak load conditions on the grid. The largest use of energy consumption in buildings is attributed to the heating, ventilation and air-conditioning (HVAC) systems. Therefore, measures directed at the HVAC systems are attractive for load curtailment and energy-efficient building operation.

Control-oriented predictive models of an energy system's dynamics and energy consumption, are needed for understanding and improving the overall energy efficiency and operating costs. With a reasonably accurate forecast of future weather and building operating conditions, dynamical models can be used to predict the energy needs of the building over a prediction horizon, as is the case with model predicitve control (MPC). However, a major challenge with MPC is in accurately modeling the dynamics of the underlying physical system.

Learning predictive models of building's dynamics using first principles based approaches (e.g. with EnergyPlus [8]) is very cost and time prohibitive and requires retrofitting the building with several sensors [26]. The user expertise, time, and associated sensor costs required to develop a model of a single building is very high. This is because usually a building modeling domain expert typically uses a software tool to create the geometry of a building from the building design and equipment layout plans, add detailed information about material properties, about equipment and operational schedules. There is always a gap between the modeled and the real building and the domain expert must then manually tune the model to match the measured data from the building [23].

The alternative is to use black-box, or completely data-driven modeling approaches, to obtain a realization of the system's input-output behavior. The primary advantage of using data-driven methods is that it has the potential to eliminate the time and effort required to build white and

BuildSys '16, November 16-17, 2016, Palo Alto, CA, USA
© 2016 ACM. ISBN 978-1-4503-4264-3/16/11…\$15.00
DOI: http://dx.doi.org/10.1145/2993422.2993582

grey box building models. Listening to real-time data, from existing systems and interfaces, is far cheaper than unleashing hoards of on-site engineers to physically measure and model the building. Improved building technology and better sensing is fundamentally redefining the opportunities around smart buildings. Unprecedented amounts of data from millions of smart meters and thermostats installed in recent years has opened the door for systems engineers and data scientists to analyze and use the insights that data can provide, about the dynamics and power consumption patterns of these systems. The challenge now, with using data-driven approaches, is to close the loop for real-time control and decision making for large $C/I/I$ buildings .

In our previous work [3], we developed and evaluated a model based control with regression trees (mbCRT) algorithm which enables closed-loop control of buildings for demand response while using regression trees based, data-driven predictive models. In this paper we present a new approach with significant improvements over mbCRT [3] for implementing finite receding horizon control using regression trees based data-driven models.

This work has the following data-driven contributions:

1. We present a method for constructing a multi-variate output predictive model using regression trees. This is via node level optimization for determining variable selection and splitting criteria. The proposed algorithm achieves an accuracy of 90% on test data set.

2. We address the limitations of mbCRT, and present a data predictive control with regression trees (DPCRT) algorithm for finite receding horizon control. DPCRT bypasses the cost and time prohibitive process of building high fidelity models of buildings that use grey and white box modeling approaches while still being suitable for receding horizon control design (like MPC). In application to peak power curtailment, we observe that DPCRT beats mbCRT by 8.6%.

The DPCRT algorithm is first of its kind that does finite receding horizon control with regression trees. We evaluate the performance of our methods using a U.S. Department of Energy (DoE) commercial reference building model.

The paper is organized as follows. In Sec. 2, our approach to predictive modeling with a multi-output regression tree is described. Sec. 3 presents the finite receding horizon control for regression tree algorithm. The performance of DPCRT is compared to mbCRT in Sec. 4 , using a realistic commercial building simulation. Sec. 5 provides an overview and a comparison to existing work in this area. We conclude the paper with a summary of the results and a brief discussion of future work in Sec. 6.

2. REGRESSION TREES WITH MULTI-VARIATE OUTPUT

Our goal is to construct data-driven functional models that relates the value of the response variable, say power consumption, $\mathbb{Y}_{kW}$ with the values of the predictor variables or features $[\mathbb{X}_1, \cdots, \mathbb{X}^n]$ which can include weather data, set-point information and building schedules. When the data has lots of features, as is the case in 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, where the interactions are more manageable. We then partition the partitions again; this is called recursive partitioning, until finally we get to chunks of the data space which are so tame that we can fit simple models to them. Regression trees is an example of an algorithm which belongs to the class of recursive partitioning algorithms. The seminal algorithm for learning regression trees is CART as described in [5]. Some of its modifications include MARS [11] and C4.5 algorithm [22]. However, regression tree based methods are predominantly univariate output, i.e. defined only for single output variable. We describe a splitting criteria for the trees which enables us to predict multiple outputs [25]. If we consider these new outputs as the future states of the single output system, the multi-output tree enables us to implement receeding horizon control as the prediction can be made for multiple steps. For example, consider a training dataset with information about the building states like zone temperatures, control set points and ambient weather. The output we are interested in is the power consumption of the building. With a single output model, we can estimate the power consumption of the building at only one time step T. The new approach allows us to predict the power consumption of the same building at multiple time steps, i.e. with a tree with p outputs, we can estimate the power consumption at $T, T+1, \ldots, T+p$. This is termed as look-ahead capability of a multi-variate output tree. We will consider this example in detail in Sec. 4.

In this section, we first explain how a CART based regression tree is built, and then modify it into a multi-variate output model suitable for finite horizon prediction.

2.1 Model Construction

We use the following notation. We represent a dataset with N observations, where each input has n features and model has p outputs as

$$\begin{aligned} x_i &:= [x_i^1, \ldots, x_i^n]^T \in \mathbb{R}^n, \\ y_i &:= [y_i^1, \ldots, y_i^p]^T \in \mathbb{R}^p, \\ & i \in \{1, 2, \ldots, N\}. \end{aligned} \tag{1}$$

Splitting of nodes is shown in Fig. 1. At i^{th} node, CART splits the data set into 2 subsets. The left branch R_L contains the data corresponding to $x^i \leq t_i$ and the right branch R_R corresponding to $x^i > t_i$. The optimal split at each node is then determined by minimizing the sum of mean square error in both the branches:

$$(x^k, t_k) = \arg\min \sum_{\{i|x_i \in R_L\}} (y_i - \bar{y}_L)^2 + \sum_{\{i|x_i \in R_R\}} (y_i - \bar{y}_R)^2 \tag{2}$$

where $y_i \in \mathbb{R}$ and $\bar{y}_L$ 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. In order 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

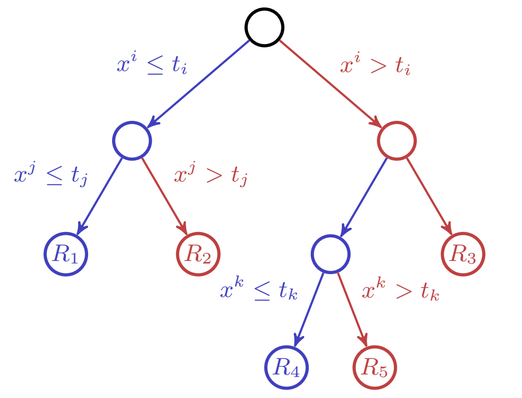


Figure 1: Binary regression tree: first split occurs with input x^i at t_i, second split with input x^j at t_j and so on, resulting in 5 regions in this case $R_1, \dots, R_5$.

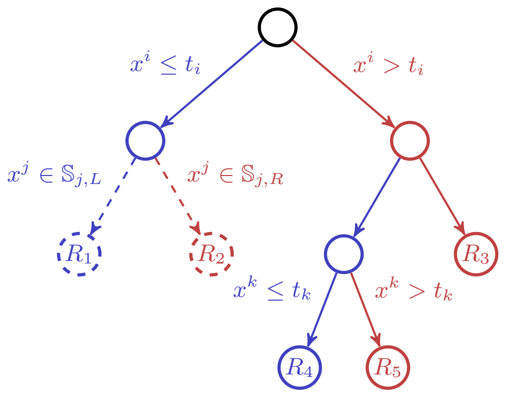


Figure 2: Binary regression tree: first split occurs with continuous input x^i at t_i, second split with categorical input x^j at t_j^r such that $\mathbb{S}_{j,L} = \{t_j^1, \dots, t_j^r\}$ and $\mathbb{S}_{j,R} = \{t_j^{r+1}, \dots, t_j^q\}$.

p outputs. Appropriately, we modify (2) as follows:

$$(x^k, t_k) = \arg\min \sum_{\{i|x_i \in R_L\}} ||y_i - \bar{y}_L||_l^2 + \sum_{\{i|x_i \in R_R\}} ||y_i - \bar{y}_R||_l^2. \tag{3}$$

In this case, $\bar{y}_L, \bar{y}_R \in \mathbb{R}^p$ and represent the mean of $\forall y_i \in R_L$ 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 in the outputs or l^2 norm which will minimize the sum of squares across all the outputs. Further, we can introduce weights matrix $Q \in \mathbb{R}^{p \times p}$ as other tuning parameter and choose the optimization objective similar to the cost function in MPC:

$$(x^k, t_k) = \arg\min \sum_{\{i|x_i \in R_L\}} (y_i - \bar{y}_L)^T Q (y_i - \bar{y}_L) + \sum_{\{i|x_i \in R_R\}} (y_i - \bar{y}_R)^T Q (y_i - \bar{y}_R). \tag{4}$$

Both (3) and (4) can be solved numerically by discretizing the search space of t_k between $max(x^k)$ and $min(x^k)$ calculated across N data points. Finer the resolution res, better the accuracy of splits. The terminating condition for growing the tree remains unchanged in (3) and (4).

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 $\mathbb{S}_c = \{t_c^1, \dots, t_c^q\}$. Number of ways to partition $\mathbb{S}_c$ 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 res. Hence, when q is large, 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 simliar to the one described in [24] for single-output system. We first find out all y_is corresponding to each element in $\mathbb{S}_c$ and then order the set $\mathbb{S}_c$ according to an increasing mean:

$$\bar{y}_q = \frac{\sum_{\{i|x^c = t_c^q\}} ||y_i||_l}{N_q}, \tag{5}$$

where N_q is the number of data points for which $x^c = t_q$. Once $\mathbb{S}_c = \{t_c^1, \dots, t_c^q\}$ is ordered such that $\bar{y}_1 < \dots < \bar{y}_q$, we split the variable as if it is a continuous variable using (3) or (4) depending upon the chosen type of formulation. If the cost is minimized for $x^c \le t_c^r$, then the left branch contains $x^c \in \mathbb{S}_{c,L} = \{t_c^1, \dots, t_c^r\}$ and the right branch contains $x^c \in \mathbb{S}_{c,R} = \{t_c^{r+1}, \dots, t_c^q\}$. A tree with a mix of continuous and categorical variables is shown in Fig. 2.

In summary, when the data set contains both types of variables- 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

$$\begin{pmatrix} y^1 \\ \vdots \\ y^p \end{pmatrix} = f\left(x^1, \dots, x^n\right). \tag{6}$$

In the context of a building model, this approach is validated in Sec. 4.2.

2.2 Interpretability of regression trees

Regression trees based approaches are our choice of data-driven models since they highly interpretable, by design. Interpretability is a fundamental desirable quality in any predictive model. Complex predictive models like neural-networks , support vector regression etc. 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. 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 in 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

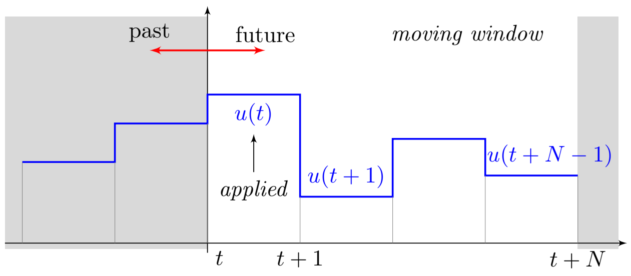


Figure 3: Finite-horizon moving window of MPC: at time t, the MPC optimization problem is solved for a finite length window of N steps and the first control input $u(t)$ is applied; the window then recedes one step forward and the process is repeated at time $t+1$.

machine learning algorithms more interpretable is an active area of research [12], one that is essential for incorporating human centric models in cyber-physical energy systems.

3. DATA PREDICTIVE CONTROL

The data-driven algorithm described so far uses the forecast of features to obtain building power consumption predictions. In this section, we describe a control algorithm which utilizes the multi-variate output capability of our regression tree model to implement receding horizon control. This is one of our primary contributions.

Recall that the objective of learning a regression tree is to learn a model f for predicting the response $\mathbb{Y}$ with the values of the predictor variables or features $\mathbb{X}^1, \ldots, \mathbb{X}^n$; i.e. $Y = f(\mathbb{X}^1, \ldots, \mathbb{X}^n)$. Given a forecast of the features $\hat{X}^1, \ldots, \hat{X}^n$ we can predict the response $\hat{Y}$. Now consider the case where a subset, $\mathbb{X}_c \subset \mathbb{X}$ of the set of features/variables $\mathbb{X}$'s are manipulated variables i.e. we can change their values in order to drive the response $\mathbb{Y}$ towards a certain value. In the case of buildings, the set of variables can be separated into disturbances (or non-manipulated) variables like outside air temperature, humidity, wind etc. while the controllable (or manipulated) variables would be the temperature and lighting set-points within the building. Our goal is to modify the regression trees and make them suitable for synthesizing the optimal values of the control variables in real-time. In our previous work, we proposed an algorithm for model based control with regression trees (mbCRT) [3]. This algorithm utilizes a *separation of variables* principle to allow for a control optimization in the leaves of the tree. Although mbCRT enables control with regression trees based models, it suffers from two significant limitations:

1. mbCRT is based on uni-variate output regression tree models and is unable to make multi-variate predictions.

2. The mbCRT algorithm is a 'one-step look-ahead' algorithm. It can only account for an unexpected disturbance only one time-step before it occurs, thus making it sub-par as compared to receding horizon control algorithms.

In mbCRT, given building data $(\mathbb{X}, \mathbb{Y})$ in the form of (1), we can separate the controllable $\mathbb{X}_c$ (or manipulated variables) and uncontrollable variables $\mathbb{X}_d$ (or non-manipulated

Algorithm 1 Data Predictive Control with Regression Trees

```
    DESIGN TIME
    procedure MODEL TRAINING USING SEPARATION OF VARIABLES
        Set $\mathbb{X}_c \leftarrow$ manipulated features
        Set $\mathbb{X}_d \leftarrow$ non-manipulated features
        Build a predictive tree $\mathfrak{T}$ with $(\mathbb{Y}, \mathbb{X}_d)$ using (6)
        for all regions $R_i$ at the leaves of $\mathfrak{T}$ do
            Choose a function $h$ for the problem
            Fit linear model $h\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c$
        end for
    end procedure
    RUN TIME
    procedure PREDICTIVE CONTROL
        while $t < t_{\text{stop}}$ do
            Determine the leaf and region $R_i(t)$ using $\mathbb{X}_d(t)$
            Obtain the linear model at $R_i(t)$
            Choose a cost function $g$ for the problem
            Solve optimization in (12) or (13) to determine optimal control action $[\mathbb{X}^*_c(t), \ldots, \mathbb{X}^*_c(t+p)]^T$
            Apply the first input $\mathbb{X}^*_c(t)$
        end while
    end procedure
```

variables or disturbances) in the features such that $\mathbb{X}_c \cup \mathbb{X}_d \equiv \mathbb{X}$. This is called separation of variables. Applying this separation of variables, the regression tree is built only on the non-manipulated variables or disturbances $(\mathbb{X}_d, \mathbb{Y})$. We obtain a model in the following form:

$$\mathbb{Y} = f\left(\mathbb{X}^1_d, \ldots, \mathbb{X}^n_d\right). \tag{7}$$

In the leaf R_i of the tree, we fit a parametric model (linear regression) which is a function only of the controllable/manipulated variables:

$$\mathbb{Y}_{R_i} = \beta_{0,i} + \beta_i^T \mathbb{X}_c. \tag{8}$$

In this manner, we train a regression tree using only $\mathbb{X}_d$, and then in each leaf we train a linear model which is a function only of $\mathbb{X}_c$. To solve the control problem when only $\mathbb{X}_d$ is known, we navigate to an appropriate leaf R_i and determine $\mathbb{X}_c$ from the following optimization problem:

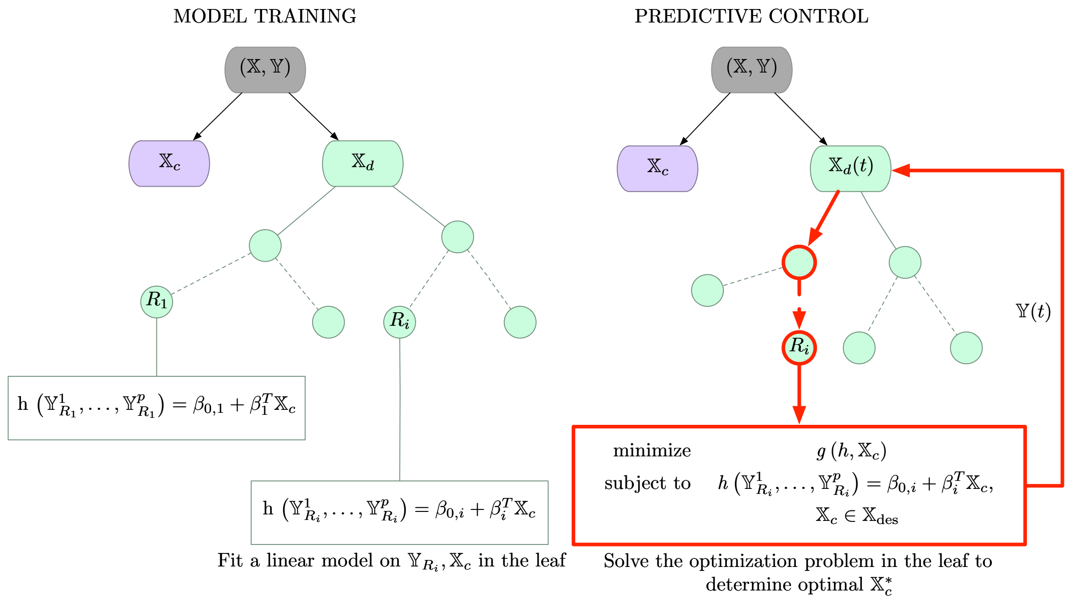
$$\begin{aligned} \text{minimize} \quad & g\left(\mathbb{Y}_{R_i}, \mathbb{X}_c\right) \\ \text{subject to} \quad & \mathbb{Y}_{R_i} = \beta_{0,i} + \beta_i^T \mathbb{X}_c, \\ & \mathbb{X}_c \in \mathbb{X}_{\text{des}}, \end{aligned} \tag{9}$$

where g is some function of the response variable and/or control variables we wish to minimize.

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 are computed, the first input is applied, then at the next step the optimization is solved again as shown in Fig. 3.

In the following section, we extend the mbCRT algorithm such that we can implement receding horizon control and address the limitations in mbCRT. This algorithm is called data predictive control with regression trees (DPCRT).

3.1 DPC with Regression Trees



MODEL TRAINING
PREDICTIVE CONTROL
$(\mathbb{X}, \mathbb{Y})$
$\mathbb{X}_c$
$\mathbb{X}_d$
R_1
R_i
$\mathrm{h}\left(\mathbb{Y}^1_{R_1}, \ldots, \mathbb{Y}^p_{R_1}\right) = \beta_{0,1} + \beta_1^T \mathbb{X}_c$
$\mathrm{h}\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c$
Fit a linear model on $\mathbb{Y}_{R_i}, \mathbb{X}_c$ in the leaf
$\mathbb{X}_d(t)$
$\mathbb{Y}(t)$
minimize $g(h, \mathbb{X}_c)$
subject to $h\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c,$
$\mathbb{X}_c \in \mathbb{X}_{\text{des}}$
Solve the optimization problem in the leaf to determine optimal $\mathbb{X}_c^*$

Figure 4: Data Predictive Control with Regression Trees with Model Training Process (L) and Receding Horizon Control (R). During control step, the optimal control action $[\mathbb{X}_c^*(t), \ldots, \mathbb{X}_c^*(t+p)]^T$ is determined. The first input $\mathbb{X}_c^*(t)$ is applied to the system. The resulting output $\mathbb{Y}(t)$ which is a feature for the next time step is fed back to determine to determine $R_i(t+1)$.

The central idea behind DPCRT is to build a tree model which can also predict future states of the system. Thus, while training a regression tree with multiple response variables, we still use separation of variables as in mbCRT, the difference lies in the number of output variables in each leaf. Therefore, (7) is appropriately modified as

$$\begin{pmatrix} \mathbb{Y}^1 \\ \vdots \\ \mathbb{Y}^p \end{pmatrix} = f\left(\mathbb{X}_d^1, \ldots, \mathbb{X}_d^n\right). \tag{10}$$

In each leaf of the tree, we now fit a linear model on a function $h : \mathbb{R}^p \to \mathbb{R}$ of all the response variables such that

$$h\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c. \tag{11}$$

A simple example of h is an affine function which we will use for our case study in Sec. 4. Once the model is trained, we solve the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & g(h, \mathbb{X}_c) \\ \text{subject to} \quad & h\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c, \\ & \mathbb{X}_c \in \mathbb{X}_{\text{des}}. \end{aligned} \tag{12}$$

We solve this optimization in the same manner as finite receding horizon control, and $\mathbb{X}_c$ includes all the control variables for the chosen horizon, i.e. $\mathbb{X}_c := [\mathbb{X}_c(t), \ldots, \mathbb{X}_c(t+p)]^T$. We choose the first optimal control input $\mathbb{X}_c^*(t)$ and proceed to the next time step.

If the number of control variables is large, the optimization problem (12) 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 which will ease the selection of a lower *minLeaf*:

$$\begin{aligned} \text{minimize} \quad & g\left(h^1, \ldots, h^{\text{nc}}, \mathbb{X}_c\right) \\ \text{subject to} \quad & h^1\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \beta_{0,i} + \beta_i^T \mathbb{X}_c^1, \\ & \vdots \\ & h^{\text{nc}}\left(\mathbb{Y}^1_{R_i}, \ldots, \mathbb{Y}^p_{R_i}\right) = \gamma_{0,i} + \gamma_i^T \mathbb{X}_c^{\text{nc}}, \\ & \mathbb{X}_c \in \mathbb{X}_{\text{des}}. \end{aligned} \tag{13}$$

Here, h^j is a linear model which depends only in the control variable $\mathbb{X}_c^j$. Note that $\mathbb{X}_c^j$ can still be a vector when horizon length is greater than 1. With suitable choice of g and h, the problems (12) and (13) can be formed as convex optimization problems.

Our algorithm for DPC with regression trees in summarized in Algo. 1 and a schematic is shown in Fig. 4. During training process, the tree is trained only on uncontrollable variables with linear models in the leaves which are a function only of controllable variables. During control step, at time t, the uncontrollable features $\mathbb{X}_d(t)$ are known and thus the leaf $R_i(t)$ is known. The optimization problem in $R_i(t)$ is solved to determine the control action $[\mathbb{X}_c^*(t), \ldots, \mathbb{X}_c^*(t+p)]^T$. The first input $\mathbb{X}_c^*(t)$ is applied to the system. The resulting output $\mathbb{Y}(t)$ which is a feature for the next time step is fed back to determine to determine $R_i(t+1)$.

Again, in the context of a building model, we show the efficacy of DPCRT in Sec. 4.3.

4. CASE STUDY

In this section, we present a comprehensive case study to show how DPCRT can be used for receding horizon control for peak power reduction in buildings.

4.1 Building test-bed

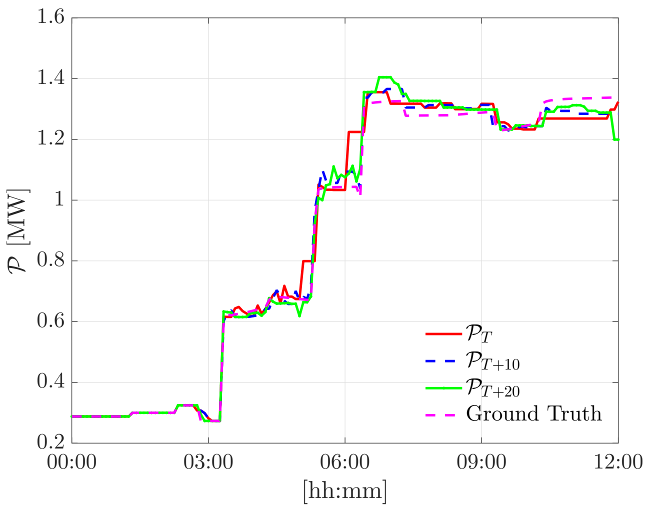


Figure 5: Building power consumption at time T predicted at time T is denoted by $\mathcal{P}_T$, predicted 10 steps ahead at time $T-10$ is denoted by $\mathcal{P}_{T+10}$, and predicted 20 steps ahead at time $T-20$ is denoted by $\mathcal{P}_{T+20}$.

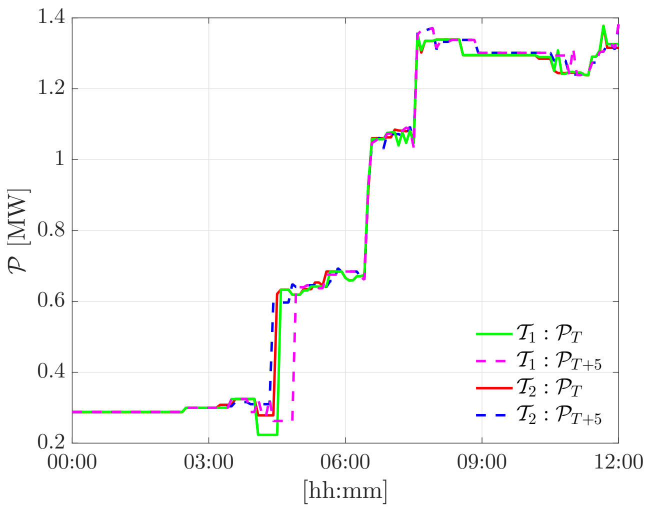


Figure 6: A comparison of power consumption of the building for two different regression trees. $\mathcal{T}_1$ is trained on all the features while $\mathcal{T}_2$ is trained only on non-manipulated features using separation of variables.

We use the DoE Commercial Reference Building (DoE CRB) simulated in EnergyPlus [9] as the virtual test-bed building. This is a large 12 story office building consisting of 73 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 multi-variate output regression trees are built using a training data-set from July 2012. The training data-set contains the following types of features/variables:

1. **Weather Data $\mathcal{W}$:** This includes 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 $\mathcal{S}$:** We create proxy variables which correlate with repeated patterns of electricity consumption e.g., due to occupancy or equipment schedules. Day of Week is a categorical predictor which takes values from 1-7 depending on the day of the week. This variable can capture any power consumption patterns which 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 $\mathcal{B}$:** The state of the building includes (i) Chilled Water Supply Temperature, (ii) Hot Water Supply Temperature, (iii) Zone Air Temperature, (iv) Supply Air Temperature, and (v) Lighting levels.

4. **Power Consumption $\mathcal{P}$:** This is the response variable, in addition to zone temperatures. We also considered autoregressive terms of power consumption in the input. An auto-regressive tree model is a regular regression tree except that the lagged values of the response variable are also predictor variables for the regression tree.

If the response variables of the regression tree are chosen as the power consumption of the building over a finite horizon, at time T, the outputs are $\mathcal{P}_T, \mathcal{P}_{T+1} \dots \mathcal{P}_{T+p}$. Thus, the tree model $\mathfrak{T}$ is written as

$$\begin{pmatrix} \mathcal{P}_T \\ \mathcal{P}_{T+1} \\ \vdots \\ \mathcal{P}_{T+p} \end{pmatrix} = f\left(\mathcal{W}_T, \dots, \mathcal{W}_{T+p}, \mathcal{S}, \mathcal{B}, \mathcal{P}_{T-1}, \dots, \mathcal{P}_{T-\delta}\right). \tag{14}$$

4.2 Model Validation: Multi-variate outputs

For a tree with order of autoregression $\delta = 6$ and a prediction horizon $p = 20$ and Q in (4) as Identity matrix, the results on the test dataset are shown in Fig. 5. The test set shows a day from July 2013, which the model has never seen before. It shows the building power consumption predicted at any time $\mathcal{P}_T$ as compared to the actual power consumption of the building. Since we can predict the power for multiple steps (horizon) at a time, we compare it to $\mathcal{P}_{T+10}$ calculated 10 steps before T, i.e. $T-10$, $\mathcal{P}_{T+20}$ calculated 20 steps before T, i.e. $T-20$, and the ground truth from the test dataset. It can be seen that even with a relatively long horizon, the multi-variate output tree model captures the rapid changes in the response variable (power consumption) very accurately.

4.3 Model Validation: DPCRT

The performance of DPCRT depends on two key assumptions: (a) first, is that the separation of variables doesn't

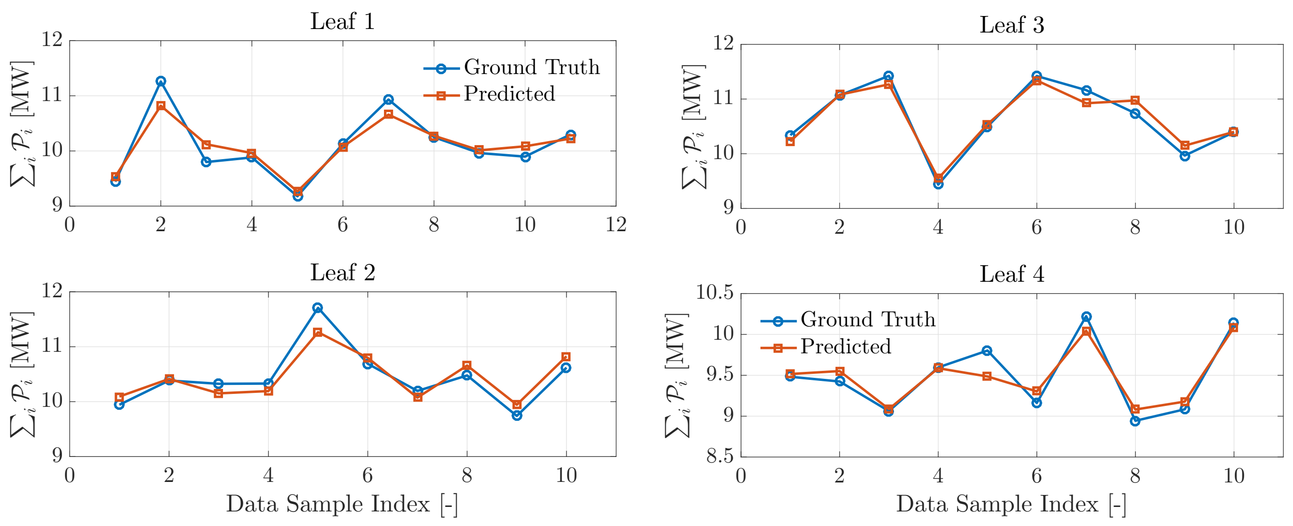


Figure 7: Model validation for linear regression at the leaves of the tree. The predicted and the actual power consumption are very close.

Table 1: NRMSE for regression trees with ($\mathcal{T}_1$) and without ($\mathcal{T}_2$) controllable features.

	$\mathcal{P}_T$	$\mathcal{P}_{T+1}$	$\mathcal{P}_{T+2}$	$\mathcal{P}_{T+3}$	$\mathcal{P}_{T+4}$	$\mathcal{P}_{T+5}$
$\mathcal{T}_1$	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

introduce significant errors while training the tree, and (b) second, that 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.

4.3.1 *Separation of Variables*

We train two kinds of regression trees: (1) a tree $\mathcal{T}_1$ that has all the features as described earlier, and (2) a tree $\mathcal{T}_2$ that was learned from non-manipulated variables only with a linear model on the control/manipulated variables at the leaves. The predicted power consumption of the building at time T and $T+5$ (see (14)), i.e. $\mathcal{P}_T$ and $\mathcal{P}_{T+5}$, respectively, for both trees is shown in Fig. 6. The normalized root mean square error (NRMSE) for these 2 outputs on the test dataset is shown in Tab. 1. We notice a small loss in model accuracy with $\mathcal{T}_2$ due to the separation of variables. This is the opportunity cost for integrating control synthesis with the tree, since otherwise the control features would have been a part of the splitting critea rather than a linear model in the leaves of the tree. In the case of $\mathcal{T}_2$ we exploit the tree structure to reach the right leaf, the actual output is determined by fitting a linear model which is a function of the controlled variables.

4.3.2 *Linear Approximation at Leaves*

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. Recall (11) which is now expressed as

$$\mathbb{Y}^1_{R_i} + \cdots + \mathbb{Y}^p_{R_i} = \beta_{0,i} + \beta_i^T \mathbb{X}_c. \tag{15}$$

Equivalently,in terms on power consumption, at each leaf we

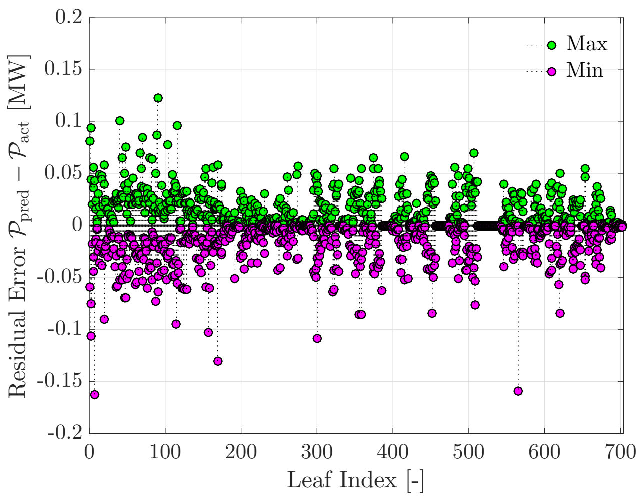


Figure 8: The tree has 703 leaves. For each leaf, a maximum and a minimum error in prediction of average power consumption over the control horizon $\mathcal{P}_{\text{pred}} - \mathcal{P}_{\text{act}}$ is calculated from the data points that end up in that leaf.

fit a linear model:

$$\mathcal{P}_T + \cdots + \mathcal{P}_{T+p} = \beta_0 + \beta^T \mathbb{X}_c. \tag{16}$$

For 4 randomly selected leaves, the fit of the linear model against the actual power consumption is shown in Fig. 7. The error observed in the predicted and actual power consumption for all leaves is depicted in Fig. 8. it can be seen that for a small number of samples in the leaves of the tree the linear model assumption is valid.

4.4 DPC for Peak Power Reduction

We will now compare 2 different control algorithms: mbCRT [2] and DPCRT for peak power curtailment. Recall the model of the building in Sec. 4.2. As described before, the

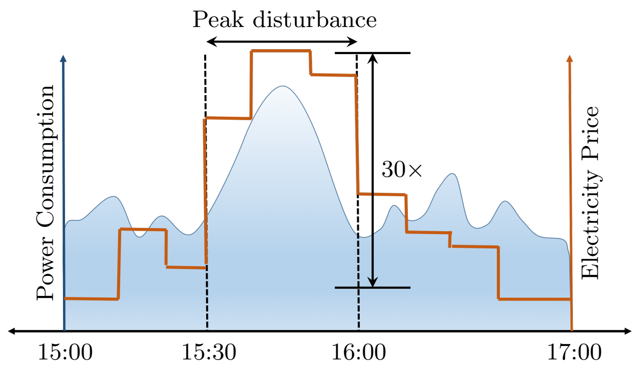


Figure 9: The test scenario used in the simulations has a 1.5× peak disturbance between 15:30 and 16:00 hrs. The electricity consumption for this peak is typically heavily penalized and is of the order of 30×.

data samples consist of 4 types of features, namely weather data, schedule data, building data and autoregressive terms of building power consumption. Of all the features, we use 3 features from the Building Data as controllable variables. These are

1. Zone Temperature Cooling Set Point $\mathcal{C}$ [°C],
2. Chilled Water Temperature Set Point $\mathcal{H}$ [°C], and
3. Lighting Set Point $\mathcal{L}$ [-].

To predict the power consumption at a future time instance, we first need to predict the zone temperature at that time because they are used as features in the power tree. To accomplish this, two types of trees are built.

1. Power tree which predicts the power consumption of the building $\mathcal{P}$ using all the features except the controllable features, and
2. Temperature trees which predict the mean zone temperature of each zone. The tree for the i^{th} zone predicts the temperature $\mathcal{T}_i$ using weather data, schedule data and autoregessive terms of the temperature of that zone.

Both the trees use a linear model at the leaves which is a function solely of the controllable variables.

In the case of mbCRT, we use the formulation (9), which uses single output regression trees. The objective function g consists of 2 terms: the cost for the power consumption and the cost for the thermal comfort. The choice of the factor λ can be subjective and is used to trade-off between the two. The optimal control input $[\mathcal{C}^*, \mathcal{H}^*, \mathcal{L}^*]^T$ is determined by solving the following optimization problem at the leaf.

$$\begin{aligned}
\text{minimize} \quad & \mathcal{P} + \lambda \sum_i |\mathcal{T}_i - \mathcal{T}_{\text{ref}}| \\
\text{subject to} \quad & \mathcal{T}_i = \alpha_{0,i} + \alpha_{1,i}\mathcal{C} + \alpha_{2,i}\mathcal{H} + \alpha_{3,i}\mathcal{L}, \\
& \mathcal{P} = \beta_0 + \beta_1\mathcal{C} + \beta_2\mathcal{H} + \beta_3\mathcal{L} \\
& \mathcal{C}_{\min} \leq \mathcal{C} \leq \mathcal{C}_{\max}, \\
& \mathcal{H}_{\min} \leq \mathcal{H} \leq \mathcal{H}_{\max}, \\
& \mathcal{L}_{\min} \leq \mathcal{L} \leq \mathcal{L}_{\max}.
\end{aligned} \tag{17}$$

For DPC with Regression Trees (DPCRT), we use the formulation (13). Now the objective function covers the cost for the complete horizon. So the governing optimization problem becomes

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{p} \frac{\mathcal{P}_{\text{j}}^{\mathcal{C}} + \mathcal{P}_{\text{j}}^{\mathcal{H}} + \mathcal{P}_{\text{j}}^{\mathcal{L}}}{3} + \lambda \sum_{j=1}^{p} \sum_i \left| \frac{\mathcal{T}_{\text{ij}}^{\mathcal{C}} + \mathcal{T}_{\text{ij}}^{\mathcal{H}} + \mathcal{T}_{\text{ij}}^{\mathcal{L}}}{3} - \mathcal{T}_{\text{ref}} \right| \\
\text{subject to} \quad & \mathcal{T}_{\text{ij}}^{\mathcal{C}} = \alpha_{0,\text{ij}}^{\mathcal{C}} + \alpha_{1,\text{ij}}^{\mathcal{C}}\mathcal{C}_1 + \cdots + \alpha_{\text{p,ij}}^{\mathcal{C}}\mathcal{C}_p, \\
& \mathcal{T}_{\text{ij}}^{\mathcal{H}} = \alpha_{0,\text{ij}}^{\mathcal{H}} + \alpha_{1,\text{ij}}^{\mathcal{H}}\mathcal{H}_1 + \cdots + \alpha_{\text{p,ij}}^{\mathcal{H}}\mathcal{H}_p, \\
& \mathcal{T}_{\text{ij}}^{\mathcal{L}} = \alpha_{0,\text{ij}}^{\mathcal{L}} + \alpha_{1,\text{ij}}^{\mathcal{L}}\mathcal{L}_1 + \cdots + \alpha_{\text{p,ij}}^{\mathcal{L}}\mathcal{L}_p, \\
& \sum_{j=1}^{p} \mathcal{P}_{\text{j}}^{\mathcal{C}} = \beta_0^{\mathcal{C}} + \beta_1^{\mathcal{C}}\mathcal{C}_1 + \cdots + \beta_{\text{p}}^{\mathcal{C}}\mathcal{C}_p, \\
& \sum_{j=1}^{p} \mathcal{P}_{\text{j}}^{\mathcal{H}} = \beta_0^{\mathcal{H}} + \beta_1^{\mathcal{H}}\mathcal{H}_1 + \cdots + \beta_{\text{p}}^{\mathcal{H}}\mathcal{H}_p, \\
& \sum_{j=1}^{p} \mathcal{P}_{\text{j}}^{\mathcal{L}} = \beta_0^{\mathcal{L}} + \beta_1^{\mathcal{L}}\mathcal{L}_1 + \cdots + \beta_{\text{p}}^{\mathcal{L}}\mathcal{L}_p, \\
& \mathcal{C}_{\min} \leq \mathcal{C}_j \leq \mathcal{C}_{\max}, \\
& \mathcal{H}_{\min} \leq \mathcal{H}_j \leq \mathcal{H}_{\max}, \\
& \mathcal{L}_{\min} \leq \mathcal{L}_j \leq \mathcal{L}_{\max}, \\
& j = \{1, \ldots, p\}.
\end{aligned} \tag{18}$$

Here, the optimization variables are the control inputs for the complete horizon, i.e. $[\mathcal{C}_1, \ldots, \mathcal{C}_p]^T$, $[\mathcal{H}_1, \ldots, \mathcal{H}_p]^T$ and $[\mathcal{L}_1, \ldots, \mathcal{L}_p]^T$.

We now compare the solutions from the two problems (17) and (18) when simulated in closed-loop with the EnergyPlus mode of the building. The primary difference between mbCRT and DPCRT is that mbCRT is only a single step look-ahead control algorithm while DPCRT is a receding horizon control algorithm. In order to compare the performance of DPCRT we consider a scenario in which there is a significant disturbance which is only anticipated 30 mins in advance and leads to a sudden increase in zone temperatures in the building. This maybe akin to a sunned spike in occupancy or equipment being witched ON at a brief notice. Under this scenario, it is important to react to the disturbance in a predictive manner in order to minimize the peak power consumption. This scenario is shown in Fig. 9. Between 15:30 and 16:00 hrs, an enormous spike in the power consumption (1.5×) is expected because of a scheduled operation.

The control strategies are tested over a 2 hour duration between 15:00 hrs and 17:00 hrs. Fig. 10 shows 3 control strategies: mbCRT, DPCRT and a Naive load reduction strategy. The naive strategy is equivalent to not responding to the disturbance at all. It maintains the desired zone temeprature set point $\mathcal{C}$, chiller water temperature set point $\mathcal{H}$ and lighting level $\mathcal{L}$ throughout the test period. In Fig. 11, 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 hrs, DPCRT strategy is same as the greedy one. At 15:05 hrs, the downstream disturbance is visible to DPCRT algorithm and it starts to pre-cool the building by decreasing both cooling

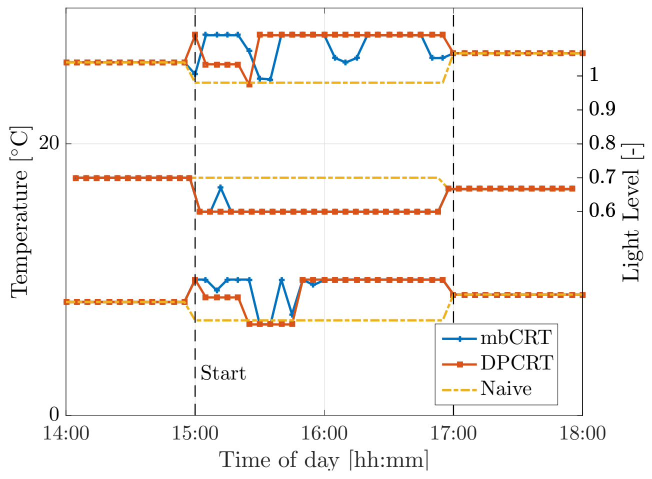


Figure 10: A comparison of different control strategies obtained from mbCRT and DPCRT along with the greedy strategy.

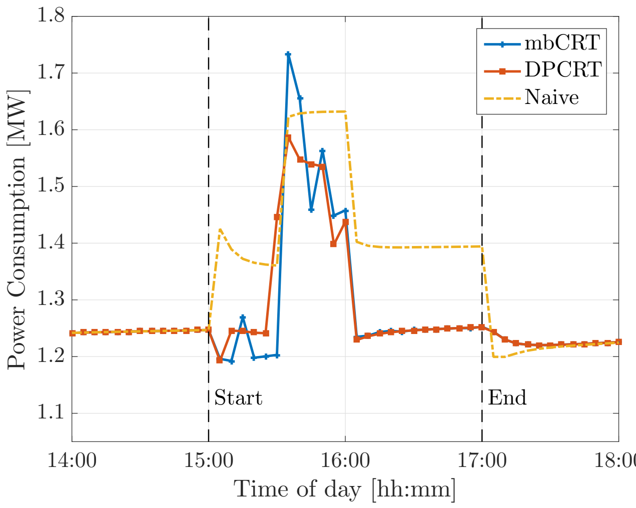


Figure 11: A comparison of building power consumption obtained from mbCRT and DPCRT along with the greedy strategy.

and chiller water set points. At 15:25 hrs, the $\mathcal{C}$ and the $\mathcal{H}$ are reduced to mimimum 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 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 that $\mathcal{C}$, $\mathcal{H}$ and $\mathcal{L}$ 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 Fig. 11. The DPCRT algorithm gradually increases the power consumption because it can see the disturbance before it actually reaches 15:30 hrs, 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 degrees while both mbCRT and the naive strategy have large deviations from the desired temperature.

Table 2: Quantitative comparison for Energy Consumption, Peak Power and % Reduction in Peak Power of DPCRT compared to Naive approach and mbCRT.

	Energy [kWh]	Peak Power [MW]	Peak Reduction
Naive	5358	1.63	3.1%
mbCRT	5097	1.73	**8.6%**
DPCRT	5102	1.58	-

The quantitative comparison is presented in Tab. 2. Between 15:00 and 17:00 hrs, DPCRT and mbCRT result in similar energy usage, 5102 kWh and 5097 kWh, respectively, both outperforming the Naive strategy which incurs 5358 kWh. Peak power in the case of DPCRT is 1.58 MW which is lower than both mbCRT (1.73 MW) and the Naive strategy (1.63 MW), although Naive outperforms mbCRT. The peak power with DPCRT is 8.6 % less than mbCRT and 3.1 % less than the Naive. While both DPCRT and mbCRT account for thermal comfort, DPCRT deviates less from the desired temperature. The Naive strategy does not trade off on thermal comfort. Thus, DPCRT outperforms mbCRT both in terms of a reduced peak power consumption and better thermal comfort.

5. RELATED WORK

There exist several different approaches to balance the power consumption in buildings and avoid peaks, e.g. by load shifting and load shedding [16, 14]. However, they operate on coarse grained time scales and do not guarantee any thermal comfort. Another popular approach to energy efficient control for commercial buildings and data centers is model predictive control [18, 20]. These usually assume that the model of the system is either perfectly known or found in literature, whereas the task is much more complicated and time consuming in case of a real building and sometimes, it can be even more complex and involved than the controller design itself. After several years of work on using first principles based models for demand response, multiple authors [26, 27] 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.

In data-driven optimal control literature, the models are trained on optimal solutions obtained from MPC. The resulting models can then be used for explicit MPC, as in [4]. This approach has been applied to problems of stabilization [6] and freeway traffic systems using regression trees [21]. Another class of methods solve nonlinear optimization directly on the trained models to do receding horizon control. This has been done for wind turbine control using evolutionary optimization algorithm [17], and for building control using branch and bound search algorithm [10].

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.

6. CONCLUSION

Data predictive control with regression trees (DPCRT) is an algorithm for implementing receding horizon control with data-driven (regression trees) based models. DPCRT uses multi-variate output regression trees where the different outputs represent the future states of the system. By separating the controllable and uncontrollable features during training, and fitting a linear model on just the controllable features, the optimization is reduced to a simple convex program. This enables the use of receding horizon control synthesis for problems of peak power reduction in buildings which otherwise are dependent on first principles based model of the building. The performance of DPCRT is evaluated on a DoE commercial reference virtual test-bed. DPCRT results in a much lower energy consumption when compared to a Naive control strategy. DPCRT also leads to 8.6% decrease in the peak power reduction of the building as compared to mbCRT and 3.1% as compared to the Naive strategy. The receding horizon prediction ability of DPCRT results in a smooth control actions as we the disturbance occurs while mbCRT leads to a sudden control action and higher peak consumption. In the future, we will compare the results directly with MPC.

7. REFERENCES

[1] Energy price risk and the sustainability of demand side supply chains. *Applied Energy*, 123(0):327 – 334, 2014.

[2] M. Behl, A. Jain, and R. Mangharam. Data-driven modeling, control and tools for cyber-physical energy systems. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*, pages 1–10. IEEE, 2016.

[3] M. Behl, F. Smarra, and R. Mangharam. Dr-advisor: A data-driven demand response recommender system. *Applied Energy*, 170:30 – 46, 2016.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

[5] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[6] L. Cavagnari, L. Magni, and R. Scattolini. Neural network implementation of nonlinear receding-horizon control. *Neural computing & applications*, 8(1):86–92, 1999.

[7] M.-H. Construction. *Energy efficiency trends in residential and commercial buildings*. US Department of Energy, Energy Efficiency and Renewable Energy, 2010.

[8] D. B. Crawley, L. K. Lawrie, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33(4):319 – 331, 2001.

[9] M. Deru, K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M. Rosenberg, et al. Us department of energy commercial reference building models of the national building stock. 2011.

[10] P. Ferreira, A. Ruano, S. Silva, and E. Conceicao. Neural networks based predictive control for thermal comfort and energy savings in public buildings. *Energy and Buildings*, 55:238–251, 2012.

[11] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[12] C. Giraud-Carrier. Beyond predictive accuracy: what? *Proc. ECML Workshop on Upgrading Learning to Meta-Level: Model Selection and Data Transformation*, pages 78–85, 1998.

[13] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

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

[15] E. in Buildings and C. Programme. Ebc annex 53 total energy use in buildings: Analysis & evaluation methods, 2013.

[16] S. Kiliccote, M. Piette, and D. Hansen. Advanced controls and communications for demand response and energy efficiency in commercial buildings. In *Second Carnegie Mellon Conf. in Elec. Power Sys.*, 2006.

[17] A. Kusiak, Z. Song, and H. Zheng. Anticipatory control of wind turbines with data-driven predictive models. *Energy Conversion, IEEE Transactions on*, 24(3):766–774, 2009.

[18] Y. Ma, F. Borrelli, B. Hencey, B. Coffey, S. Bengea, and P. Haves. Model predictive control for the operation of building cooling systems. In *Proc. ACC'10*, pages 5106–5111, 2010.

[19] P. I. Michael J. Kormos. Pjm response to consumer reports on 2014 winter pricing. 2014.

[20] F. Oldewurtel, A. Ulbig, A. Parisio, G. Andersson, and M. Morari. Reducing peak electricity demand in building climate control using real-time pricing and model predictive control. In *Proc. IEEE CDC'10*, pages 1927–1932, 2010.

[21] A. N. Oleari, J. R. D. Frejo, E. F. Camacho, and A. Ferrara. A model predictive control scheme for freeway traffic systems based on the classification and regression trees methodology. In *Control Conference (ECC), 2015 European*, pages 3459–3464. IEEE, 2015.

[22] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[23] J. R, J. Sanyal, M. Bhandari, and S. Shrestha. Autotune e+ building energy models. *Proceedings of the 5th National SimBuild of IBPSA-USA*, 2012.

[24] B. D. Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[25] J. Struyf and S. Džeroski. Constraint based induction of multi-objective regression trees. In *International Workshop on Knowledge Discovery in Inductive Databases*, pages 222–233. Springer, 2005.

[26] D. Sturzenegger, D. Gyalistras, M. Morari, and R. Smith. Model predictive climate control of a swiss office building: Implementation, results, and cost-benefit analysis. *Control Systems Technology, IEEE Transactions on*, PP(99):1–1, 2015.

[27] E. Žáčeková, Z. Váňa, and J. Cigler. Towards the real-life implementation of mpc for an office building: Identification issues. *Applied Energy*, 135:53–62, 2014.