2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Learning Piecewise ARX models via Regression Trees with probabilistic guarantees

Alessandro D'Innocenzo and Francesco Smarra

Abstract— **Recent research literature shows that system identification techniques can be successfully combined with machine learning to improve the accuracy of the models obtained. In this context, the contribution of this work builds upon a research line that combines the Regression Trees method with AutoRegressive eXogenous identification to derive models of dynamical systems exploiting historical data. The main contribution of this paper is to formally relate such methodology with the scenario approach framework, thus providing probabilistic guarantees on the derived model. The proposed method is validated on a real experimental setup: first a comparison in terms of accuracy with the former method - which does not provide probabilistic guarantees - is provided, then the effectiveness of the derived probabilistic guarantees is validated on the testing dataset from our experimental setup.**

I. Introduction

Control of complex cyber-physical systems has received increasing attention in recent years [1], [2]. In this context, Model Predictive Control (MPC) is a well-known control strategy used to design optimal control actions to optimize a performance metric while guaranteeing a desired system behavior (i.e., reference tracking and constraints), and has been widely applied in past years to control a large variety of systems, as for example energy systems such as smart buildings, smart grids and power systems [3], [4], [5]. To provide the optimal control strategy, MPC leverages a mathematical model to predict the system's behavior over a finite time horizon. However, creating a physics-based model for a large-scale system as the ones mentioned above is often cost and time prohibitive [6]. Several works deal with this problem, and use both system identification from control theory and machine learning algorithms from computer science to construct models to be used for MPC. In what follows, we provide a survey of literature related to the topic of this paper and illustrate the novelty of our work with respect to the state of the art.

Previous work. To the best of the authors' knowledge, the use of regression trees with predictive control purposes has been addressed for the first time in [7], where data-driven models have been developed to enable one-step lookahead closed-loop control for the Demand-Response problem in buildings. This approach has been extended in [8], where the authors proposed a regression trees and random forests-based strategy that implements MPC over a horizon of arbitrary length. The basic idea was to bridge machine learning and control theory by adapting regression trees and random forests techniques to build control-oriented models. The aforementioned approaches make use of data-driven *static* models, namely where the input-output relation is represented by static affine functions instead of dynamical models. Such a modeling framework neglects the presence of the internal state evolution and loses the information of the past inputs applied to the system over the predictive horizon: in [9], [10] our approach has been extended, proving a novel methodology to build a switching affine dynamical model of a system using historical data by appropriately combining the regression trees and random forests classical algorithms with AutoRegressive eXogenous (ARX) identification. On the same research line, in [11] a methodology has been provided to build a Markov switching affine model that extracts the dynamics of the disturbance as a Markov chain exploiting historical data: the resulting model is a special case of Markov jump systems [12], and can be used to implement stochastic MPC via standard algorithms [13].

Paper contribution. The methodology illustrated above has been applied on real datasets of a variety of application domains [14], [15], [16], [17], [18], [19], [20] obtaining excellent performance compared to the state of the art with the added value that, while the static models derived in [7], [8] do not allow to formally define and characterize fundamental properties such as stability, stabilizability, controllability, etc., the models built in [10], [11] do. On the other hand, the analysis of such properties on the constructed models is not endowed with any kind of guarantees.

The main contribution of this paper is to provide probabilistic guarantees for models constructed with the methodology in [10] exploiting the well known *scenario approach* (see e.g., [21], [22], [23], [24], [25], [26] and references therein). More precisely, we provide a bound to the probability that testing samples (i.e. not belonging to the training dataset) do not satisfy the estimation accuracy obtained on the training dataset: to the best of the authors' knowledge, this is a novel contribution to the scientific literature.

As we show in the paper, the scenario approach cannot be applied directly to the methodology in [10]. To this aim we will modify the original algorithm in [10] and then formally derive probabilistic guarantees. We validate our approach on a real experimental setup consisting of a two-

This work was partially supported by: the Italian Government under Grants SICURA (CUP C19C20000520004) and CIPE 70/2017 (Centre EX-Emerge); the European Union under Grants DigInTraCE (GA 101091801) and Resilient Trust (GA 1011112282); the NRRP MUR program funded by the NextGenerationEU under Grants PNRR ICSC National Research Centre for High-Performance Computing, Big Data and Quantum Computing (CN00000013) and National Innovation Ecosystem (ECS00000041).

The authors, listed in alphabetical order, are with the Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, 67100 AQ, Italy. alessandro.dinnocenzo@univaq.it, francesco.smarra@univaq.it

story building, where each floor is composed of 5 zones, located inside the campus of the University of L'Aquila, Italy. We apply the proposed method to two different scenarios: room temperature and energy consumption prediction. We first compare in terms of accuracy our novel algorithm with that of [10], which does not provide probabilistic guarantees; then we statistically evaluate, on the testing dataset, the effectiveness of the theoretical guarantees derived in this paper. Our experiments show that the modifications introduced to our original algorithm suffer just a slight decrease of the prediction accuracy, and that the theoretical probabilistic guarantees are coherent with the results obtained on the testing dataset.

II. Scenario approach background

In this section, we provide the mathematical background on the scenario approach, which we will exploit to derive probabilistic guarantees on the technique proposed in this paper. For more details, the reader is referred to [26] and references therein.

Consider a probability space $(\Delta, \mathcal{D}, \mathcal{P})$ and a sample $(\delta^{(1)}, \dots, \delta^{(N)}) \in \Delta^N$ of N elements drawn independently from Δ according to the same probability measure $\mathcal{P}$. We call each observation $\delta^{(\cdot)}$ a *scenario*. Let us also consider a set $\Theta \subseteq \mathbb{R}^d$, called decision space, and define a function $\mathcal{A}_N : \Delta^N \to \Theta$ denoted as the *scenario decision* $\theta_N^* := \mathcal{A}_N(\delta^{(1)}, \dots, \delta^{(N)})$. Let the scenario decision $\mathcal{A}_N$ be a unique solution θ_N^*, possibly after applying a tie-break rule, of a constrained optimization program:

$$\min_{\theta \in \Theta} f(\theta) \text{ subject to } \theta \in \Theta_{\delta^{(i)}}, i = 1, \dots, N, \qquad (1)$$

where f, Θ and $\Theta_{\delta^{(\cdot)}}$ can be any function and constraint (i.e., convex or nonconvex). Let us define the *violation probability* of a decision $\theta \in \Theta$ as

$$V(\theta) = \mathcal{P}\{\delta \in \Delta : \theta \notin \Theta_\delta\}. \qquad (2)$$

For a given reliability parameter $\epsilon \in (0, 1)$, we say that $\theta \in \Theta$ is ϵ-feasible if $V(\theta) \leq \epsilon$. The violation of a scenario decision $V(\theta_N^*)$ is a random variable over Δ^N. The idea behind the scenario approach framework is to characterise the distribution of $V(\theta_N^*)$ and find a confidence bound $1 - \beta$ such that the relation $V(\theta) \leq \epsilon$ is satisfied.

Convex case: When the optimization program (1) is convex, as the deepest result it has been shown in [23] that $V(\theta_N^*)$ is dominated by a Beta distribution, i.e.

$$\mathcal{P}^N\{V(\theta_N^*) > \epsilon\} \leq \beta, \ \beta = \sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i}, \quad (3)$$

where we recall that d is the dimension of the optimization variable, and that the only assumption on the probability space is that each scenario is drawn independently.

Non-convex case: When the optimization program (1) is non-convex, a different approach is used in [26]. Assume that $\mathcal{A}_N(\delta^{(1)}, \dots, \delta^{(N)})$ is the (unique, possibly suboptimal) scenario decision given a sample $(\delta^{(1)}, \dots, \delta^{(N)})$: we define a support subsample $(\delta^{(i_1)}, \dots, \delta^{(i_k)}), k \leq N$, as a k-tuple of elements of $(\delta^{(1)}, \dots, \delta^{(N)})$ such that $\mathcal{A}_k(\delta^{(i_1)}, \dots, \delta^{(i_k)}) = \mathcal{A}_N(\delta^{(1)}, \dots, \delta^{(N)})$. Let a support subsample of cardinality k exist, then the following holds:

$$\mathcal{P}^N\{V(\theta_N^*) > \epsilon(k)\} \leq \beta$$
$$\epsilon(k) = \begin{cases} 1 & \text{k=N,} \\ 1 - \left(\frac{\beta}{N\binom{N}{k}}\right)^{(N-k)^{-1}} & \text{otherwise.} \end{cases} \qquad (4)$$

III. Learning ARX models via regression trees

In this section, we first provide a short description of the CART algorithm [27], and then we illustrate a technique, that we proposed for the first time in [9], [10], to identify from a dataset a model of a dynamical system that combines the CART algorithm and the ARX identification.

The CART algorithm: Due to space limitations, we only briefly recall the partitioning algorithm of CART, and refer the reader to [27] for more details. In a supervised learning framework, we consider an input dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ and an output dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ of N samples each, where $y_i \in \mathbb{R}$ and $x_i \in \mathbb{R}^n$. The final goal of CART is to identify a function $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}$ to estimate $\hat{y} = \mathcal{T}(x)$: to this aim, the data set is iteratively partitioned, according to a tree graph structure grown during the algorithm, by a set of hyper-rectangles $R_1, \dots, R_{|\mathcal{T}|}$ that induce a partition of the input space $\mathbb{R}^n$: to each hyper-rectangle corresponds one (and only one) leaf of the grown tree graph.

More in detail, without any loss of generality we restrict our attention to recursive binary partition of the CART algorithm [27]: starting with the whole dataset, which is associated to the root node of the tree, consider a split variable j over the n available and a split point s, and define the half-spaces $R_L(j, s) = \{x_i \mid x_{i,j} < s\}$ and $R_R(j, s) = \{x_i \mid x_{i,j} \geq s\}$, where $x_{i,j}$ is the j-th component of sample $x_i \in \mathbb{R}^n$. The CART algorithm solves the following optimization problem to find the optimal ι^* and s^*:

$$\min_{\iota, s} \left[\min_{c_L} \sum_{x_i \in R_L(\iota, s)} (y_i - c_L)^2 + \min_{c_R} \sum_{x_i \in R_R(\iota, s)} (y_i - c_R)^2 \right], \quad (5)$$

and for any choice of ι and s the inner minimization is solved by $c_L = \text{ave}(y_i \mid x_i \in R_L(\iota, s))$ and $c_R = \text{ave}(y_i \mid x_i \in R_R(\iota, s))$, where ave$(\cdot)$ is the arithmetic mean of the output samples. In other words, the optimal ι^* and s^* minimize the sum of the squared prediction errors of the left and right partitions induced by the split variable and split point. For each splitting variable, the determination of the split point s^* can be done very quickly and hence, by scanning through all the input components, the determination of the best pair (ι^*, s^*) is computationally tractable. Once the best split is found, the dataset is partitioned into the two resulting regions, and the splitting procedure is repeated starting from each of the nodes associated to the new defined regions. The process is repeated until a stopping criterion is applied, e.g. the tree size is a tuning parameter that should be chosen to avoid overfitting and variance phenomena. Then, $\hat{y}$ is estimated in each leaf ℓ_j using a constant c_{ℓ_j} given by the average of the samples in the partition.

In the rest of this work we will denote with $\mathcal{T}$ the regression tree, with ℓ_j the j^{th} leaf of $\mathcal{T}$, with $|\mathcal{T}|$ the number of leaves of $\mathcal{T}$ and with $|\ell_j|$ the number of samples in the leaf ℓ_j. Also, slightly abusing the notation, we will denote with $c_j = \text{ave}(y_i | x_i \in \ell_j)$ the prediction associated to leaf ℓ_j, with R_j the hyper-rectangular partition set associated to leaf ℓ_j and with $\hat{y} = \mathcal{T}(x)$ the regression tree prediction to a new sample x given by $\mathcal{T}(x) = \sum_{j=1}^{|\mathcal{T}|} c_j \cdot I\{x \in R_j\}$, where $I\{x \in R\}$ is the indicator function that is equal to 1 if $x \in R$, and 0 otherwise. The algorithmic complexity of growing a tree via the CART algorithm is $O(nN \ln N)$.

The CART algorithm combined with ARX identification: We briefly illustrate the approach proposed in [10], which combines the CART algorithm and ARX identification to build a model of a dynamical system starting from a dataset of trajectories. Consider a set of N sample trajectories $\{z_i(0), \dots, z_i(r), d_i(0), \dots, d_i(r-1)\}_{i=1}^N$, with a measurement variable $z_i(t) \in \mathbb{R}, t = 0, \dots, r$ and an exogenous variable $d_i(t) \in \mathbb{R}^\nu, t = 0, \dots, r-1$. All trajectories, each considered as a scenario $\delta^{(i)}$, are assumed to be collected as independent drawns from a same probability measure. Let us define an input dataset $\mathcal{X} = \{x_i\}_{i=1}^N$ with $x_i := (z_i(0), \dots, z_i(r-1), d_i(0), \dots, d_i(r-1)) \in \mathbb{R}^n, n = r(\nu+1)$, and an output dataset $\mathcal{Y} = \{y_i\}_{i=1}^N$ with $y_i := z_i(r) \in \mathbb{R}$. In [10] we derived Algorithm 0: we first applied the standard CART algorithm to a training dataset $\mathcal{X}$ and $\mathcal{Y}$, thus obtaining a regression tree $\mathcal{T}$ with leaves $\ell_1, \dots, \ell_{|\mathcal{T}|}$ and corresponding hyper-rectangles $R_1, \dots, R_{|\mathcal{T}|}$ that partition the input space; then, instead of using the constant predictor c_j in each leaf, we proposed to assign an affine predictor $m_j \in \mathbb{R}^n, m_{j,0} \in \mathbb{R}$ to each leaf, thus obtaining $\mathcal{T}(x)$ the prediction of the regression tree given by $\mathcal{T}(x) = \sum_{j=1}^{|\mathcal{T}|} (m_j^T x + m_{j,0}) \cdot I\{x \in R_j\}$.

Algorithm 0

1: **Input:** dataset $(\mathcal{X}, \mathcal{Y})$
2: Apply CART algorithm to $(\mathcal{X}, \mathcal{Y})$, obtaining as output a tree structure $\mathcal{T}$ with leaves $\ell_1, \dots, \ell_{\mathcal{T}}$
3: **for all** $j = 1, \dots, |\mathcal{T}|$ **do**
4: Compute in leaf ℓ_j an affine predictor $m_j \in \mathbb{R}^n, m_{j,0} \in \mathbb{R}$ by solving the quadratic program

$$\min_{m_j, m_{j,0}} \sum_{x_i \in \ell_j} (y_i - m_j^T x_i - m_{j,0})^2 \tag{6}$$

5: **end for**

IV. Learning ARX models via regression trees with probabilistic guarantees

Algorithm 0 has the goal of finding a (possibly suboptimal) solution of the following nonconvex optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\iota}, \boldsymbol{s}, \boldsymbol{m}} \sum_{i=1}^N (y_i - \mathcal{T}(x_i))^2 \\ &\text{s.t.} \mathcal{T}(x_i) = \sum_{j=1}^{|\mathcal{T}|} (m_j^T x_i + m_{j,0}) \cdot I\{x_i \in \ell_j(\boldsymbol{\iota}, \boldsymbol{s})\}, i = 1, \dots, N, \end{aligned} \tag{7}$$

where $\boldsymbol{\iota} := \{\iota_1, \dots, \iota_p\}$, $\boldsymbol{s} := \{s_1, \dots, s_p\}$ are sequences of length p of splitting variables and splitting points, $p \in \mathbb{N}$ is a natural number representing the number of splits performed by the CART algorithm (which depends on the chosen stopping criterion, see [27] for more details), and $\boldsymbol{m} := \{(m_j, m_{j,0})\}_{j=1}^{|\mathcal{T}|}$ are coefficients of the affine models associated to each leaf. Note that each leaf $\ell_j(\boldsymbol{\iota}, \boldsymbol{s})$ is a deterministic function of the optimization variables $\boldsymbol{\iota}, \boldsymbol{s}$. Let us define $(\boldsymbol{\iota}^*, \boldsymbol{s}^*, \boldsymbol{m}^*) := \mathcal{A}_N(x_1, \dots, x_N)$, where $\mathcal{A}_N$ is a solution of the optimization program (7) obtained applying Algorithm 0. Note that $(\boldsymbol{\iota}^*, \boldsymbol{s}^*, \boldsymbol{m}^*)$ is possibly suboptimal, and can be assumed unique since the CART algorithm is deterministic and provides a unique solution to any given dataset: we recall that its uniqueness is indeed a necessary condition to apply the scenario approach. However, the scenario approach cannot be directly applied to this problem neither in the convex case (since the problem is nonconvex) nor in the nonconvex case (since given any strict support subsample also the cost function changes, therefore the results in [26] cannot be applied to characterise the distribution of the violation probability), as illustrated in Section II.

As the main result of this paper we propose two modifications of Algorithm 0 so that the scenario approach can be applied, and we derive formal probabilistic guarantees for each of them. The first algorithm, formalised in Algorithm 1, is based on the following idea: we apply the CART algorithm to a candidate support subsample $X \cup Y$ set that initially only consists of the first sample of the dataset (line 3); at each iteration of the **while** cycle (line 4) we first apply CART to the dataset X, Y (line 5) and then we update X, Y adding the sample corresponding to the largest violation, if any, of the error bound of the constructed predictor (line 15); we stop the algorithm when all samples not belonging to X, Y satisfy the error bound derived only using samples in X, Y (line 16): when this happens we have found a support subsample.

Proposition 1: Let Algorithm 1 be applied to a dataset of inputs $\mathcal{X}$ and outputs $\mathcal{Y}$, then for any given $\beta \in (0, 1)$

$$\mathcal{P}^N\left\{\mathcal{P}\{x \in \mathbb{R}^n, y \in \mathbb{R} : (y - \mathcal{T}(x))^2 \geq h^*\} > \epsilon\right\} \leq \beta, \tag{9}$$

with

$$\epsilon = \begin{cases} 1 & k^* = N, \\ 1 - \left(\frac{\beta}{N\binom{N}{k^*}}\right)^{(N-k^*)^{-1}} & \text{otherwise.} \end{cases} \tag{10}$$

Proof: Using Algorithm 1 the derived prediction $\mathcal{T}(x)$ of the regression tree $\mathcal{T}$ is a possibly suboptimal solution, using input and output datasets X, Y with cardinality k^* each, of the following nonconvex optimization problem:

$$\begin{aligned} &\min_{h, \boldsymbol{\iota}, \boldsymbol{s}, \boldsymbol{m}} \; h \\ &\text{such that } (y_i - m_j^T x_i - m_{j,0})^2 \leq h, x_i \in \ell_j(\boldsymbol{\iota}, \boldsymbol{s}), \\ &\qquad j = 1, \dots, |\mathcal{T}|, \\ &\qquad x_i \in X, y_i \in Y \end{aligned} \tag{11}$$

with $h^*, \boldsymbol{\iota}^*, \boldsymbol{s}^*, \boldsymbol{m}^* = \mathcal{A}_{k^*}(X, Y)$ the obtained solution using Algorithm 1. Note that all steps of Algorithm 1, included

Algorithm 1

1: **Input:** datasets $\mathcal{X} = \{x_i\}_{i=1}^N, \mathcal{Y} = \{y_i\}_{i=1}^N$
2: **Output:** $\mathcal{T}, h^*, k^*$
3: Set $X := \{x_1\}, Y := \{y_1\}$
4: **while** $X \neq \mathcal{X}$ **do**
5: Apply CART algorithm to (X, Y), obtaining as output a tree structure $\mathcal{T}$ with leaves $\ell_1, \dots, \ell_{\mathcal{T}}$
6: **for all** $j = 1, \dots, |\mathcal{T}|$ **do**
7: Compute in leaf ℓ_j an error bound h_j^* and an affine predictor $m_j^* \in \mathbb{R}^n, m_{j,0}^* \in \mathbb{R}$ by solving the QP:

$$\min_{h_j, m_j, m_{j,0}} h_j$$
$$\text{subject to } (y_i - m_j^T x_i - m_{j,0})^2 \leq h_j, \forall x_i \in \ell_j \quad (8)$$

8: **end for**
9: Set $\tilde{h} := 0, \tilde{i} := 0, h^* := \max(h_1^*, \dots, h_{|\mathcal{T}|}^*)$
10: **for all** $i \in \{1, \dots, N\} : x_i \notin X$ **do**
11: **if** $(y_i - \mathcal{T}(x_i))^2 \geq \max\{\tilde{h}, h^*\}$ **then**
12: update $\tilde{h} := (y_i - \mathcal{T}(x_i))^2$ and $\tilde{i} := i$
13: **end if**
14: **end for**
15: **if** $\tilde{h} > 0$ **then** set $X := X \cup \{x_{\tilde{i}}\}, Y := Y \cup \{y_{\tilde{i}}\}$
16: **else** Return $\mathcal{T}, h^*, |X|$ *(support subsample found)*
17: **end if**
18: **end while**
19: Return $\mathcal{T}, h^*, N$ *(no support subsample found)*

the CART algorithm, are deterministic: as a consequence $\mathcal{A}_{k^*}(X, Y)$ is a unique, possibly suboptimal, solution of (11). We can now map our optimization program (11) with the optimization program (1), and note that all conditions required to apply the scenario approach for the nonconvex case are satisfied. In particular, the set $X \cup Y$ constructed in Algorithm 1 is by construction a support subsample (not necessarily the smallest one): indeed we check in lines 9-17 whether the obtained error bound h^* is also satisfied for all samples in $\mathcal{X} \setminus X, \mathcal{Y} \setminus Y$, which implies that the solution $\mathcal{A}_{k^*}(X, Y)$ is also a unique, possibly suboptimal, solution of (11) using all the samples $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$. By condition (4) stated in Section II, (9) holds for ϵ as defined in (10). This concludes the proof. ∎

Algorithm 1 is based on a suboptimal search for a support subsample set, which requires to apply the CART for every candidate support subsample. In some real cases, as happens for the experimental dataset we use in Section V, this approach can provide very conservative probabilistic bounds. To overcome this issue we propose Algorithm 2, based on the idea that the CART algorithm is applied once on the whole dataset (line 3), and the candidate support subsample sets are searched within each leaf by removing, one by one, samples in the leaf (line 8) until the reduced set satisfies the support subsample property (lines 10-13). This is formalised in Algorithm 2 below, where for the j-th leaf we will denote by $\ell_j = \{x_{j_1}, \dots, x_{j_{|\ell_j|}}\}$ the input samples in ℓ_j and by $\{y_{j_1}, \dots, y_{j_{|\ell_j|}}\}$ the corresponding output samples.

Algorithm 2

1: **Input:** datasets $\mathcal{X} = \{x_i\}_{i=1}^N, \mathcal{Y} = \{y_i\}_{i=1}^N$
2: **Output:** $\mathcal{T}, h^*, \{|\ell_j|, k_j^*\}_{j=1}^{|\mathcal{T}|}$
3: Apply CART algorithm to $(\mathcal{X}, \mathcal{Y})$, obtaining as output a tree structure $\mathcal{T}$ with leaves $\{\ell_j\}_{j=1}^{|\mathcal{T}|}$
4: **for all** $j = 1, \dots, |\mathcal{T}|$ **do**
5: Compute in leaf ℓ_j an error bound h_j^* and an affine predictor $m_j^* \in \mathbb{R}^n, m_{j,0}^* \in \mathbb{R}$ by solving the QP:

$$\min_{h_j, m_j, m_{j,0}} h_j$$
$$\text{subject to } (y_{j_i} - m_j^T x_{j_i} - m_{j,0})^2 \leq h_j, \forall x_{j_i} \in \ell_j \quad (12)$$

6: Set $X_j := \ell_j, Y_j := \{y_{j_i}\}_{i=1}^{|\ell_j|}, k_j^* := |X_j|$
7: **while** $X_j \neq \varnothing$ **do**
8: **for all** $\zeta = 1, \dots, |X_j|$ **do**
9: Compute in leaf ℓ_j an error bound $\tilde{h}_j^*$ and an affine predictor $\tilde{m}_j^* \in \mathbb{R}^n, \tilde{m}_{j,0}^* \in \mathbb{R}$ by solving the QP:

$$\min_{\tilde{h}_j, \tilde{m}_j, \tilde{m}_{j,0}} \tilde{h}_j$$
$$\text{subject to } (y_{j_i} - \tilde{m}_j^T x_{j_i} - \tilde{m}_{j,0})^2 \leq \tilde{h}_j, \forall x_{j_i} \in X_j \setminus x_\zeta \quad (13)$$

10: **if** $h_j^* = \tilde{h}_j^*, m_j^* = \tilde{m}_j^*, m_{j,0}^* = \tilde{m}_{j,0}^*$ **then**
11: Set $X_j := X_j \setminus \{x_\zeta\}, Y_j := Y_j \setminus \{y_\zeta\}$
12: Set $k_j^* := |X_j|$
13: Exit **For** cycle *(support subsample found)*
14: **else** if $\zeta = |X_j|$ Exit **For** and **While** cycles *(no support subsample found)*
15: **end if**
16: **end for**
17: **end while**
18: **end for**
19: Return $\mathcal{T}, \max(h_1^*, \dots, h_{|\mathcal{T}|}^*), \{|\ell_j|, k_j^*\}_{j=1}^{|\mathcal{T}|}$

Proposition 2: Let Algorithm 2 be applied to a dataset of inputs $\mathcal{X}$ and outputs $\mathcal{Y}$, then for any given $\beta \in (0, 1)$

$$\mathcal{P}^N\{\mathcal{P}\{x \in \mathbb{R}^n, y \in \mathbb{R} : (y - \mathcal{T}(x))^2 \geq h^*\} > \epsilon\} \leq \beta, \quad (14)$$

where $\epsilon = \max_{j=1,\dots,|\mathcal{T}|} \min\{\epsilon_{j1}, \epsilon_{j2}\}$,

$$\epsilon_{j1} = \begin{cases} 1 & k_j^* = |\ell_j|, \\ 1 - \left(\frac{\beta}{|\ell_j|\binom{|\ell_j|}{k_j^*}}\right)^{(|\ell_j| - k_j^*)^{-1}} & \text{otherwise.} \end{cases} \quad (15)$$

and where ϵ_{j2} can be computed solving the following

$$\beta = \sum_{i=0}^{n+1} \binom{|\ell_j|}{i} (\epsilon_{j2})^i (1 - \epsilon_{j2})^{|\ell_j| - i}. \quad (16)$$

Proof: Algorithm 2 first applies the CART algorithm to the whole dataset, and considers separately the convex optimization programs (12) solved in each leaf ℓ_j. Let $(h_j^*, m_j^*, m_{j,0}^*) = \mathcal{A}_{j,N}(\ell_j)$ be the obtained solution in each

leaf, which is clearly optimal and unique. We can now map our optimization programs (12) with the optimization program (1), and note that all conditions required to apply the scenario approach, both for the convex and nonconvex cases, are satisfied.

In particular, the set $X_j \cup Y_j$ iteratively generated in Algorithm 2 is by construction a support subsample (not necessarily the smallest one): indeed in lines 8-16 we search, removing one sample by one (x_ζ), until the optimal solution of (12), using all samples in ℓ_j, is equal to the solution of (13), only using samples in $X_j \setminus x_\zeta$. Formally, the set X_j of cardinality $k_j^* \leq N$ is constructed such that $\mathcal{A}_{j,k_j^*}(X_j) = \mathcal{A}_{j,|\ell_j|}(\ell_j)$. By conditions (3), (4) respectively stated in Section II, (14) holds for the worst case violation probability of all leaves, i.e. for $h^* = \max_{j=1,\ldots,|\mathcal{T}|} h_j^*$, and $\epsilon = \max_{j=1,\ldots,|\mathcal{T}|} \epsilon_j$, where ϵ_j is the least conservative bound obtained by applying both the nonconvex-case and the convex-case conditions to the inner optimization in each leaf ℓ_j. More precisely, for each ℓ_j, $\epsilon_j := \min\{\epsilon_{j1}, \epsilon_{j2}\}$. It directly follows by (3) that ϵ_{j1} is defined as in (15), with the remark that in the optimization program (12) the dimension d (using the same notation as in (3)) of the optimization variable $(h_j, m_j, m_{j,0})$ is given by $d = 1 + n + 1$. Also, it directly follows by (4) that ϵ_{j2} is defined as in (16). This concludes the proof. ■

V. Experimental validation

In this section we validate the proposed method on a dataset collected from a real benchmark consisting of a building HVAC system (see [15] for a detailed description): we first compare it in terms of accuracy with the existing method described in Section III, which cannot provide probabilistic guarantees; then we statistically evaluate, on the testing dataset, the effectiveness of the theoretical guarantees derived in this paper. We provide results on two different scenarios: room temperature and energy consumption prediction.

Case study description. We consider a two-story building located inside the Coppito campus of the University of L'Aquila, Italy, coordinates 42°22'10.4"N and 13°20'54.2"E. Each floor is composed of 5 zones – 4 rooms and a small lobby, and can be independently controlled. The gross area of the ground and first floor is 72 m^2 and 77 m^2, respectively. About the heating system, the building is equipped with a variable refrigerant flow heat pump (a type of rooftop unit) from Mitsubishi. The heating system comprises of (1) an outdoor unit on the roof that includes a compressor and an evaporator, and (2) an indoor unit (also called split) in each zone that includes a fan and a condenser. Heating is provided through refrigerant conduits connecting indoor and outdoor equipment. The thermal energy from the evaporation and compression phases is carried by the refrigerant. This energy is transferred by the condenser into the zones where warm air is then distributed by the fan. Additionally, each room and lobby is equipped with a temperature sensor and a thermostat that are used to monitor and control, via temperature setpoint selection, the room temperature. Power consumption of the building is measured using a multimeter. The system is configured to be programmatically controlled and monitored via M-NET (Mitsubishi network) protocol. Temperature control is performed via a traditional control system for buildings that relies on fixed rules for manipulation of temperature setpoints. During winter, when the setpoint in a zone is kept constant, the corresponding indoor unit is switched ON when the measured temperature is $\sim$1.5°C below the setpoint. As the zone starts to heat up, the indoor unit is switched OFF when the measured temperature exceeds the setpoint by $\sim$0.5°C. Since different rooms have different temperatures, the external unit may be kept ON for usually longer periods of time. For further details about the benchmark please see [15].

Dataset description. We collected from the building two different datasets: a *training dataset*, $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$, that is used to learn the models, and a *testing dataset*, $(\mathcal{X}_{\text{test}}, \mathcal{Y}_{\text{test}})$, for the validation process, both for the model accuracy and the probability guarantees. The sampling time is of 2 minutes. The training dataset contains measurements from November 17^{th}, 2018, to February 26^{th}, 2019, for a total of $N = 72369$ samples, while the testing dataset contains data from March 6^{th}, 2019, to March 22^{th}, 2019, for a total of 11250 samples. The input datasets are composed by 23 variables consisting of the 10 room temperatures T_i, $i = 1, \ldots, 10$, the 10 temperature setpoints to control each room $\mathsf{T}_{\mathsf{sp,i}}$, $i = 1, \ldots, 10$, the apparent energy consumption in the sampling interval, i.e. the apparent energy consumption since the last measurement, E, and 3 disturbances related to weather conditions $\mathsf{D} = [\mathsf{T}_{\mathsf{out}}, \mathsf{H}, \mathsf{S}]$, that correspond to outside air temperature, external humidity, and solar radiation respectively. As we will see later on, depending on the application, e.g. temperature or energy consumption prediction, a subset of such variables is used, and regressive terms are added to the datasets to improve the modeling accuracy. The output datasets are composed by the room temperatures, T_i, $i = 1, \ldots, 10$, and apparent energy, E, measurements.

Results on temperature prediction. In this section we show the results obtained by applying the methodologies described above for the prediction of the room temperatures. For the simulation we consider a single room, e.g. room 1, with datasets composed by temperature, control setpoint and weather variables, together with their regressive terms. Simulations for the other 9 rooms provided similar results. The training dataset is thus defined as follows, the testing dataset is defined similarly with testing data:

$$\begin{aligned}\mathcal{X}_{\text{train}} = \{&\mathsf{T}_1(t), \mathsf{T}_1(t-1), \mathsf{T}_{\text{sp},1}(t), \mathsf{T}_{\text{sp},1}(t-1),\\ &\mathsf{D}(t), \ldots, \mathsf{D}(t-5)\}_{t=5}^{N},\\ \mathcal{Y}_{\text{train}} = \{&\mathsf{T}_1(t+1)\}_{t=5}^{N}.\end{aligned}$$

We used $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$ to identify the models. In particular, we identified 3 different models: (i) the first is obtained following Algorithm 0 from [10], and we refer to it as *NAHS20*; (ii) the second is obtained following Algorithm 1, and we refer to it as *Case 1*; (iii) the third is obtained following Algorithm 2, and we refer to it as *Case 2*.

The results of the validation process on the testing dataset

are shown in Figure 1. Since the testing dataset is composed by 17 days the figure would have been messy, thus for the sake of clarity we randomly picked 12 hours and showed the plot of the trajectories of March 8^{th} from 6 am to 6 pm.

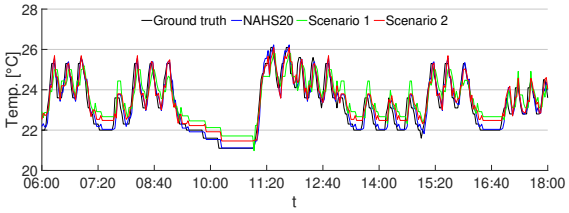

Fig. 1: Temperature trajectories comparison from 6 am to 6 pm on March 8^{th}, 2022.

Results show that the accuracy provided by *NAHS20* is the best one, and this is expected as the learning procedure aims at minimizing the mean square error. Differently, *Case 1* and *Case 2* approaches minimize an upper bound of such square value, hence giving a suboptimal solution with the advantage of providing probabilistic guarantees as described in the above sections. The accuracy error over the whole testing dataset is provided in Table I in terms of NRMSE, normalized with respect to the mean of the temperature values in the testing dataset, and expressed in percentage.

Method	*NAHS20*	*Case 1*	*Case 2*
NRMSE	1.85%	2.98%	2.19%

TABLE I: Room temperature prediction errors provided by the validation of the 3 models over the testing dataset .

In Table II the violation bounds are reported. For the computation, we chose $\beta = 10^{-6}$. We can see how the theoretical bound ϵ of *Case 1* is equal to 1, i.e. it is useless in the considered experimental dataset - of course, it might be useful on different datasets. Differently, the results from *Case 2* show a theoretical bound ϵ that is coherent to the one obtained statistically on the testing dataset.

Remark 1: We recall that the scenario approach can be applied with the only assumption on the probability space that each scenario is drawn independently. It is well known from the scientific literature that this huge generality is paid by some conservatism of the derived probabilistic bounds: this is the reason why we believe that, in our validation setup, the theoretical bound we have derived can be considered reasonably close to the one obtained statistically. We also remark that, in order to satisfy the assumption that each scenario is drawn independently, our probabilistic bound on the estimation error holds when both the training and testing datasets are generated using the same control policy.

Results on energy prediction. In this section we show the results obtained by applying the methodologies described above for the prediction of the energy consumption. Differently from the previous case, where for the prediction of the temperature of a room the measurements from that room are

Violation probability bound	Case 1: $h^* = 1.52$	Case 2: $h^* = 1.59$
Theoretical	$\epsilon = 1$	$\epsilon = 0.0290$
Statistical	$\hat{\epsilon} = 0.0060$	$\hat{\epsilon} = 0.0009$

TABLE II: Violation bounds obtained from the simulations of *Case 1* and *Case 2* over the temperature dynamics. The first line of the table represents the theoretical bounds as described in Section IV. The second line represents the constraints violation ratio derived on the testing dataset.

enough for a good model identification, the energy is affected by the behaviour of the whole building. For this reason, we consider the measurements from all the rooms, in addition to the energy itself and the weather variables. The training dataset, and similarly the testing dataset, are thus defined as follows:

$$\mathcal{X}_{\text{train}} = \{\mathsf{E}(t), \mathsf{T}(t), \mathsf{T}(t-1), \mathsf{T}_{\text{sp}}(t), \mathsf{T}_{\text{sp}}(t-1), \mathsf{D}(t), \dots, \mathsf{D}(t-5)\}_{t=5}^{N},$$
$$\mathcal{Y}_{\text{train}} = \{\mathsf{E}(t+1)\}_{t=5}^{N},$$

where $\mathsf{T} = [\mathsf{T}_1, \dots, \mathsf{T}_{10}]$ and $\mathsf{T}_{\text{sp}} = [\mathsf{T}_{\text{sp},1}, \dots, \mathsf{T}_{\text{sp},10}]$.

Similarly to the temperature case, we used $(\mathcal{X}_{\text{train}}, \mathcal{Y}_{\text{train}})$ to identify the 3 different models described above. The results of the validation process on the testing dataset are shown in Figure 2. As before, for the sake of clarity, only the trajectories of March 8^{th} from 6 am to 6 pm are shown.

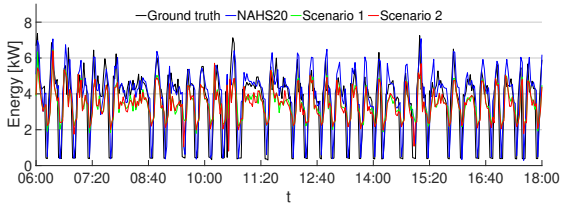

Fig. 2: Energy trajectories comparison from 6 am to 6 pm on March 8^{th}, 2022.

The accuracy error over the whole testing dataset is provided in Table III in terms of NRMSE, normalized with respect to the minimum and maximum values of the energy in the testing dataset, and expressed in percentage. Results show that the accuracy provided by *NAHS20* is the best one. As usual, the energy consumption prediction is much less accurate than the temperature prediction; nevertheless, the obtained accuracy is coherent with the 15% tolerance considered as acceptable by the *American Society of Heating, Refrigerating and Air-Conditioning Engineers* (ASHRAE).

Method	*NAHS20*	*Case 1*	*Case 2*
NRMSE	12.69%	16.72%	16.77%

TABLE III: Modeling accuracy errors provided by the validation of the 3 models over the testing dataset for the prediction of the energy consumption.

In Table IV the violation bounds are reported. For the computation we chose $\beta = 10^{-6}$. Results are coherent with the theoretical bounds, which in the case of energy consumption prediction are, as expected given the nonlinear nature of the energy variable, more conservative than the temperature case.

Violation probability bound	Case 1: $h^* = 4.26$	Case 2: $h^* = 4.18$
Theoretical	$\epsilon = 1$	$\epsilon = 0.0670$
Statistical	$\hat{\epsilon} = 0.0004$	$\hat{\epsilon} = 0.0004$

TABLE IV: Violation bounds obtained from the simulations of *Case 1* and *Case 2* over the energy dynamics. The first line of the table represents the theoretical bounds as described in Section IV. The second line represents the constraints violation ratio derived on the testing dataset.

VI. Conclusions

In this paper we extended our research line on learning techniques based on a combination of the Regression Trees method and ARX identification to derive models of dynamical systems exploiting historical data. We improved our methodology to formally relate it with the scenario approach framework, thus providing bounds on the probability that testing samples do not satisfy the estimation model accuracy obtained on the training dataset. We validated our technique on a dataset collected from a real experimental setup, showing that the obtained performance is valid both in terms of accuracy and of effectiveness of the derived probabilistic guarantees. In future work we plan to extend our results to provide probabilistic guarantees on the structural properties of the identified model and to consider other application domains [18], [19], [28].

References

[1] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[2] D. Ding, Q.-L. Han, X. Ge, and J. Wang, "Secure state estimation and control of cyber-physical systems: A survey," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 176–190, 2020.

[3] Y. Ma, J. Matusko, and F. Borrelli, "Stochastic model predictive control for building hvac systems: Complexity and conservatism." *IEEE Trans. Contr. Sys. Techn.*, vol. 23, no. 1, pp. 101–116, 2015.

[4] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, pp. 15–27, 2012.

[5] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. S. Vincentelli, "Handling model uncertainty in model predictive control for energy efficient buildings," *Energy and Buildings*, vol. 77, pp. 377–392, 2014.

[6] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith, "Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 1–12, 2016.

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

[8] A. Jain, F. Smarra, and R. Mangharam, "Data predictive control using regression trees and ensemble learning," in *2017 IEEE 56th annual conference on decision and control (CDC)*. IEEE, 2017, pp. 4446–4451.

[9] F. Smarra, A. Jain, R. Mangharam, and A. D'Innocenzo, "Data-driven switched affine modeling for model predictive control," in *IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'18)*. IFAC, 2018, pp. 199–204.

[10] F. Smarra, G. D. Di Girolamo, V. De Iuliis, A. Jain, R. Mangharam, and A. D'Innocenzo, "Data-driven switching modeling for mpc using regression trees and random forests," *Nonlinear Analysis: Hybrid Systems*, vol. 36, p. 100882, 2020.

[11] F. Smarra and A. D'Innocenzo, "Learning markov jump affine systems via regression trees for mpc," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5252–5257, 2020.

[12] O. L. V. Costa, M. D. Fragoso, and R. P. Marques, *Discrete-time Markov jump linear systems*. Springer Science & Business Media, 2006.

[13] D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1468–1480, 2012.

[14] F. Smarra, A. Jain, T. De Rubeis, D. Ambrosini, A. D'Innocenzo, and R. Mangharam, "Data-driven model predictive control using random forests for building energy optimization and climate control," *Applied energy*, vol. 226, pp. 1252–1272, 2018.

[15] A. Jain, F. Smarra, E. Reticcioli, A. D'Innocenzo, and M. Morari, "Neuropt: Neural network based optimization for building energy management and climate control," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 445–454.

[16] G. D. Di Girolamo, F. Smarra, V. Gattulli, F. Potenza, F. Graziosi, and A. D'Innocenzo, "Data-driven optimal predictive control of seismic induced vibrations in frame structures," *Structural Control and Health Monitoring*, vol. 27, no. 4, 2020.

[17] L. F. Florenzan Reyes, F. Smarra, Y. Z. Lun, and A. D'Innocenzo, "Learning markov models of fading channels in wireless control networks: a regression trees based approach," in *2021 29th Mediterranean Conference on Control and Automation (MED)*, 2021, pp. 232–237.

[18] E. Reticcioli, G. Di Girolamo, C. Di Carlo, F. Smarra, and A. D'Innocenzo, "Machine learning-based approaches comparison for netflix/dazn streaming and real traffic prediction," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 3102–3107.

[19] E. Reticcioli, G. D. Di Girolamo, F. Smarra, A. Torzi, F. Graziosi, and A. D'Innocenzo, "Modeling and control of priority queueing in software defined networks via machine learning," *IEEE Access*, vol. 10, pp. 91 481–91 496, 2022.

[20] F. Smarra, J. Tjen, and A. D'Innocenzo, "Learning methods for structural damage detection via entropy-based sensors selection," *International Journal of Robust and Nonlinear control*, vol. 32, no. 10, pp. 6035–6067, 2022.

[21] G. C. Calafiore and M. C. Campi, "Uncertain convex programs: randomized solutions and confidence levels," *Math. Program.*, vol. 102, no. 1, pp. 25–46, 2005.

[22] ——, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, May 2006.

[23] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, 2008.

[24] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annu. Rev. Control*, vol. 33, no. 2, pp. 149–157, 2009.

[25] M. C. Campi, S. Garatti, and F. A. Ramponi, "Non-convex scenario op- timization with application to system identification," in *Proc. 54th IEEE Conf. Decis. Control, Osaka, Japan*. IEEE, 2015, pp. 4023–4028.

[26] ——, "A general scenario theory for nonconvex optimization and decision making," *IEEE Transactions on Automatic Control*, vol. 63, no. 12, pp. 4067–4078, Dec. 2018.

[27] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[28] F. Simonetti, A. D'Innocenzo, and C. Cecati, "Neural network model predictive control for chb converters with fpga implementation," *IEEE Transactions on Industrial Informatics*, pp. 1–12, 2023.