Energy & Buildings 311 (2024) 114161

Contents lists available at ScienceDirect

Energy & Buildings

journal homepage: www.elsevier.com/locate/enbuild

Demand response for residential building heating: Effective Monte Carlo Tree Search control based on physics-informed neural networks

Fabio Pavirani [a,*], Gargya Gokhale [a], Bert Claessens [a,b], Chris Develder [a]

[a] IDLab Ghent university – imec, Technologiepark Zwijnaarde 126, 9052, Gent, Belgium
[b] Beebop, Belgium

ARTICLE INFO

Keywords:
Building control
Demand response
Monte Carlo Tree Search
Physics informed neural network
Machine learning
Thermal dynamics modeling

ABSTRACT

To reduce global carbon emissions and limit climate change, controlling energy consumption in buildings is an important piece of the puzzle. Here, we specifically focus on using a demand response (DR) algorithm to limit the energy consumption of a residential building's heating system while respecting user's thermal comfort. In that domain, Reinforcement learning (RL) methods have been shown to be quite effective. One such RL method is Monte Carlo Tree Search (MCTS), which has achieved impressive success in playing board games (go, chess). A particular advantage of MCTS is that its decision tree structure naturally allows to integrate exogenous constraints (e.g., by trimming branches that violate them), while conventional RL solutions need more elaborate techniques (e.g., indirectly by adding penalties in the cost/reward function, or through a backup controller that corrects constraint-violating actions). The main aim of this paper is to study the adoption of MCTS for building control, since this (to the best of our knowledge) has remained largely unexplored. A specific property of MCTS is that it needs a simulator component that can predict subsequent system states, based on actions taken. A straightforward data-driven solution is to use black-box neural networks (NNs). We will however extend a Physics-informed Neural Network (PiNN) model to deliver multi-timestep predictions, and show the benefit it offers in terms of lower prediction errors (−32% MAE) as well as better MCTS performance (−4% energy cost, +7% thermal comfort) compared to a black-box NN. A second contribution will be to extend a vanilla MCTS version to adopt the ideas applied in AlphaZero (i.e., using learned prior and value functions and an action selection heuristic) to obtain lower computational costs while maintaining control performance.

1. Introduction

The urgent need to address climate change implies an increased pressure to limit energy usage and mitigate carbon emissions. In the power grid, this has spurred a significant shift towards "greener" energy sources, particularly renewable energy sources (RES). Yet, since such RES introduce a larger degree of supply uncertainty, maintaining the power grid's balance requires controlling the demand, e.g., through demand response (DR). Of that demand, a substantial fraction constitutes residential building heating: e.g., residential consumption in the EU amounted to 26.1%, of which 63.6% is represented by heating [1]. Hence our focus on residential heating system control in this paper.

To realize such residential heating control, two main strategies have been adopted [2]: either (i) model-based controllers, or (ii) purely data-driven controllers. For (i), Model Predictive Control (MPC) is typically used, defining a mathematical optimization problem that relies on an explicit model of the building's thermal dynamics. Solving the problem then delivers (an approximation of) the optimal control decision. Such model-based techniques, although very efficient, are highly dependent on the model used, which typically cannot generalize over different houses [3]. In terms of (ii), one of the most used data-driven techniques is Reinforcement Learning (RL), which learn from directly interacting with the environment (e.g., the building). Yet, often a large amount of such interactions (i.e., data) is needed to learn a good policy taking (near-)optimal control actions, which can be hard/costly to obtain. To alleviate this, a simulator (rather than the actual building environment) can be used, but as in MPC, explicitly defining such building models is hampered by the reliance on expert knowledge and lack of generalization across buildings.

Despite their promising results in producing effective control policies, several challenges regarding RL algorithms in the Building Energy Management (BEM) domain are yet to be solved, such as their lack of

[*] Corresponding author.
E-mail address: fabio.pavirani@ugent.be (F. Pavirani).

https://doi.org/10.1016/j.enbuild.2024.114161
Received 16 January 2024; Received in revised form 27 March 2024; Accepted 5 April 2024
Available online 15 April 2024

Nomenclature

t	Time-step index
k	Depth of the node index
Δ_t	Length of a time-step
(X, U, f, ρ)	Stochastic POMDP
$(\tilde{X}, \tilde{U}, \tilde{f}, \tilde{\rho})$	Deterministic FOMDP
x	Full system state of the MDP
$x^{\mathrm{b}} \in \mathrm{X}^{\mathrm{b}}$	Observable building-related state
$x^{\mathrm{e}} \in \mathrm{X}^{\mathrm{e}}$	Observable exogenous state
$x^{\mathrm{h}} \in \mathrm{X}^{\mathrm{h}}$	Non-observable state
z	PiNN latent state
τ	Time of the day
T_{r}	Room temperature
T_{m}	Bulk temperature of the building
$T_{\mathrm{r\_set}}$	Desired room temperature
u	Action signal of the heat pump
u^{phys}	Energy consumed by the heat pump
T_{a}	Outside temperature
λ	Electricity price
$N(\tilde{x})$	Number of tree visit of state $\tilde{x}$
$N(\tilde{x}, \tilde{u})$	Occurrence of action $\tilde{u}$ from $\tilde{x}$ in the tree
$P(\tilde{x}, \tilde{u})$	Prior probability in the tree traversals
h	Prediction horizon
d	Maximum depth of the tree
l	Leaf node's depth in a tree traversal
c_1, c_2	Parameters to tune the reward function
α	Parameter to balance tree exploration
$\Delta_T^{\pm}$	Temperature flexibility bounds

interpretability and safety [4]. Indeed, RL techniques typically rely on data-driven function approximators, such as Neural Networks [5], to provide their control actions, resulting in a 'black-box' decision-making structure. The black-box nature implies that the resulting actions cannot be explained/motivated, which triggers reluctance and distrust among users and manufacturers in using such technology [4]. Furthermore, since RL agents learn from historical data only, they may take poor decisions particularly in cases they have not encountered before: their actions are not based on any notion of, e.g., physical laws governing the system behavior and thus cannot be interpreted [6,7]. This requires backup controllers to override any RL agent actions that would be hazardous [4]. In general, it is hard to impose constraints underlying such backup controllers directly into learning an RL policy.

To address these challenges of (i) interpretability, and (ii) incorporating constraints, Monte Carlo Tree Search (MCTS) algorithms are an attractive solution. Indeed, they are based on estimating the "value" of actions (learned from multiple possible rollouts of action sequences), and their tree structure naturally allows to include constraints (e.g., by accordingly pruning the allowed actions from the tree). Such MCTS solutions relatively recently have achieved significant success in long-standing challenges of board games [8]. However, MCTS has remained under-explored in building energy management systems. Thus, our paper aims to set baseline performance benchmarks of MCTS solutions in this domain, particularly for heating systems.

We developed an MCTS algorithm that optimizes energy cost while following user-defined constraints. To simulate the environment dynamics in the MCTS rollouts, we need a model that captures the thermal dynamics of a building. Starting from a basic data-driven black-box neural network (NN) to encode the system state, we further propose a physics-informed Neural Network (PiNN) forecaster. For the latter, we extend the PhysNet presented in [9] to a multi-step prediction model. Furthermore, we enhance the standard MCTS structure by incorporating an additional Neural Network (NN) to guide tree construction, similar to the approach proposed in AlphaZero [10]. This addition enables the tree search to achieve better actions with lower computational cost, compared to its vanilla version. Our contributions in studying the effectiveness of MCTS to control a residential heat pump to maximize thermal comfort and minimize energy costs thus can be summarized as follows:

(1) For modeling the system state transitions in our MCTS solution, we extend the PhysNet model of Gokhale et al. [9] to a multi-step forecaster (Section 3.2) and show that it offers clear performance benefits, not only in reducing state forecasting errors (−32% MAE; see Section 5.1), but also in the eventual MCTS control policy performance (−4% energy cost, +7% thermal comfort; see Section 5.2);

(2) Next to the vanilla MCTS solution (Section 3.4) that adopts (1), we propose an AlphaZero-inspired extension that provides non-uniform action priors and an adjusted action selection function. We show that this AlphaZero-MCTS reduces computational cost significantly (half the number of simulations required to converge; see Section 5.3).

The main innovation of this work lies in the combination of (1)–(2), by incorporating a physics-informed model into a neural-network-enhanced MCTS solution. To the best of our knowledge, this is the first such study of MCTS in the demand response domain (in our case for heat pump control), making this work a novel contribution to the field.

2. Related work

Below we summarize key works that are relevant to our contributions stated above. Since MCTS solutions require a system model to rollout possible action sequences, we first outline recent common approaches to thermal modeling of buildings (in the context of controlling them) in Section 2.1, providing the context for contribution (1). Subsequently, in Section 2.2 we list the key control paradigms for building energy management, and thus position MCTS against relevant alternatives including MPC and common RL solutions, to frame our contribution (2).

2.1. Thermal dynamics modeling of buildings

Creating an accurate and scalable model that describes the thermal dynamics of a building is crucial to obtain a sequential controller. These models can be divided into three main categories: white-box models, black-box models, and hybrid models [11–18].

White-box models (or first principle models) are typically physics-based methods that model the system using ordinary differential equations [19,20]. Despite their accuracy, the main drawback of white-box models is the complexity of the physical model developed [12], which limits large-scale adoption (cf. a building-specific model needs to be constructed).

Rather than explicitly modeling the system behavior grounded in fundamental mathematical equations (thus requiring expert knowledge), *black-box methods* purely learn from observations, and thus construct a model of the thermal dynamics directly from historical data, which in principle eases their wide-scale adoption. Today, black-box models are typically based on (deep) neural networks (NN). To properly generalize over a large range of conditions, such NNs need a large and complete dataset, which often is hard to obtain [6]. Furthermore, their black-box nature inherently makes NN models uninterpretable, and they may still take surprising decisions, implying generalization issues [7].

Hybrid models (or gray-box models) aim to combine the best of both black-box and white-box models, trying to solve their respective limitations while exploiting their benefits. The idea is to use generic and usually manageable physical equations together with data-based training to obtain a simple yet accurate model of the building [12,21,22]. Because of their simplicity, typically linear RC models are adopted [23,24].

One particular subfamily of hybrid models is based on *Physics-informed Neural Networks (PiNN)* [25], where prior physical knowledge is incorporated into a data-driven NN prediction model. The latter is typically achieved by using (possibly simplified) physical laws phrased as partial differential equations, which are then used to regularize NN predictions, e.g., by penalizing NN predictions that deviate from the behavior as dictated by physical laws. By "infusing" such physical prior knowledge into a NN model, it can provide a higher level of physical consistency, while still allowing to exploit the NN capacity to capture highly non-linear correlations (that would potentially not be reflected in an approximate physical model). These characteristics are particularly relevant for the problem of optimal control action search, making PiNN techniques a very valuable solution in building energy control. Besides their better modeling performance, PiNN has shown promising results regarding their sample efficiency, requiring less data for acceptable results compared to pure black-box models [9].

Various types of PiNNs have been applied in modeling the thermal dynamics of buildings from a control-oriented point of view, such as [6,9,26]. Gokhale et al. [9] modified the loss function of an encoder-based NN to guide it toward physical adherence of the latent state. Drgoňa et al. [26] modeled the state, input, disturbance, and output matrices of a classical linear equation using four different NNs and enforced physical constraints on them. Di Natale et al. [6] modeled the NN architecture to enforce physical consistency on its prediction. Both the models in [6,26] infuse physical knowledge as an arrangement of the NNs to obtain physical consistency of the predictions. Despite their good results, these networks do not directly access to the prior physical knowledge. Differently from those, the PiNN used in [9] directly infuses the prior knowledge *inside* the NN equations, by adding a physics loss in the training process. Moreover, the architecture in [9] can provide a low-dimensional and physically interpretable latent space in addition to their predictions. The latent space's values are trained to adhere to relevant and hard-to-measure insight of the building, such as the bulk temperature. These values can be used, for example, to extend the input taken by controllers to improve their performance. For this reason, we extended the PhysNet of Gokhale et al. [9] towards a multi-step predictor to enable accurate MCTS tree rollouts. We then incorporated the hidden state in the controller's input, enabling it to make more conscious decisions.

2.2. Demand response control of buildings

With demand response, we refer to control strategies that optimize the energy usage in response to external signals, e.g., a time-varying electricity price. We specifically consider building energy management systems (BEMS), for which [2] provides a quantitative comparison of state-of-the-art solutions. In terms of algorithms, they can be roughly classified into two main categories, which we discuss below in more detail: Model Predictive Control (MPC), and Reinforcement Learning (RL) approaches.

2.2.1. Model Predictive Control (MPC)

With MPC, an optimization problem is solved (or approximated) for each control action in a receding horizon approach. MPC uses a model to estimate future state transitions (depending on actions taken) and thus requires an accurate model to get satisfactory results. As outlined in Section 2.1, such models can be either white-box, black-box, or gray-box (hybrid), each with their respective benefits and drawbacks. All three approaches have been applied successfully in real applications [27]. For a more detailed list and review of specific MPC variants, we refer to [28].

White-box MPC models have been shown to be promising when a good model is provided. Despite this, their application needs a non-negligible amount of expert knowledge in defining/selecting the model and framing the mathematical optimization problem, making it difficult to generalize to a broad scale of settings (e.g., to multiple varying buildings' parameters). To reduce the human effort in designing an MPC controller, data-driven MPC algorithms have received increasing attention. Among these, deep learning MPC methods using neural networks (NN) are the most popular [29–32]. Yet, adopting highly non-linear models such as NN-based ones implies a more complex optimization problem, in the sense that it can no longer be formalized as a linear program (as is typically the case in MPC). This complicates solving the optimization problem using readily available commercial solvers [33]

2.2.2. Reinforcement Learning (RL)

Another family of data-driven algorithms that has shown promising results in the last decade is Reinforcement Learning (RL). RL algorithms formalize the system to be controlled as a Markov Decision Process (MDP) and interact with it to learn an adequate policy that maximizes the expected reward[1] obtained from the environment. Such RL controllers have been applied successfully to several problems in the building energy management domain [34–38].

Compared to MPC solutions, RL algorithms efficiently learn optimal control actions in a data-driven fashion, thus requiring less expert knowledge to scale the technique. However, multiple challenges arise with RL techniques [4, Question 10]. For example, RL algorithms often require a large amount of data for training to reach acceptable performance. Also, most RL algorithms used in the BEM environment are model-free [3], which makes it hard to incorporate explicitly the sequential planning nature across multiple timesteps, as well as specific constraints. Instead, model-free RL techniques have to implicitly learn underlying constraints from observed data [4].

2.2.3. Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) can be a solution to the problems described above, particularly (i) the dependence of MPC on an explicit system model, and (ii) the difficulty of RL in dealing with a sequential planning problem as well as specific constraints to be respected. Since MCTS just needs the model to be able to role out possible scenarios by simulating them, it is oblivious to the specific model nature, thus alleviating (i) and allowing more scalable/practical models, including black-box or hybrid (gray-box) models. Given the tree-based nature of MCTS-based planning, it can manage future exogenous constraints naturally by trimming off actions that would lead to violating them (which RL could struggle with, as (ii) suggests). This makes MCTS a promising candidate for DR problems where different external signals and objectives need to be followed while handling different operational constraints.

MCTS was first introduced in 2006 by Coulom [39] and gained its popularity thanks to the UCT algorithm [40]. MCTS has already been applied in various demand response solutions for balancing the electrical grid [41–44]. Specifically for our case of residential demand response on heating systems, the most relevant MCTS work is that of Kiljander et al. [45]. They modeled the household thermal dynamics with a Feed Forward Neural Network (FFNN) and used it as a simulator in conjunction with a standard MCTS algorithm to generate planning decisions. The algorithm was then evaluated in a real household, demonstrating the applicability of the technique in real-world scenarios. Compared to Kiljander et al., we use a similar conceptual approach using a NN-based simulator in an MCTS framework. Two notable differences pertain to our contributions listed in Section 1. First,

[1] Alternatively, minimize the expected cost.

rather than a black-box NN, we adopt a gray-box, physics-informed NN (PiNN) to model the household, thus improving the physical consistency of the simulator. Second, inspired by the recent success of Silver et al. [10,46,47], we add a DL layer to the MCTS algorithm to estimate action values, making it more computationally efficient. Our problem formulation and PiNN/MCTS methodology is presented in detail in the next section.

3. Methodology

3.1. Markov Decision Process (MDP)

We model the sequential control problem of heating a building as a Markov Decision Process (MDP) [48]. An MDP is a mathematical structure described by the tuple $(\mathrm{X}, \mathrm{U}, f, \rho)$ where X is the state space, U is the action space, $f : \mathrm{X} \times \mathrm{U} \times \mathrm{X} \to [0,1]$ is the state transition probability function, and $\rho : \mathrm{X} \times \mathrm{U} \times \mathrm{X} \to \mathbb{R}$ is the reward function. The transition probability function $f(x,u,x')$ gives the probability of transitioning from state x to a new state x' after applying action u, whereas the reward $\rho(x,u,x')$ values how “good” that action u was. Through this mathematical structure, we can model the control problem as an optimization problem to find the control policy $\pi : \mathrm{X} \to \mathrm{U}$ that gives the action $\pi(x)$ to apply in system state x to maximize the expected reward obtained from the environment. Formally, given a policy π, we consider the definition of its Q-function [48], which assesses the long-time reward resulting from applying action u in state x:

$$Q^{\pi}(x,u) \doteq \mathbb{E}_{x' \sim f(x,u,\cdot)} \left[\rho(x,u,x') + \gamma R^{\pi}(x') \right], \tag{1}$$

where:

$$R^{\pi}(x_0) \doteq \lim_{T \to +\infty} \mathbb{E}_{x_{t+1} \sim f(x,u,\cdot)} \left[\sum_{t=0}^{T} \gamma^t \rho(x_t, \pi(x_t), x_{t+1}) \right]. \tag{2}$$

We then consider the definition of the optimal Q-function [48], that is: $Q^*(x,u) \doteq \max_{\pi} Q^{\pi}(x,u)$. The optimization problem is then to find an optimal policy π^* [48] such that:

$$\pi^*(x) \in \arg\max_{u} Q^*(x,u)\,;\ \forall x \in \mathrm{X}\,. \tag{3}$$

MDPs can be classified into two distinct categories: Fully Observable Markov Decision Process (FOMDP) and Partially Observable Markov Decision Process (POMDP). With FOMDP, the states of the system $x \in \mathrm{X}$ are fully observable by the policy π. Conversely, in POMDP only a subset of the state variables is observable by the policy, thus we have $\pi : \mathrm{X}^{\mathrm{obs}} \to \mathrm{U}$, with $\dim(\mathrm{X}^{\mathrm{obs}}) < \dim(\mathrm{X})$.

The control problem of heating a building is generally represented as a stochastic POMDP, with the objective to minimize the energy cost while staying close to the desired temperature set by the users $T_{\mathrm{r\_set}}$. We use t to note the time-step index of each system/control variable. We partition the full system state vector x_t in a building-related one x_t^{b}, exogenous variables x_t^{e} (e.g., including current weather conditions at t), and unobservable components x_t^{h} which remain hidden from our control agent (e.g., internal heat gains of the building). Thus, the full state at timestep t is $x_t = (x_t^{\mathrm{b}}, x_t^{\mathrm{e}}, x_t^{\mathrm{h}})$ whereas the observable part is limited to $x_t^{\mathrm{obs}} = (x_t^{\mathrm{b}}, x_t^{\mathrm{e}})$.

More specifically, we define the observable building state at a timestep t as: $x_t^{\mathrm{b}} = (T_{\mathrm{r},t}, u_{t-1}^{\mathrm{phys}}) \in \mathrm{X}^{\mathrm{b}}$, where $T_{\mathrm{r},t}$ is the room temperature of the building we wish to control and u_{t-1}^{phys} is the energy consumed by the heating system in the previous time-step. The exogenous influences are defined as $x_t^{\mathrm{e}} = (\tau_t, T_{\mathrm{a},t}) \in \mathrm{X}^{\mathrm{e}}$ where $\tau_t \in [0,24[$ indicates the hour of the day and $T_{\mathrm{a},t}$ is the outdoor temperature. The action $u \in \mathrm{U}$ is a continuous single value indicating the control signal for the heating component. Higher values of u_t indicate more heating required for timestep t, subsequently increasing the energy consumption u_t^{phys}. The reward function is defined as:

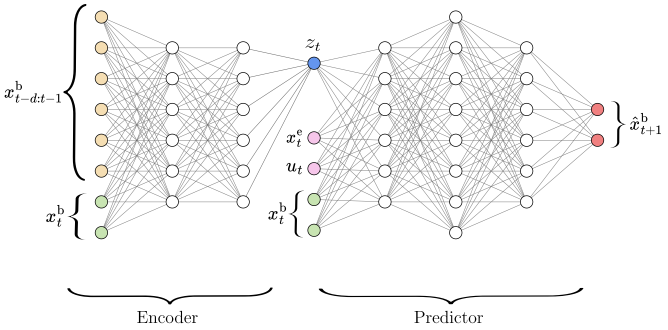

Fig. 1. Architecture of the PiNN. Yellow nodes represent past observations (until $t-1$), green nodes are the current observations at timestep t, the pink nodes represent the current exogenous state and action, the blue node is the current hidden state, and the red nodes are the predicted observations for the next timestep $t+1$. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

$$\rho(x_t,u_t) \doteq \underbrace{-u_t^{\mathrm{phys}}\lambda_t}_{\text{Cost optimization}} \underbrace{-(T_{\mathrm{r\_set},t} - T_{\mathrm{r},t+1})^{+} c_1 - (T_{\mathrm{r},t+1} - T_{\mathrm{r\_set},t})^{+} c_2}_{\text{Thermal Comfort optimization}}, \tag{4}$$

where λ_t is the energy price at time-step t, $T_{\mathrm{r\_set},t}$ is the desired temperature set by the users and c_1, c_2 are hyperparameters used to balance the energy cost objective with the user constraints. In our experiments, the user thermal comfort optimization will use asymmetrical settings ($c_1 > c_2$), because we want to avoid excessive penalization for preheating the room.

Ultimately, we will formalize the MDP as a deterministic FOMDP, which will approximate the stochastic POMDP which captures the real building behavior. That FOMDP is further detailed in Section 3.3. As stated before, to model the transition function f in the MDP, which essentially defines the thermal dynamics of the building, we will make use of a PiNN that predicts state transitions, which we describe next.

3.2. Physics-informed Neural Network (PiNN)

We use a PiNN architecture to forecast the next building states of a system $x^{\mathrm{b}} \in \mathrm{X}^{\mathrm{b}}$ by inserting physical equations into the learning loss function of the NN. We follow the PhysNet structure proposed by Gokhale et al. [9], using an encoder-based neural network. PhysNet is specifically built for projecting a time series spanning the $d \in \mathbb{N}$ most recent building states $x_{t-d:t}^{\mathrm{b}} \doteq (x_{t-d}^{\mathrm{b}}, \ldots, x_t^{\mathrm{b}})$ into a compact hidden state z_t, training the NN such that these latent values adhere to certain physical laws. In particular, z_t will be trained to represent the building mass temperature in a simple RC model of the building’s thermodynamic behavior. A separate predictor NN component will then use that z_t, together with observable states (comprising building variables x_t^{b} and exogenous variables x_t^{e}) to predict state transitions, i.e., the next building state x_{t+1}^{b}, given the current action $u_t \in \mathrm{U}$. The overall encoder-predictor structure of the NN architecture is illustrated in Fig. 1.

We expanded the model [9] to obtain a multi-time-step forecast by deploying a sequential, autoregressive, multi-loss training loop with windowed inference. We considered the 2R2C model [24] shown in Fig. 2 to infuse physical knowledge into the encoder training. The physical equations obtained are then:

$$\begin{bmatrix} \dot{T}_{\mathrm{r}} \\ \dot{T}_{\mathrm{m}} \end{bmatrix} = \begin{bmatrix} -\left(\frac{1}{C_{\mathrm{r}} R_{\mathrm{ra}}} + \frac{1}{C_{\mathrm{r}} R_{\mathrm{rm}}} \right) & \frac{1}{C_{\mathrm{r}} R_{\mathrm{rm}}} \\ \frac{1}{C_{\mathrm{m}} R_{\mathrm{rm}}} & -\frac{1}{C_{\mathrm{m}} R_{\mathrm{rm}}} \end{bmatrix} \cdot \begin{bmatrix} T_{\mathrm{r}} \\ T_{\mathrm{m}} \end{bmatrix} + \\ + \begin{bmatrix} \frac{c_{\mathrm{p}} T_{\mathrm{s}}}{C_{\mathrm{r}}} \\ 0 \end{bmatrix} \cdot \dot{u} + \begin{bmatrix} \frac{1}{C_{\mathrm{r}} R_{\mathrm{ra}}} & \frac{\gamma}{C_{\mathrm{r}}} & \frac{1}{C_{\mathrm{r}}} \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} T_{\mathrm{a}} \\ G \\ I_{\mathrm{g}} \end{bmatrix}. \tag{5}$$

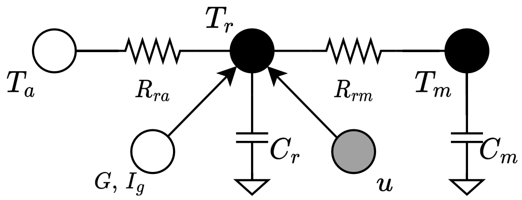

Fig. 2. RC model used in the encoder loss [24]. Black nodes ● represent building-related state variables, white nodes ○ the exogenous state variables, and the gray node ● is the control variable.

We are particularly interested in the building mass temperature, indicated as T_m. For a detailed explanation of the RC model and the notations in (5), we refer to [24]; the $R_\cdot$ and $C_\cdot$ parameters will be fine-tuned during training of the model.

Note that the encoder-predictor model of Fig. 1 only predicts the immediate next observable state (at $t+1$). To extend the predictions up to a horizon of $h \in \mathbb{N}^+$ timesteps ahead, we autoregressively feed it back to the same encoder-predictor model (e.g., to obtain $t+2$ predictions, by using inputs covering the time up to $t+1$, etc.). To train the PhysNet, two different losses are adopted for the training loop: a regression loss term for the observable state prediction (i.e., to minimize the room temperature prediction error), and a physics loss term to steer the latent state representation (i.e., to have z_t as close as possible to the building thermal mass temperature of the RC model):

$$\mathcal{L} = \mathcal{L}_\mathrm{reg} + \mathcal{L}_\mathrm{phys}\,. \tag{6}$$

More specifically, the regression loss is an L2 loss between the predicted observable states and the measured ones:

$$\mathcal{L}_\mathrm{reg} \doteq \sum_{t=1}^{N} \left(\hat{x}^\mathrm{b}_{t+1:t+h} - x^\mathrm{b}_{t+1:t+h}\right)^2. \tag{7}$$

The physics loss $\mathcal{L}_\mathrm{phy}$ is an L2 loss that compares the predicted mass temperatures vector with a target vector $\mathbf{T}^t_\mathrm{m}$ obtained by applying Eq. (5). To better understand the training process of the encoder, we introduce the following notations:

- We define $\hat{\mathbf{T}}^t_\mathrm{m} \in \mathbb{R}^h$ as the vector containing the h autoregressive predictions of the mass temperatures when starting the prediction from the time-step t (i.e., the hidden states $z_{t:t+h}$);
- We define $\mathbf{T}^t_\mathrm{r} \in \mathbb{R}^h$ as the vector containing the h actual measured room temperatures, starting from the time-step t.

The target value for the encoder training when starting from the time-step t is then defined by applying Eq. (5) as:

$$\mathbf{T}^t_\mathrm{m} = \hat{\mathbf{T}}^{t-1}_\mathrm{m} + \frac{\Delta_t}{C_\mathrm{m} R_\mathrm{rm}} \left(\mathbf{T}^{t-1}_\mathrm{r} - \hat{\mathbf{T}}^{t-1}_\mathrm{m}\right), \tag{8}$$

where Δ_t is the duration of a time-step. The loss function to be minimized will then be:

$$\mathcal{L}_\mathrm{phys} \doteq \sum_{t=1}^{N} \left(\hat{\mathbf{T}}^t_\mathrm{m} - \mathbf{T}^t_\mathrm{m}\right)^2. \tag{9}$$

To study the impact of the PiNN component, we will also perform a black-box ablation that omits $\mathcal{L}_\mathrm{phys}$ (see Section 5.1). More details about the neural networks hyperparameters are in Appendix A.

3.3. Control-oriented modeling

To approach the control power described in Section 3.1, we will adopt an MCTS technique that requires a deterministic MDP with a discrete action space. In particular, our MCTS agent will rely on a deterministic, fully observable MDP (FOMDP) that approximates the stochastic partially observable MDP (POMDP) that we introduced in Section 3.1. We consider a FOMDP with deterministic dynamics described by the PhysNet introduced in Section 3.2, i.e., the exogenous information x^e_t gets forecasted for a fixed horizon h, thus enabling MCTS to deterministically simulate a state trajectory based on chosen (heating) actions.

To differentiate it from the real household POMDP, we indicate this approximated environment with a different notation: $(\tilde{\mathrm{X}}, \tilde{\mathrm{U}}, \tilde{f}, \tilde{\rho})$. The state at a certain time-step t is a continuous vector defined as:

$$\tilde{\mathbf{x}}_t \doteq (\tau_t,\, T_{\mathrm{r},t},\, T_{\mathrm{m},t}) \in \tilde{\mathrm{X}} \tag{10}$$

where: $\tau_t \in [0, 24[$ is the time of the day, $T_{\mathrm{r},t} \in \mathbb{R}$ is the room temperature, and $T_{\mathrm{m},t} \in \mathbb{R}$ is the (estimated) mass temperature of the building (as predicted by the PiNN). The action space is discrete and single-dimensional $\tilde{\mathrm{U}} \subset \mathrm{U}$, representing the heating system control action. The transition function $\tilde{f}$ is a trained PiNN as described in Section 3.2. Last, the reward function $\tilde{\rho}$ is defined, similarly to the original one introduced in Eq. (4), as:

$$\tilde{\rho}(\tilde{x}_t, \tilde{u}_t) \doteq \underbrace{-u^\mathrm{phys}_t \lambda_t}_{\text{Cost optimization}} \underbrace{-\left(T_{\mathrm{r\_set},t} - T_{\mathrm{r},t+1}\right)^+ c_1 - \left(T_{\mathrm{r},t+1} - T_{\mathrm{r\_set},t}\right)^+ c_2}_{\text{Thermal Comfort optimization}} \tag{11}$$

This structure provides the capability to simulate future states — which approximate the real dynamics through using a PiNN — enabling the use of MCTS.

3.4. Monte Carlo Tree Search (MCTS)

In MCTS, the possible scenarios that the current state may evolve into are represented in a tree structure, modeling subsequent states of the environment as nodes and the actions governing transitions between them as edges. We will consider deterministic environments with the assumption that a generative model of the MDP is available (i.e., the environment can be simulated), as explained in Section 3.3. As introduced there, we will refer to the state space of the environment as $\tilde{\mathrm{X}}$, the action space as $\tilde{\mathrm{U}}$, the (deterministic) transition function as $\tilde{f} : \tilde{\mathrm{X}} \times \tilde{\mathrm{U}} \rightarrow \tilde{\mathrm{X}}$ and the reward function as $\tilde{\rho} : \tilde{\mathrm{X}} \times \tilde{\mathrm{U}} \rightarrow \mathbb{R}$.

The ultimate goal of MCTS (or any tree search algorithm) is to enable to search for the most promising action to take from each system state, i.e., the one that allows to reach maximal reward. Thus, it will attach a "value" to each node — usually formalized as a Q-value function — representing how rewarding it is to move to that state, and do so based on simulation results. The general structure of the MCTS algorithm comprises four sequential phases that are repeated iteratively, until an acceptable solution is obtained (i.e., the node "values" are deemed to be representative for making good decisions):

1. **Selection:** select a node (i.e., a system state) to further explore actions for.
2. **Expansion:** expand the tree by adding new node(s), i.e., roll out possible actions from that state.
3. **Simulation:** evaluate the node value by performing Monte Carlo simulations.
4. **Backpropagation:** propagate the information acquired back to the root node.

After iterating through these phases ("searching"), the tree is used at inference time to select the actions based on the node values they lead to. For selection strategies like UCT, it has been proven that the search converges to the optimal solution [40]. Recent research has advanced MCTS performance by changing the UCT score formula [49] and adding NNs for value and policy estimation in the tree search [8,10,46,47].

In our experiments, we will particularly adopt improvements as proposed in the work of [8,47]. Following [47], we avoid random rollouts by Monte Carlo simulations, but rather use a trained NN to approximate the Q-values based on more targeted exploration of the tree. This implies that the Simulation step in the original MCTS algorithm above can be omitted. Thus, our *Vanilla MCTS* algorithm comprises the following steps:

1. **Selection:** Starting from a root node $\tilde{x}^0 \in \tilde{\mathrm{X}}$, the current tree is traversed by selecting the action $\tilde{u}^k \in \tilde{\mathrm{U}}^k, k \in \mathbb{N}$, until a leaf node $\tilde{x}^\ell$ is reached. Note that we will restrict the possible actions $\tilde{\mathrm{U}}^k \subseteq \tilde{\mathrm{U}}$ from a given state $\tilde{u}^k$, based on constraints from our environment (e.g., no heating possible if the temperature exceeds the user setpoint beyond a given tolerance). The selection of each action in this phase is based on[2]:

$$\tilde{u}^k = \arg\max_{\tilde{u} \in \tilde{\mathrm{U}}^k} \left\{ \underbrace{\tilde{Q}(\tilde{x}^k, \tilde{u})}_{\text{Value score}} + \alpha \underbrace{\frac{\sqrt{N(\tilde{x}^k)}}{1 + N(\tilde{x}^k, \tilde{u})}}_{\text{Exploration score}} \right\} , \tag{12}$$

$\forall k \in 0, 1, \ldots, \ell - 1$

where $N(\tilde{x})$ is the number of times the node $\tilde{x}$ has been reached so far, $N(\tilde{x}, \tilde{u})$ is the number of times the action $\tilde{u}$ has been selected, and $\tilde{Q}(\tilde{x}, \tilde{u})$ is a state-action value function that estimates the expected future average rewards obtained by executing the action $\tilde{u}$ from state $\tilde{x}$. (After this selection, the $N(\cdot)$ values along the followed path will be updated accordingly.) The balance between exploration and exploitation can be regulated by the hyperparameter $\alpha > 0$. In our experiments, we set $\alpha = 1$, unless stated otherwise.

2. **Expansion:** Upon reaching a leaf node $\tilde{x}^\ell$, if its depth does not exceed a fixed maximum value, it is expanded with new nodes (thus adding to the tree constructed so far), one for each possible action $\tilde{u} \in \tilde{\mathrm{U}}^\ell$.

3. **Backpropagation:** After the expansion phase, reward information gets backpropagated to the Q-values along the selected trajectory, from the original leaf node $\tilde{x}^\ell$ upwards to the root node, as follows[3]:

$$\tilde{Q}(\tilde{x}^k, \tilde{u}^k) = \frac{N(\tilde{x}^k, \tilde{u}^k)\tilde{Q}(\tilde{x}^k, \tilde{u}^k) + \frac{G^{k+1}}{(\ell - k)}}{N(\tilde{x}^k, \tilde{u}^k) + 1} , \tag{13}$$

$\forall k = \ell - 1, \ldots, 0$.

where $\tilde{Q}(\tilde{x}^k, \tilde{x}^k)$ is initialized as $\tilde{\rho}(\tilde{x}^k, \tilde{u}^k)$ and G^k is the discounted accumulated reward defined as:

$$\begin{cases} G^\ell = \tilde{\rho}(\tilde{x}^{l-1}, \tilde{u}^{\ell-1}) \\ G^k = \tilde{\rho}(\tilde{x}^{k-1}, \tilde{u}^{k-1}) + \gamma\, G^{k+1} \ , \quad \forall k = \ell - 1, \ldots, 1 \end{cases}$$

where γ is the discount value.

The rewards $\tilde{\rho}(\tilde{x}, \tilde{u})$ considered in this algorithm are normalized between 0 and 1, ensuring that every $\tilde{Q}$ value in Eq. (13) is also contained in $[0, 1]$. At the end of iterating through these phases, the resulting tree is used to choose the most selected[4] action $\tilde{u}$ from the current system state $\tilde{x}$ (i.e., the root of the tree that was built).

To further evaluate MCTS techniques in the BEM domain, we also deployed a more advanced version that uses a NN to lead the tree search through more optimal nodes, similarly to what was proposed in AlphaZero [47]. Our resulting *AlphaZero MCTS* algorithm iterates over the same selection/expansion/backpropagation phases, but modifies the Selection phase (step 1) by changing Eq. (12) to:

$$\tilde{u}^k = \arg\max_{\tilde{u} \in \tilde{\mathrm{U}}^k} \left\{ \underbrace{\tilde{Q}(\tilde{x}^k, \tilde{u})}_{\text{Value score}} + P(\tilde{x}^k, \tilde{u})\ \alpha \underbrace{\frac{\sqrt{N(\tilde{x}^k)}}{1 + N(\tilde{x}^k, \tilde{u})}}_{\text{Exploration score}} \right\} , \tag{14}$$

$\forall k \in 0, 1, \ldots, \ell - 1$.

Here $P(\tilde{x}, \tilde{u})$ is a prior probability value associated with taking action $\tilde{u}$ from state $\tilde{x}$. This probability will be learned by a NN (with $\tilde{x}$ as inputs and a probability distribution over $\tilde{u} \in \tilde{U}$ as output). This NN is trained offline by using sampled trajectories obtained with the Vanilla MCTS algorithm, considering the FOMDP simulator for both the search and evaluation environment. Details on the NN structure are listed in Appendix A. For balancing exploration/exploitation, when using AlphaZero MCTS, we set $\alpha = 3.5$.[5]

4. Experiment setup

4.1. Building model

As a building model representative of real-world conditions, we used the BOPTEST benchmark framework developed by Blum et al. [50]. To efficiently integrate BOPTEST with our python-based implementation of the aforementioned PiNN and MCTS components, we used the OpenAI Gym toolkit BOPTEST-Gym [51]. The specific model we used is their single-zone Hydronic Heat Pump simulation, representing a five-member residential dwelling located in Brussels, Belgium. The single-zone model considers a $12 \times 16\,\mathrm{m}^2$ rectangular floor, heated with an air-to-water modulating heat pump of 15 kW nominal heating capacity. The action space consists of the modulation signal of the heat pump, $\mathrm{U} \doteq [0, 1]$.

Because the MCTS algorithm assumes discrete actions, we consider 5 equally spaced actions: $\tilde{\mathrm{U}} \subset \mathrm{U} \doteq [0, 1]$. To analyze our MCTS algorithm's potential, we consider two different price scenarios: (i) one with real-world day-ahead price data from BELPEX of 2019,[6] and (ii) a synthetic one using a square wave profile.

4.2. Constraints

An advantage of MCTS is that it can easily incorporate dynamic (i.e., state-dependent) constraints on the possible actions (cf. our reduced action space $\tilde{\mathrm{U}}^k \subseteq \tilde{\mathrm{U}}$ from a current state $\tilde{x}^k$), which amounts to pruning action edges from the tree. Such backup constraints can be picked with total freedom based on the available state values, allowing for rather complex constraints, beyond simple clipping. This enables more efficient constraint management compared to classic RL agents, where backup actions are enforced a posteriori (i.e., correcting the chosen RL actions if need be), without the RL controller being aware of them.

As an example of such constraints (which could be easily adapted), we will specifically assume a backup controller that forces the heat pump to activate (respectively turn off) when the room temperature falls below the desired one by Δ_{T}^- (respectively exceeds it by Δ_{T}^+).

4.3. Validation objectives and metrics

Our experiments discussed in the next section have as primary objectives to evaluate the performance of both (1) the system state forecasting component, i.e., our PhysNet extension (from Section 3.2), and (2) the MCTS-based controller (from Section 3.4).

To evaluate the PhysNet forecasting ability, we used BOPTEST-Gym to generate the relevant data time series at 30 min resolution by simulating the heat pump usage with a continuous controller (details in

[2] Note that we omitted a portion of the formula used in [8]. Specifically, we removed $\log\left(\frac{N(\tilde{x}^k)+c+1}{c}\right)$ as it would become null when $c \gg N(\tilde{x}^k)$, which is the case when using the values suggested in the original work.

[3] Note that compared to [8], we added the $\ell - k$ division of the G^{k+1} term. This leads to normalized Q-values that are contained in $[0, 1]$.

[4] Or, equivalently, the one with the highest $\tilde{Q}(\tilde{x}, \tilde{u})$ value.

[5] Note that the higher α in AlphaZero MCTS as opposed to Vanilla MCTS stems from the additional multiplication with $P(\tilde{x}, \tilde{u}) \leq 1$.

[6] The price data used is provided by BOPTEST [50] as the "higly dynamical electricity price" scenario, see https://ibpsa.github.io/project1-boptest/testcases/ibpsa/testcases_ibpsa_bestest_hydronic_heat_pump/.

Table 1
Description of the data used.

Symbol	Description	Value domain
τ	Time of the day [h]	$\{0, 0.5, 1, \ldots, 23.5\}$
T_r	Indoor temperature [°C]	[15, 25]
T_a	Outside temperature measured in a dry bulb [°C]	[−10, 20]
u	Heat pump modulating signal for the compressor speed	[0, 1]
u^{phys}	Heat pump electrical power consumed [W]	[0, 4000]
λ	Belgian day-ahead energy prices as determined by the BELPEX electricity market [€/kWh]	[−0.4, 0.4]

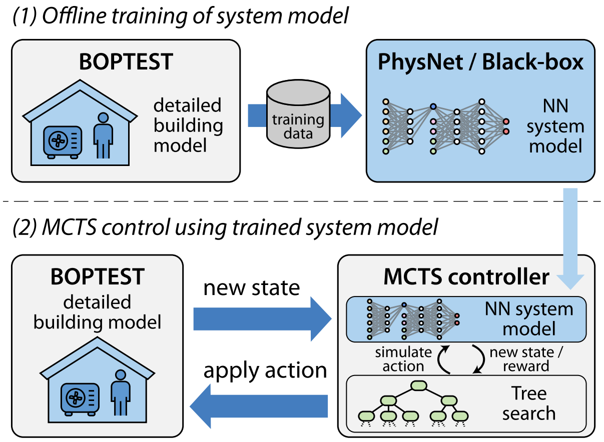

Fig. 3. Overview of our proposed MCTS control strategy. The NN-based system model is first trained with historical data. It is then used in MCTS for simulating action roll-outs, from which the tree search algorithm eventually decides on its optimal action to take, which is then applied to the actual system (in BOPTEST). These steps are repeated as needed.

Appendix B) starting from January 1, 2019. The parameters that we record the time series for are listed in Table 1. We add cumulative noise to the outside temperature to emulate the prediction error of a weather forecaster tool (see Appendix C for more details). To assess the benefits of infusing physical knowledge into the model, we benchmark it against a black-box ablation that shares the same architecture but omits the physical loss defined in Eq. (9). As a performance metric, we use the Mean Absolute Error (MAE) between model predictions and recorded values.

Besides evaluating the predictive performance of PhysNet, we also aim to evaluate the benefit it brings (compared to a black-box baseline) when adopted in our proposed MCTS controller. To train the system state transition models (PhysNet and black-box), we simulate the selected building environment with BOPTEST-Gym for 10 days, using a discrete rule-based controller (details in Appendix B) to take the heating actions. We then use our trained networks (PhysNet or black-box) for simulating system state transitions in building the trees in the MCTS algorithms described in Section 3.4, starting from the current state in BOPTEST-Gym. Based on the tree built, the action to apply is chosen, executed in BOPTEST-Gym, which then provides the actual state at the following decision timepoint. For that new state, a new MCTS tree is built, and the process repeats. The PhysNet/black-box system model's NNs are retrained after each day of evaluation, by adding the newly obtained data (with the MCTS controller's actions) to the initial 10-day training set. We consider a test period of 11 consecutive days, following the 10 initial ones. Fig. 3 illustrates this setup.

To quantitatively evaluate our controller's performance, we use the average cumulative reward (defined in Eq. (11)) obtained from the environment over the course of a day: higher rewards indicate better adherence of room temperatures to desired levels, achieved while minimizing costs. We normalize this reward per timestep to [0,1], resulting in a maximal reward of 48 over a single day, given that we use 30 min timesteps. We provide full details about the normalization process in Appendix D. To further benchmark our MCTS controllers' results, we also consider a price-agnostic rule-based controller, namely a bang-bang controller that naively heats the room whenever its temperature falls below the desired one (for details see Appendix B).

5. Results

As explained in Section 4.3, our experimental results will subsequently answer the following research questions:

Q1: Is the PhysNet model more *accurate and data-efficient* than a black-box model? (Section 5.1)
Q2: Is the PhysNet model able to provide *better control actions* when used in a Vanilla MCTS algorithm, compared to a black-box model? (Section 5.2)
Q3: What computational benefits arise from a more targeted tree search exploration, i.e., using AlphaZero MCTS instead of Vanilla MCTS? (Section 5.3)

5.1. Forecasting results

We first compare the forecasting performance of the PiNN model to its black-box counterpart. Training sets of 2, 5, and 24 days were generated with the BOPTEST framework, each for prediction horizons of 3, 6, and 12 hours. As indicated in Section 4.3, we adopt a continuous controller (see Appendix B) to decide on heating actions in these scenarios. We evaluate each trained model with a test set spanning 6 days. To allow robust comparison, we average results over 100 runs (varying only the random initialization seed) for each model. The Mean Absolute Error (MAE) results are shown in Fig. 4, where the scatter plot represents the median value and interquartile range for each model combination (architecture + training length + prediction horizon).

We first focus on a small training dataset (i.e., the first column of Fig. 4, with 2 days of training data). Regarding the temperature predictions (top row), we note a substantial improvement of the PhysNet compared to its black-box ablation. For each prediction horizon, PhysNet achieves lower prediction error (with an average reduction of 32%) as well as a more stable performance (i.e., the interquartile bands are smaller) compared to the black-box model. Looking at the energy consumption predictions (bottom row of graphs in Fig. 4), PhysNet still outperforms the black-box model for higher prediction horizons, but with larger performance variation (cf. larger interquartile whiskers) and less pronounced MAE benefits compared to the black-box model (10% reduction on average). Given the PhysNet architecture, these results are in line with our expectations. The PhysNet addition consists of physics prior knowledge regarding the bulk temperature of the building, which mostly defines the thermal exchange of the room with the structure's mass. These exchanges mostly influence the room temperature, and not the energy consumed by the heater component. For this reason, we did expect a more pronounced improvement of the PhysNet in the temperature predictions compared to the energy consumption ones.

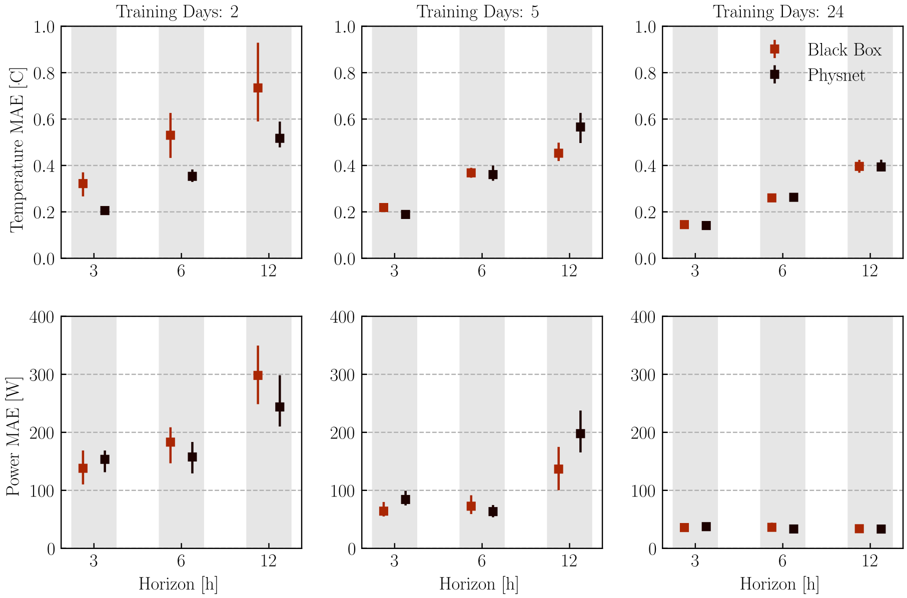

Fig. 4. Prediction results of the PhysNet architecture and its black-box counterpart with different training sizes and prediction horizons. PhysNet's temperature predictions outperform the black-box ones when a small dataset (2 days) is considered. The performance of the two architectures converges to the same prediction errors when a bigger training set (24 days) is used.

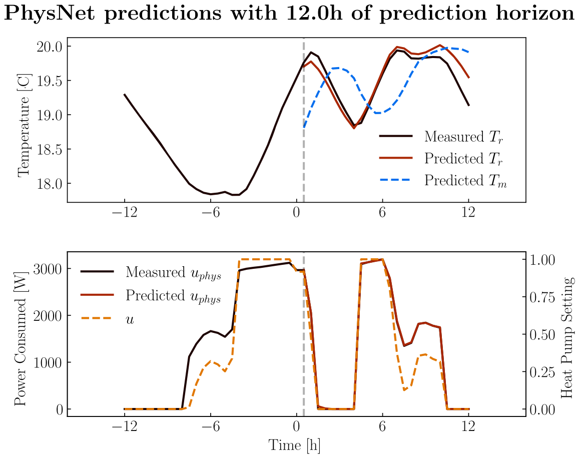

Fig. 5. PiNN predictions over a horizon of 12 hours. The temperature graph on top illustrates the expected behavior of the building mass as thermal storage that exchanges heat with the room at a slower pace, i.e., T_m lags behind T_r.

When increasing the training data to 5 days, the black-box models achieve a noticeable MAE reduction compared to when a smaller training set is considered. Conversely, PhysNet predictions do not benefit from a similar improvement, especially on the temperature side, where the MAE for a 5-day training set is on par with the 2-day based model. Finally, when increasing the training data to 24 days, both models converge to the same results with very high stability in both temperature and energy consumption predictions. This result outlines that — when solely considering the forecasting metric (i.e., MAE) — these two models achieve equivalent results when fed with a big enough training dataset. Nonetheless, with a small amount of data, PhysNet seems to outperform the black-box model, especially with regard to the temperature forecasts.

Finally, we wanted to validate the mass temperature estimation of PhysNet. Yet, since building mass temperature cannot be easily measured in practice, we provide qualitative rather than quantitative evaluation. To verify that the predicted mass temperature T_m behaves as expected, Fig. 5 presents the predicted quantities over a horizon of 12 h. We observe that the predicted T_m varies with a higher delay compared to T_r, consistent with the expected behavior of a building mass's greater inertia.

Through this first experiment, we established the value of our models, numerically evaluating their performance as forecasters. We also highlighted the first benefits of the PiNN model compared to the black-box one, namely its higher data efficiency and its hidden state generation that correctly resembles an insightful and hard-to-measure physical value of the building.

5.2. PiNN control benefits

We now focus on the hypothesized benefit of using a PiNN-based system model within an MCTS controller. We plug the PhysNet (and its black-box reference model) into our Vanilla MCTS controller, which we evaluate at different levels of computational complexity: we vary the number of times we cycle through the selection/expansion/backpropagation steps. This number of iterations, which affects the eventual MCTS tree depth, is commonly referred to as 'number of simulations'. The resulting controller performance is shown in Fig. 6 (daily reward) and Fig. 7 (monetary cost and temperature setpoint deviation).

We first focus on the differences between the MCTS application with a PhysNet and a black-box model. We observe in Fig. 6 that Physnet enables the algorithm to obtain significantly higher daily rewards (in the range of +3%) compared to the black-box model, and consistently does so when increasing the number of simulations to construct the MCTS tree. On the latter, we observe that performance improves logarithmically with increasing number of simulations. This suggests that the main control behavior is learned relatively fast, but further improvements are achieved through deeper tree rollouts. Intuitively, it makes sense that expanding that forward-looking horizon very far only brings limited benefit (hence the diminishing benefit in daily reward).

Given that the prediction performance differences observed in Section 5.1 between black-box and PhysNet models seemed limited, it may be rather surprising that when deployed in an MCTS loop they lead to significantly different results. We hypothesize that this is due to the higher physical consistency of the PhysNet model, which allows it to better predict the system behavior for control actions that are "out-of-distribution" with respect to the training data (as observed in Di Natale

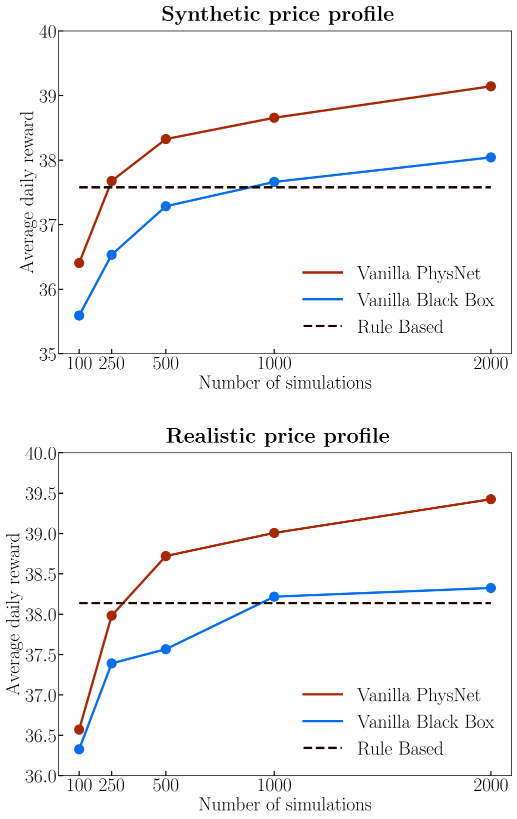

Fig. 6. Average daily rewards obtained by the Vanilla MCTS when applied to 11 test days with a PhysNet and a black-box simulator. The performance obtained with the PhysNet model remarkably outperforms the one obtained with the black-box simulator.

et al. [6]). Indeed, in building the MCTS tree, we need accurate predictions for a wide range of actions, e.g., heating when it's already hot, or conversely not heating when the temperature gets too low. Such scenarios will not be represented in the training set for the system model, and PhysNet can rely on its "physical knowledge" to better predict the corresponding state trajectories than a purely data-driven black-box counterpart.

To contextualize the general performance of our MCTS solution, we compare it against the naive rule-based (bang-bang) controller that simply heats whenever the temperature drops below the setpoint. We note that the black-box controller needs MTCS to use 1,000 simulations to be able to beat bang-bang, whereas the PhysNet-based MCTS already matches it at ∼250 simulations and significantly outperforms it for more. When looking at the actual system parameters of interest to the user (i.e., monetary cost and effective room temperature; both reflected jointly in the reward function Eq. (11)), as plotted in Fig. 7 for the synthetic price profile, we note qualitatively similar performance benefits in comparing PhysNet MCTS controllers to the black-box counterpart (average cost reduction of 4% and temperature deviation reduction of 7% for PhysNet), as well as compared to the rule-based baseline.

Finally, to achieve more intuitive insight into how our best control policy of MCTS+PhysNet behaves, Fig. 8 compares its decisions over a randomly selected test day with the rule-based baseline. We note that the rule-based controller tends to over-heat the room (e.g., at times ∼10:00 and ∼20:00), obviously without any consideration of the energy prices. Conversely, MCTS achieves to better align the room temperature with its desired setpoint. It also manages to exploit low-price points by pre-heating the room (e.g., around ∼6:00 and ∼16:00). Consequently, MCTS achieves higher rewards, as discussed previously.

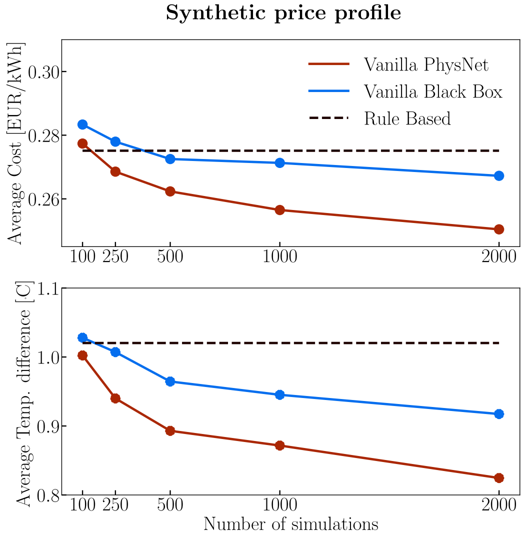

Fig. 7. Average cost per kWh for powering the heat pump following the controller's heating actions (first row) and average of the absolute value of the room temperature deviation from the desired one (second row). Conforming with the results shown in Fig. 6, the MCTS controllers achieve lower costs and more room temperature adherence to the desired one.

5.3. Efficient tree search with AlphaZero MCTS

A drawback of the Vanilla MCTS algorithm is its high computational cost to take a control action, stemming from simulating many rollouts. As has been shown for problems of game playing [8,47], the addition of prior knowledge into the tree search can increase its performance in terms of managing to achieve higher rewards using fewer simulations. As an example of such an improved MCTS algorithm, we implemented the AlphaZero MCTS algorithm (based on [46], see Section 3.4). To quantify the computational cost required by MCTS to obtain each action, we use the number of simulations as a metric (cf. a higher number of simulations implies a higher computational cost). The NN used for the prior probability $P(\tilde{x}^k, \tilde{u})$ in Eq. (14) is trained offline at the end of each day by using simulated samples obtained with data from previous days. These samples are obtained by using the Vanilla MCTS algorithm and by searching and evaluating on the same simulated environment. Since that training is performed offline, it can be done at times where a smart controller is typically not required (e.g., at night). Given the offline nature, in collecting the training sample trajectories, we may use a high number of simulations for the Vanilla MCTS to guide them, and thus obtain high-quality experience samples to train the prior-policy NN.

We compare the results of the Vanilla MCTS algorithm with AlphaZero MCTS, using the same approach as Section 5.2. Fig. 9 shows the average daily reward obtained for a varying number of simulations. We adopt the PhysNet model for these simulations in MCTS, since we previously assessed its performance to be superior to its black-box counterpart. When a low number of simulations is performed, AlphaZero achieves a notably higher reward (2% higher on average for ≤500 simulations), both for the synthetic and the realistic price scenarios compared to the Vanilla version. When increasing the number of MCTS simulations, this benefit continues to hold but diminishes until AlphaZero and Vanilla MCTS ultimately converge (since this implies the large number of unrolled tree scenarios for Vanilla MCTS also suffice to make a (near-)optimal decision). From a practical perspective, we conclude that for real-world applications the AlphaZero MCTS controller is to be preferred since it efficiently finds valid control actions with a low number of simulations.

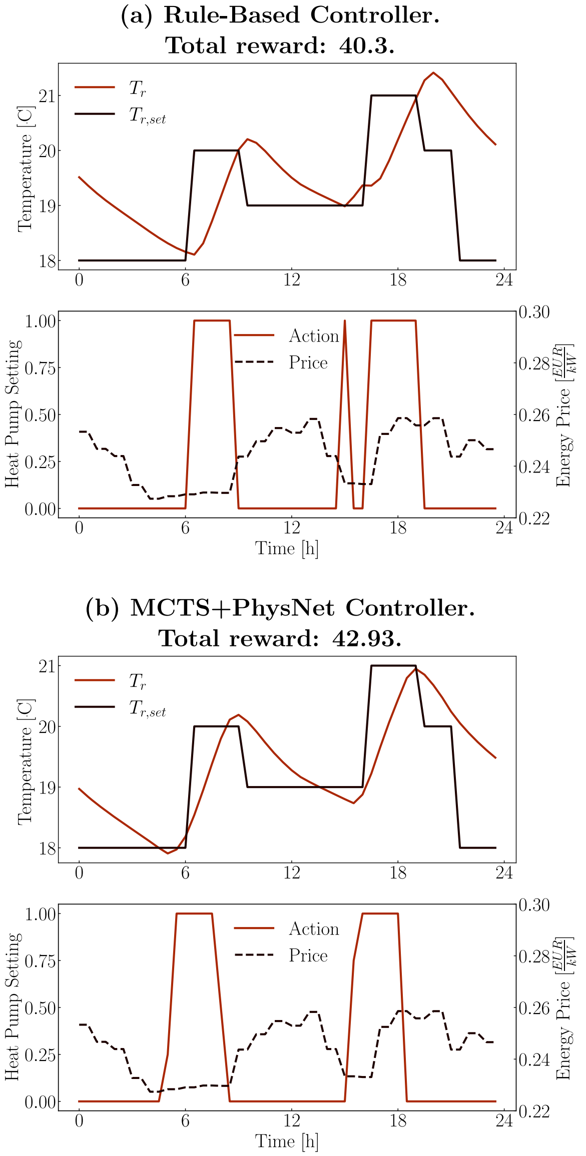

Fig. 8. Action sequence selected by (a) the rule-based bang-bang policy and (b) the PhysNet-based Vanilla MCTS respectively, for the same randomly selected day. The MCTS policy manages to stay closer to the desired temperature, while heating the room at times with lower prices, thus attaining a higher reward compared to the rule-based controller.

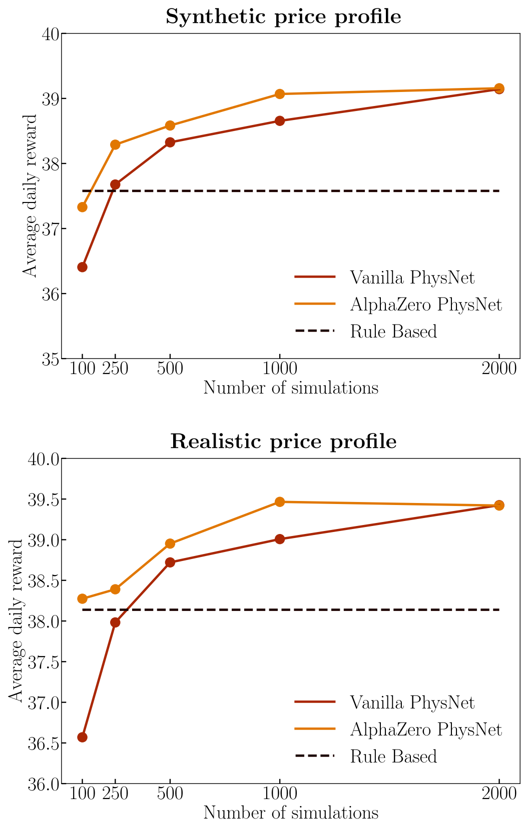

Fig. 9. Average daily rewards obtained by the Vanilla MCTS and AlphaZero MCTS when applied to 11 test days with a PhysNet simulator. The addition of a prior-policy NN (i.e., the AlphaZero algorithm) achieves a substantial increment in rewards when a low number of simulations is performed.

6. Final discussion

6.1. Conclusions

In this paper, we presented an MCTS algorithm with a PiNN applied to a residential heating Demand Response problem. We first deployed and expanded PhysNet, a previously proposed PiNN, allowing it to produce multi-time-step predictions of room temperature and energy consumption of a heat pump. After establishing PhysNets predictive performance (32% lower MAE prediction error in a low training data regime, compared to a black-box model), we use it as a simulator for the MCTS control algorithm. Compared to a rule-based controller baseline, we noted a substantial improvement of a Vanilla MCTS controller using PhysNet in terms of the considered reward function (up to 9% reduction in average cost and 19% increase in thermal comfort). Subsequently, we showed how adopting the more advanced AlphaZero MCTS helps to achieve such benefits at lower computational cost (i.e., fewer tree rollout simulations), by incorporating a policy NN to guide action explorations more efficiently.

Thanks to its flexibility to incorporate state-dependent constraints on the allowed actions, as well as its planning nature, we believe MCTS is a valid technique for Demand Response control of a residential heater. Moreover, compared to conventional RL approaches, MCTS offers more interpretable control actions (e.g., one could inspect the various rollouts in the tree constructed to compare effects of other possible action trajectories). Such higher degree of interpretability (and the failure case analysis it enables) could spur higher levels of trust and adoption in real-world commercial settings. Conceptually, MCTS creates a bridge between model-based techniques (such as MPC) and model-free data-driven ones (such as RL), exploiting benefits from both (interpretability from MPC, efficient and data-driven learning from RL).

We hope that our exploration of MCTS (with our contributions of incorporating PiNN and analysis of AlphaZero-like policy network) spurs further research specifically on its application in residential energy control algorithms. Next we list current limitations (Section 6.2) and directions for such future work (Section 6.3) in this area.

6.2. Strengths and limitations

Although our methods provide several benefits, different limitations need to be taken into consideration when applied in real-world settings. Regarding the application of PhysNet, our proposed PiNN model, we showed the benefits of infusing physical knowledge, both for forecasting and control performance. PhysNet relied on estimating an interpretable latent state variable, i.e., the building mass temperature. However, this was achieved by a (simplified) RC model of the building, which requires proper initialization to work properly. The latter implies that in practice, it would call for numerical optimization or empirical estimation, which incurs a potential issue in deploying it at scale across a multitude of households. Furthermore, we note that the estimated hidden state may not accurately represent the actual building mass temperature (cf. the training objective only included room temperature prediction accuracy) — thus it cannot be reliably used as an estimation thereof.

Regarding MCTS, a core characteristic is its reliance on (reasonably accurate) simulation of action sequences and their effects on the system state. On the plus side, such explicit rollouts allow to naturally account for (possibly state-dependent) constraints, since impermissible actions can be easily pruned from the tree. Further, the constructed tree facilitates interpretability (see before, Section 6.1). Yet, a main drawback of MCTS is its computational cost at inference time, since the action decision is based on building the tree dynamically from the current system state onwards. Given that building the tree is based on Monte Carlo simulations, to make (near-)optimal decisions a potentially large number of simulations is required. The latter can be prohibitive in residential applications such as the considered heat pump case, either or both because of limited computational requirements or real-time nature (e.g., if the control timestep would be far lower than the currently considered 30 min intervals). Although more efficient tree expansion strategies (cf. the AlphaZero approach we applied) partially address this, real-world deployment may require further improvements.

6.3. Future work

Following our study offering (some of) the first steps into MCTS solutions for residential building energy management systems (BEMS), further research is needed to push the technology towards large-scale real-world adoption. Although our results already confirmed the potential of MCTS in BEMS, a comprehensive benchmarking against competing state-of-the-art control algorithms (including both MPC and RL approaches) in terms of control performance, computational cost, and scalability should be carried out to robustly establish MCTS's competitiveness. Further refining of the MCTS algorithm based on recent evolutions in the field (e.g., MuZero [8]) is another direction of future work. These innovations could further improve the technique's performance by, for example, merging the modeling and planning tasks in a single solution within the tree search. Besides methodological improvements, the exploration of MCTS in terms of application scenarios is a valuable research direction. For example, in scenarios with non-stationary constraints — e.g., they strongly vary dynamically over time — and/or more complex ones than what we considered in this paper, MCTS-like solutions could be particularly beneficial. Finally, actual field trials with real-world deployment of MCTS controllers should be carried out to establish their practical applicability.

Acknowledgments and funding

This work was supported in part by the European Union's Horizon 2020 research and innovation programme under the projects BRIGHT (grant agreement no. 957816) and BIGG (grant agreement no. 957047).

We also thank Marie-Sophie Verwee for her technical support in the deployment of our work.

CRediT authorship contribution statement

Fabio Pavirani: Writing – original draft, Conceptualization, Formal analysis, Investigation, Methodology, Software, Visualization. **Gargya Gokhale:** Writing – review & editing, Software, Conceptualization. **Bert Claessens:** Supervision, Conceptualization, Writing – review & editing. **Chris Develder:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors are unable or have chosen not to specify which data has been used.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT 3.5 in order to enhance the language quality of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

Appendix A. Hyperparameters

The PhysNet network is composed of two separate Feed Forward Neural Networks (FFNNs), the encoder and the predictor. The encoder has a single hidden layer composed of 32 neurons activated with a ReLU function. Its output layer is activated with a tanh function. The predictor has a single hidden layer composed of 64 neurons activated with a ReLU function. The output of the predictor layer has two different activation functions: tanh for the temperature, and ReLU for the energy consumption. The past observations depth value d is fixed to 24 timesteps (12 hours). All the neural network parameters are trained with an Adam optimizer. The prior policy neural network $p(\tilde{x})$ of the AlphaZero algorithm is a FFNN with 2 hidden layers of 64 and 32 neurons activated with a ReLU function. Its final output gets activated through a softmax function and the whole network gets trained with a Cross-Entropy loss and an Adam optimizer. All the neural networks used in this paper were implemented using PyTorch 2.0 [52].

Appendix B. Rule based controllers

The rule-based bang-bang controller referred to in this work heats the room whenever its temperature drops below the desired one. Mathematically, its policy is defined as:

$$u_t = \begin{cases} 1 & \text{if } T_{\mathrm{r},t} < T_{\mathrm{r\_set},t} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

The discrete rule-based controller is defined as:

$$u_t = \begin{cases} 0 & \text{if } T_{\mathrm{r},t} > T_{\mathrm{r\_set},t} \\ 0.25 & \text{else if } T_{\mathrm{r},t} > T_{\mathrm{r\_set},t} - 0.05 \\ 0.5 & \text{else if } T_{\mathrm{r},t} > T_{\mathrm{r\_set},t} - 0.15 \\ 0.75 & \text{else if } T_{\mathrm{r},t} > T_{\mathrm{r\_set},t} - 0.25 \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

Finally, the continuous controller used to generate data in Section 5.1 is defined as:

$$u_t = \min\left(2(T_{\mathrm{r\_set},t} - T_{\mathrm{r},t})^{+},\ 1\right) \tag{17}$$

Appendix C. Cumulative noise

To evaluate our experiments more realistically, we added random noise to the future outside temperature. The added noise will simulate the prediction error of a realistic forecaster tool.

For each time-step, we sample from a Bernoulli distribution to determine the sign of the added noise (positive or negative). We then sampled and added random noise from a normal distribution (with the previously decided sign) to the outside temperature. The noise added gets cumulated while considering values more far into the future. Each addition get multiplied with a σ (or $-\sigma$) hyperparameter that determines the magnitude of the added noise

Appendix D. Normalization of the rewards

To use the rewards obtained from Eq. (4) in our MCTS algorithm, we scaled the values into the closed unit interval. To do so, we used the standard min-max equation:

$$\rho_{\text{norm}} \doteq \frac{\rho - \rho_{\min}}{\rho_{\max} - \rho_{\min}} \,, \tag{18}$$

where $\rho_{\text{norm}} \in [0,1]$ is the scaled value of $\rho \in [\rho_{\min}, \rho_{\max}]$. To use this formula, we need to define $\rho_{\min}, \rho_{\max}$ as domain bound for ρ. Given the reward formula (Eq. (11)), the upper bound is easily defined as $\rho_{\max} \doteq 0$, corresponding to the scenario where $u^{\text{phys}} = 0$ and $T_{\text{r}} = T_{\text{r\_set}}$. Defining the lower bound $\rho_{\min}$ requires instead an empirical choice, as in principle the rewards are not limited to a finite interval. To define $\rho_{\min}$, we picked the two lower bounds of each part of the reward function, namely the cost optimization and the thermal comfort optimization. For the cost minimum value, we considered the highest cost multiplied by the highest heat pump energy consumption measured in our train set. For the thermal comfort minimum, we picked the worst case as a difference between T_{r} and $T_{\text{r\_set}}$ of 2°C multiplied by c_1.

References

[1] Eurostat, Energy, Transport and Environment Statistics — 2020 Edition, 2020.

[2] P. Stoffel, L. Maier, A. Kümpel, T. Schreiber, D. Müller, Evaluation of advanced control strategies for building energy systems, Energy Build. 280 (2023) 112709.

[3] Z. Wang, T. Hong, Reinforcement learning for building controls: the opportunities and challenges, Appl. Energy 269 (2020) 115036.

[4] Z. Nagy, G. Henze, S. Dey, J. Arroyo, L. Helsen, X. Zhang, B. Chen, K. Amasyali, K. Kurte, A. Zamzam, et al., Ten questions concerning reinforcement learning for building energy management, Build. Environ. 110435 (2023).

[5] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.

[6] L. Di Natale, B. Svetozarevic, P. Heer, C.N. Jones, Physically consistent neural networks for building thermal modeling: theory and analysis, Appl. Energy 325 (2022) 119806.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199, 2013.

[8] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al., Mastering Atari, Go, chess and shogi by planning with a learned model, Nature 588 (2020) 604–609.

[9] G. Gokhale, B. Claessens, C. Develder, Physics informed neural networks for control oriented thermal modeling of buildings, Appl. Energy 314 (2022) 118852.

[10] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489.

[11] A. Boodi, K. Beddiar, M. Benamour, Y. Amirat, M. Benbouzid, Intelligent systems for building energy and occupant comfort optimization: a state of the art review and recommendations, Energies 11 (2018) 2604.

[12] Z. Afroz, G. Shafiullah, T. Urmee, G. Higgins, Modeling techniques used in building HVAC control systems: a review, Renew. Sustain. Energy Rev. 83 (2018) 64–84.

[13] R.Z. Homod, Review on the HVAC system modeling types and the shortcomings of their application, J. Energy 2013 (2013).

[14] A. Foucquier, S. Robert, F. Suard, L. Stéphan, A. Jay, State of the art in building modelling and energy performances prediction: a review, Renew. Sustain. Energy Rev. 23 (2013) 272–288.

[15] X. Li, J. Wen, Review of building energy modeling for control and operation, Renew. Sustain. Energy Rev. 37 (2014) 517–537.

[16] C. Deb, F. Zhang, J. Yang, S.E. Lee, K.W. Shah, A review on time series forecasting techniques for building energy consumption, Renew. Sustain. Energy Rev. 74 (2017) 902–924.

[17] M. Bourdeau, X. qiang Zhai, E. Nefzaoui, X. Guo, P. Chatellier, Modeling and forecasting building energy consumption: a review of data-driven techniques, Sustain. Cities Soc. 48 (2019) 101533.

[18] U. Ali, M.H. Shamsi, C. Hoare, E. Mangina, J. O'Donnell, Review of urban building energy modeling (UBEM) approaches, methods and tools using qualitative and quantitative analysis, Energy Build. 246 (2021) 111073.

[19] M. Wetter, W. Zuo, T.S. Nouidui, X. Pang, Modelica buildings library, J. Build. Perform. Simul. 7 (2014) 253–270.

[20] D.B. Crawley, L.K. Lawrie, F.C. Winkelmann, W.F. Buhl, Y.J. Huang, C.O. Pedersen, R.K. Strand, R.J. Liesen, D.E. Fisher, M.J. Witte, et al., EnergyPlus: creating a new-generation building energy simulation program, Energy Build. 33 (2001) 319–331.

[21] A. Afram, F. Janabi-Sharifi, Gray-box modeling and validation of residential HVAC system for control system design, Appl. Energy 137 (2015) 134–150.

[22] B. Matthiss, A. Azzam, J. Binder, Thermal building models for energy management systems, in: 2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), IEEE, 2023, pp. 1–6.

[23] P. Bacher, H. Madsen, Identifying suitable models for the heat dynamics of buildings, Energy Build. 43 (2011) 1511–1522.

[24] E. Vrettos, E.C. Kara, J. MacDonald, G. Andersson, D.S. Callaway, Experimental demonstration of frequency regulation by commercial buildings—part I: modeling and hierarchical control design, IEEE Trans. Smart Grid 9 (2016) 3213–3223.

[25] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[26] J. Drgoňa, A.R. Tuor, V. Chandan, D.L. Vrabie, Physics-constrained deep learning of multi-zone building thermal dynamics, Energy Build. 243 (2021) 110992.

[27] J. Drgoňa, J. Arroyo, I.C. Figueroa, D. Blum, K. Arendt, D. Kim, E.P. Ollé, J. Oravec, M. Wetter, D.L. Vrabie, et al., All you need to know about model predictive control for buildings, Annu. Rev. Control 50 (2020) 190–232.

[28] Y. Yao, D.K. Shekhar, State of the art review on model predictive control (mpc) in heating ventilation and air-conditioning (HVAC) field, Build. Environ. 200 (2021) 107952.

[29] A. Afram, F. Janabi-Sharifi, A.S. Fung, K. Raahemifar, Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: a state of the art review and case study of a residential HVAC system, Energy Build. 141 (2017) 96–113.

[30] Y.M. Lee, R. Horesh, L. Liberti, Optimal HVAC control as demand response with on-site energy storage and generation system, Energy Proc. 78 (2015) 2106–2111.

[31] J. Reynolds, Y. Rezgui, A. Kwan, S. Piriou, A zone-level, building energy optimisation combining an artificial neural network, a genetic algorithm, and model predictive control, Energy 151 (2018) 729–739.

[32] J. Chen, G. Augenbroe, X. Song, Lighted-weighted model predictive control for hybrid ventilation operation based on clusters of neural network models, Autom. Constr. 89 (2018) 250–265.

[33] M. Ławryńczuk, Computationally efficient model predictive control algorithms. A neural network approach, Stud. Syst. Decis. Control 3 (2014).

[34] S. Brandi, M.S. Piscitelli, M. Martellacci, A. Capozzoli, Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings, Energy Build. 224 (2020) 110225.

[35] T. Wei, Y. Wang, Q. Zhu, Deep reinforcement learning for building HVAC control, in: Proceedings of the 54th Annual Design Automation Conference 2017, 2017, pp. 1–6.

[36] C. Patyn, F. Ruelens, G. Deconinck, Comparing neural architectures for demand response through model-free reinforcement learning for heat pump control, in: 2018 IEEE International Energy Conference (ENERGYCON), IEEE, 2018, pp. 1–6.

[37] A. Nagy, H. Kazmi, F. Cheaib, J. Driesen, Deep reinforcement learning for optimal control of space heating, arXiv preprint arXiv:1805.03777, 2018.

[38] Z. Jiang, M.J. Risbeck, V. Ramamurti, S. Murugesan, J. Amores, C. Zhang, Y.M. Lee, K.H. Drees, Building HVAC control with reinforcement learning for reduction of energy cost and demand charge, Energy Build. 239 (2021) 110833.

[39] R. Coulom, Efficient selectivity and backup operators in Monte-Carlo tree search, in: International Conference on Computers and Games, Springer, 2006, pp. 72–83.

[40] L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in: European Conference on Machine Learning, Springer, 2006, pp. 282–293.

[41] T.K. Wijaya, T.G. Papaioannou, X. Liu, K. Aberer, Effective consumption scheduling for demand-side management in the smart grid using non-uniform participation rate, in: 2013 Sustainable Internet and ICT for Sustainability (SustainIT), IEEE, 2013, pp. 1–8.

[42] E. Galván-López, C. Harris, L. Trujillo, K. Rodriguez-Vazquez, S. Clarke, V. Cahill, Autonomous demand-side management system based on Monte Carlo tree search, in: 2014 IEEE International Energy Conference (ENERGYCON), IEEE, 2014, pp. 1263–1270.

[43] F. Golpayegani, I. Dusparic, S. Clarke, Collaborative, parallel Monte Carlo tree search for autonomous electricity demand management, in: 2015 Sustainable Internet and ICT for Sustainability (SustainIT), IEEE, 2015, pp. 1–8.

[44] F. Golpayegani, I. Dusparic, A. Taylor, S. Clarke, Multi-agent collaboration for conflict management in residential demand response, Comput. Commun. 96 (2016) 63–72.

[45] J. Kiljander, R. Sarala, J. Rehu, D. Pakkala, P. Pääkkönen, J. Takalo-Mattila, K. Känsälä, Intelligent consumer flexibility management with neural network-based planning and control, IEEE Access 9 (2021) 40755–40767.

[46] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (2017) 354–359.

[47] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, Science 362 (2018) 1140–1144.

[48] L. Busoniu, R. Babuska, B. De Schutter, D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, 2017.

[49] C.D. Rosin, Multi-armed bandits with episode context, Ann. Math. Artif. Intell. 61 (2011) 203–230.

[50] D. Blum, J. Arroyo, S. Huang, J. Drgoňa, F. Jorissen, H.T. Walnum, Y. Chen, K. Benne, D. Vrabie, M. Wetter, et al., Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings, J. Build. Perform. Simul. 14 (2021) 586–610.

[51] J. Arroyo, C. Manna, F. Spiessens, L. Helsen, An open-ai gym environment for the building optimization testing (BOPTEST) framework, in: Building Simulation 2021, IBPSA, 2021, pp. 175–182.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, pp. 8024–8035.