A toolchain for domestic heat-pump control using Uppaal Stratego

Riaz, Imran; Jensen, Peter Gjøl; Larsen, Kim Guldstrand; Srba, Jiri

Published in:
Science of Computer Programming

DOI (link to publication from Publisher):
10.1016/j.scico.2023.102987

Creative Commons License
CC BY 4.0

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

Link to publication from Aalborg University

Citation for published version (APA):
Riaz, I., Jensen, P. G., Larsen, K. G., & Srba, J. (2023). A toolchain for domestic heat-pump control using Uppaal Stratego. *Science of Computer Programming*, *230*, Article 102987.
https://doi.org/10.1016/j.scico.2023.102987

Contents lists available at ScienceDirect

Science of Computer Programming

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

A toolchain for domestic heat-pump control using UPPAAL STRATEGO

Imran Riaz Hasrat *, Peter Gjøl Jensen *, Kim Guldstrand Larsen, Jiří Srba

Department of Computer Science, Aalborg University, Aalborg, Denmark

ARTICLE INFO

Article history:
Received 2 December 2022
Received in revised form 12 May 2023
Accepted 7 June 2023
Available online 14 June 2023

Keywords:
Model identification
Strategy synthesis
Heat-pump control
Floor heating

ABSTRACT

Heatpump-based floor-heating systems for domestic heating offer flexibility in energy consumption patterns, which can be utilized for reducing heating costs—in particular when considering hour-based electricity prices. Such flexibility is hard to exploit via classical Model Predictive Control (MPC), and in addition, MPC requires a priori calibration (i.e., model identification) which is often costly and becomes outdated as the dynamics and use of a building change. We solve these shortcomings by combining recent advancements in stochastic model identification and automatic (near-)optimal controller synthesis. Our method suggests an adaptive model-identification using the tool CTSM-R, and an efficient control synthesis based on Q-learning for Euclidean Markov Decision Processes via UPPAAL STRATEGO. This paper investigates three potential control strategy perspectives (i.e., fixed-target, target-band, and setbacks) to achieve energy efficiency in the heating system. To examine the performance of the suggested approaches, we demonstrate our method on an experimental Danish family-house from the OpSys project. The results show that a fixed-target strategy offers up to a 39 % reduction in heating cost while retaining comparable comfort to a standard bang-bang controller. Even better, target-band and setbacks strategies gain up to 46-49 % energy cost savings. Furthermore, we show the flexibility of our method by computing the Pareto-frontier that visualizes the cost/comfort tradeoff. Additionally, we discuss the applicability of STRATEGO for an old-fashioned binary-mode heat-pump system and report significant cost savings (33 %) as compared to the bang-bang controller. Moreover, we also present the performance analysis of STRATEGO against an industry-standard control strategy.

1. Introduction

Space heating is the primary source of energy consumption in residential and commercial buildings, consuming more than 40 % of the energy delivered to such facilities [1]. Intelligent control of heating systems promises to reduce this energy use, thereby countering global warming effects and CO_2 emissions.

Model Predictive Controllers (MPC) have been demonstrated as an efficient method for control of domestic heating systems with the potential for both energy and cost reductions [2–4]. In the classical setting of MPC, an approximate system model under control is constructed before application and paired with a control objective. In the setting of a domestic heating system, such a model describes heat dynamics of the house and external effects that may impact the temperature

* Corresponding authors.
E-mail addresses: imranh@cs.aau.dk (I.R. Hasrat), pgj@cs.aau.dk (P.G. Jensen).

https://doi.org/10.1016/j.scico.2023.102987

of the house, for instance, residential behaviour, outdoor temperature or solar radiation contributing to the temperature of the rooms. This model also defines the possible actions of the controller along with the impact of these control-actions on the behaviour of the system. When the given model is paired with a control objective (e.g. minimize cost and maximize comfort), (model, objective)-pairing induces an *optimization*-problem. Depending on the formalism used to describe the model and control objective, different *solvers* can be used to obtain the next optimal control action to use. As a result, one can apply the MPC-controller in a tight and periodic loop of "observe, solve, act" to control the system.

This paper targets a floor-heating heat-pump control problem for a standard family house. In the case of domestic heating systems, three significant challenges appear in the classical MPC application:

- the heating dynamics of a house are not known a priori,
- the behaviour and dynamics of a house change over the lifetime of the house,
- the design, tuning and deployment of MPC for learning near-optimal control strategies to minimize a desired objective function is not straightforward.

To address these issues, we propose a framework for model-identification for use with the tool UPPAAL Stratego extending on the concept presented by Larsen et al. [3]. In particular, we introduce the model estimation into the loop of the regular MPC-control. We employ the tool CTSM-R (continuous time stochastic modelling in R) [5] to identify the house model by utilizing the *historical data* of the house. We further propose an online strategy synthesis approach where the controller considers the real-time and periodically predicts future control decisions. The controller learns near-optimal strategies within the given price budget based on the selected learning method. The main contributions of the method are as follows.

1. Developing data-driven thermal dynamics and constant-coefficient estimations using CTSM-R.
2. Modelling a case-house in Stratego using the estimated heat transfer coefficients and thermal dynamics.
3. Employing Stratego MPC to learn near-optimal control strategies for operating the heat-pump under four control approaches, namely *fixed-target*, *target-band*, *setbacks* and *binary-mode* controller.
4. Examining the performance stability of Stratego by providing experimentations for four weeks.
5. Analysing the efficiency of the Stratego controller to handle the trade-off between heating cost and user comfort.
6. Investigating the impact of various measurement noise and learning parameters on the controller's performance.

Related work Several domestic heating systems have been discussed in the literature. Vogler-Finck et al. [4] study the house dynamics where the control objective is given in the restricted form of linear equations. They apply *grey-box* modelling facility from MATLAB System Identification toolbox [6] to identify a model for predictive control. They examine three family houses of different ages and the results witness reduced carbon footprint of heating by MPC optimization based on energy and CO_2. However, they use a simple deterministic model for MPC whereas we apply a stochastic model for MPC with an online strategy synthesis approach. An alternative approach is adopted by Larsen et al. [3] who utilize the tool Stratego [7,8] to obtain *near-optimal* control-optimization for switch-controlled hybrid stochastic systems. The authors suggest an online and compositional synthesis approach for a family-house floor-heating system, focusing purely on maximizing comfort and disregarding energy consumption. Hermansen et al. [9] target the energy cost optimization problem by designing an MPC for an existing heat booster substance for ultra-low temperature district heating. They validate their control strategy on a 22-flat building in Copenhagen and report significant energy cost reduction. In our work, we instead optimize for multiple objectives (comfort and cost) and introduce a model identification approach into the loop of an MPC control.

For large and complex systems, model identification becomes an important aspect of MPC control to identify a relatively simple and reduced-order model to make it practically controllable in real-time. However, Prívara et al. [10] describe that model identification is one of the main practical obstacles for large scale use of MPC. The *grey-box* modelling is one of the commonly used approaches that require a small data-set for model identification. Ferracuti et al. [11] and Fonti et al. [12] utilise low-order *grey-box* models to detect short-term (15min, 1 h and 3 h horizons) thermal behaviour of a real building while Reynders et al. [13] used *grey-box* approach to identify reduced-order energy building models. However, these works do not offer a controller synthesis for the identified models. We create a specialised model for a fully automatic heat-pump case where we first identify the model and then use it for strategy synthesis and evaluation.

Vinther et al. [14] propose *black-box* approach to estimate MPC models from multiple Artificial Neural Network (ANN) techniques with a Genetic Algorithm (GA) to predict the future set-points (for room temperatures) in an existing floor-heating system. Furthermore, Nassif et al. [15] also use ANN with GA for optimisation of a HVAC system. Harasty et al. [16] apply a differential evolution (DE) algorithm with ANN for MPC control and optimisation of room temperature for the conservation of cultural heritage. These approaches are based on offline learning, but we offer an online learning approach where the control decisions are predicted periodically for the near future by considering the real-time, making them applicable in practice even for long-lasting horizons.

This paper is an extended version of [17] with the following three additional contributions.

First, we extend our previous experiments to allow Stratego to control three more weeks (from different months) to see the stability in its performance. Second, we add more experiments related to *sensitivity analysis of learning parameters* to explore the edge points where Stratego achieves the best possible performance and stops improving further. The last and the most significant contribution is that we explore *target-band*, *setbacks*, and *binary-mode* heat-pumps as various

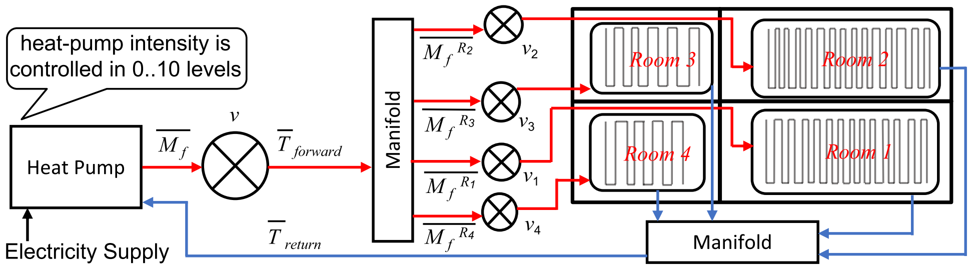


Fig. 1. An Overview of the four-rooms family house equipped with floor-heating.

control strategies. We study the *target-band* and *setbacks* controllers as potential methods of further reducing heating energy consumption. In contrast, the purpose of *binary-mode* controller is to show the ability of Stratego to save heating cost while controlling the old heat-pumps operated under a binary-mode.

Replacing a fixed-target with a comfort/target band is a potential way to unlock the heat flexibility in the thermal dynamics of the building [18,19]. Golmohamadi et al. [18] add a 4 °C comfort-band to optimize power-to-heat flexibility with an Economic Model Predictive Controller, while Williams et al. [19] utilize this flexibility to regulate the room temperatures when a shortage/access happens in the supply side. The heat flexibility offered by heat-pumps plays a key role in reducing the heating cost [20]. The heat can be stored in a thermal storage tank during off-peak periods and used in peak periods. The authors propose a genetic algorithm that shifts most heating loads to off-peak periods. These approaches are only helpful for heating cost reductions, but at the same time they provide generally unacceptable wide range of comfort-bands. However, with our *target-band* controller, we introduce a small 1 °C comfort-band that not only helps to save significant energy costs but also offers acceptable comfort.

Due to technological advancements, users can program schedules that suit their occupancy patterns and comfort preferences. They can set up a lower temperature (called a setback) for the sleeping hours and the period of their absence. Moon et al. [21] study the effect of set-point, setback temperature, and setback period for efficient energy consumption while an artificial neural network (ANN) model is proposed in [22] to estimate the good start point of a setback. In addition to programmable schedules of absence, using an intelligent system that takes the GPS location of the users into account helps to improve the preferred comfort and decrease the heating cost [23]. The intelligent system estimates the arrival time of the users and calculates the heat activation time. The system activates the heat early enough to achieve comfort before the first resident arrives home. However, in our *setbacks* controller, we stick to two fixed setback periods (one for unoccupied duration and the other for sleeping hours) and investigate their effect on comfort and energy cost reductions.

2. Usecase and method

OpSys project [24] investigates energy flexibilities in residential buildings with heat-pumps. An experimental house has been constructed in the OpSys project, and the modeling of the house has been performed using Dymola software. Dymola software [25] is a commercial modeling and simulation environment based on the open Modelica modeling language. We utilize the recorded data from Dymola model to demonstrate our approach on the same experimental family house, which is sketched in Fig. 1. It is a $150m^2$ house with four rooms of different sizes and material properties: *Room 1* is the designated living room with a build-in kitchen, *Room 2* and *Room 4* are bedrooms, and *Room 3* serves as the bathroom with a light concrete floor as opposed to the light wooden floors of the other rooms. The system uses hot water as a means of heat distribution.

As it is standard for existing houses, the heating system has two layers of control: room thermostats (with a fixed mechanical bang-bang controller), and a heat-pump (for which we derive a controller). The model reflects a realistic scenario where an existing building is retrofitted with an intelligent heat-pump. Notice here that each room thermostat acts independently of the heat-pump and the other thermostats; it is simply concerned with opening the flow of hot water when the room temperature drops below a certain set point (fixed to 22 °C in our experiments). This implies that the only control-point of concern is the intensity of the heat-pump.

The system is designed as a closed-loop system with a fixed distribution-key. This means that the ingoing massflow of water ($\overline{M_f}$) must be distributed to the rooms via the manifold. Conventionally, the manifold is calibrated s.t. a fixed proportion of water is designated to each room. This distribution-key we denote $\overline{M_f}^{R_i}$ for a room i, where $i = 1, \ldots, k$, and k is the number of rooms. The individual room thermostats regulate the binary valves v_i. This implies that the massflow is re-distributed (proportionally) to the remaining open valves if certain valves are shut. Furthermore, we denote by $\overline{T}_{forward}$ and $\overline{T}_{return}$ the forward and return water temperature.

The main purpose of the heat-pump is to balance the difference between the room temperatures $\tilde{T}_r^i$ and a given set point T_g with the cost of heating $cost(\tau) = price_\tau \cdot w_\tau$, which is a time-dependent function of the energy consumption of the heat-pump w_τ (determined directly by the controller) and the market electricity price ($price_\tau$) which varies on an hourly basis and is known 24 hours in advance. Naturally, a consumer wishes to weight comfort against cost, and we

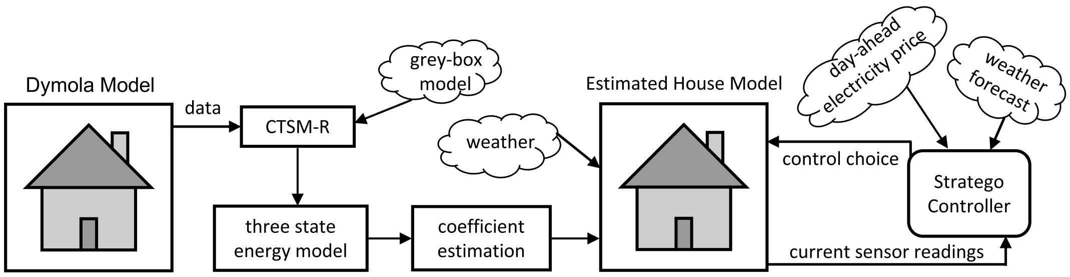


Fig. 2. Estimation and control process using observed data.

can thus state the optimization criteria and introduce the $0 \leq W_{comf} \leq 1$ weighting factor that allows for such tuning. This allows us to derive the following (W_{comf}-parameterized) fitness function for our controllers for a period τ_0 to τ_n given that the heat-pump settings, energy prices and room temperatures (denoted $\tilde{T}_r^i(\tau)$) are known for the duration for k rooms as:

$$F(\tau_0, \tau_n) = \int_{\tau_0}^{\tau_n} \left(\left((1 - W_{comf}) \cdot cost(x)\right) + W_{comf} \cdot \sqrt{\sum_i^k (T_g - \tilde{T}_r^i(x))^2} \right) dx \quad (1)$$

where the first part in the integral reflects the contribution of energy cost while the second part records the penalty given when the indoor temperature ($\tilde{T}_r^i$) in a room deviates from the set point (T_g); in the summation part, the individual penalties are added to make a combined penalty. Notice that this function penalizes more significant deviations of temperatures in a squared fashion while the cost increase has a linear impact on the fitness function.

2.1. Methodological overview

While a high-fidelity model of our experimental house (Dymola model) is provided, this model is infeasible for practical experiments both due to its high computational effort caused by the high fidelity and due to licensing issues making massive parallel experiments unfeasible. The overview of our suggested approach can be seen in Fig. 2. We initially derive a low-fidelity model of the thermal dynamics of the building via a data-driven *grey-box* model estimation using the CTSM-R software [26]. The CTSM-R library allows finding best-fit parameters for a stochastic model based on samples of a (virtual) house. The specific grey-box formulation is given in Section 3 along with an evaluation of its performance. Notice that the model-identification problem is constructed s.t. the estimation can be computed in a decomposed manner, i.e. a model is computed individually for each room, and the set of models is later recomposed. We do so to avoid an explosion in computational effort from a large set of variables to estimate. While this *grey-box* model yields us the internal heat coefficients (how the energy dissipates between rooms or a heater), it does not capture the heat transfer from the hot water flowing through the pipes. Therefore, we derive a model capturing the relationship between internal temperature changes and the drop in the heat-carrying water, yielding us the return water temperature ($\overline{T}_{return}$).

Given these two thermal models, we can develop estimated house model of the experimental house and instantiate a Euclidean Markov Decision Process (EMDP) [27] from which Stratego can learn near-optimal control strategies. This model not only simulates the behaviour of the house but also contains the expected weather and future energy price prediction. We use this model for both evaluation and predictive control in our experimental setup, including an experiment with varying degrees of noise in the model used for predictive control. For instance, we replay a historical weather scenario during the evaluation while the predictive controller only has access to an approximate weather forecast.

3. Thermal model identification

The CTSM-R software [5] allows for continuous-time *grey-box* model identification. The general method of CTSM-R is based around maximum likelihood estimation and a gradient-decent approach for convergence. It identifies Continuous Time Stochastic Model and estimates the embedded parameters. CTSM-R has been successfully applied for the identification and estimation of physical system models, e.g., heat thermodynamics of buildings and walls, thermostats and radiators, and more [28–30].

We model the thermal dynamics of the house as presented in Equations (2)–(4) from Fig. 3. For readability, we annotate the system in the following way: predicted state-variables by a tilde, e.g. $\tilde{T}_r^i$, inputs directly affected by the two levels of controls (i.e. the *heat-pump* or the *room thermostat*) with an overline, e.g. $\overline{M^i}$, impact of nature with a dot, e.g. $\dot{T}_a$, and constants (to be estimated) are left without decoration.

The thermal dynamics model follows a classical three-state framework consisting of room temperature ($\tilde{T}_r^i$), heater temperature ($\tilde{T}_h^i$) and envelope temperature ($\tilde{T}_e^i$). The three-state energy model presented in Equations (2)–(4) is a modified version of what was presented in [26]. The model [26] is suited for a single room open to outside from all sides but unable

Data	Description
$\tilde{T}^i_r$	room i air temp.[°C]
$\tilde{T}^i_h$	room i floor temp.[°C]
$\tilde{T}^i_e$	room i envelop temp.[°C]
$\overline{M^i}$	room i water mass flow [kg/s]
$\overline{T}_{forward}$	water temp. exiting heat-pump [°C]
$\dot{T}_{outdoor}$	outside temp. [°C]
$\overline{T}^i_{return}$	water temp. exiting room i [°C]
$\dot{S}^i$	solar heat to room i [Watt]

(a) Measured data

Constant	Description
α^i_h	heat resistance between floor and room air
α^i_e	heat resistance between envelope and room air
α^i_s	heat resistance between solar radiation and room
α^i_w	heat resistance between water pipes and floor
α^i_a	heat resistance between envelope and outdoor
α^i_n	heat resistance between room i and room n
β^i_h	heat capacity of floor pipies
β^e_h	heat capacity of envelop

(b) Heat exchange coefficients

$$\frac{d\tilde{T}^i_r}{dt} = \alpha^i_h(\tilde{T}^i_h - \tilde{T}^i_r) + \alpha^i_e(\tilde{T}^i_e - \tilde{T}^i_r) + \alpha^i_s \cdot \dot{S}^i \tag{2}$$

$$\frac{d\tilde{T}^i_h}{dt} = \frac{\alpha^i_h}{\beta^i_h}(\tilde{T}^i_r - \tilde{T}^i_h) + \alpha^i_w \cdot \overline{M^i}(\overline{T}_{forward} - \tilde{T}^i_h) \tag{3}$$

$$\frac{d\tilde{T}^i_e}{dt} = \frac{\alpha^i_e}{\beta^i_e}(\tilde{T}^i_r - \tilde{T}^i_e) + \alpha^i_a(\dot{T}_{outdoor} - \tilde{T}^i_e) + \sum_{\substack{n=1\\n\neq i}}^{k} \alpha^i_n(\hat{T}^n_r - \tilde{T}^i_e) \tag{4}$$

(c) Proposed three state energy model

Fig. 3. The thermodynamics system with the input data and the heat transfer coefficients where $i \in \{1, 2, 3, 4\}$ and, (α), (β) are "resistance" and "capacity".

to capture the heat effect from neighboring rooms. In contrast, our modified model captures the heat contributions from neighboring rooms well. To reduce the complexity, we here consider the rooms as individual model identification problems to overcome instability in the model identification when the number of coefficients to be estimated grows. In Equation (2), the room temperature ($\tilde{T}^i_r$) is directly affected by the heater (relative to the coefficient α^i_h), the envelope (relative to the coefficient α^i_e) and the direct solar radiation of the room $\dot{S}^i$ (relative to the coefficient α^i_s). In Equation (3), the heater temperature ($\tilde{T}^i_h$) is impacted by the room temperature. It also receives energy from the in-flowing hot water, which is proportional to the flow rate $\overline{M^i}$ and the relative difference to the forward temperature of the water ($\overline{T}_{forward}$). In Equation (4), the envelope ($\tilde{T}^i_e$) models the energy transfer through the walls surrounding the room. The envelope thus exchanges energy with the room itself ($\tilde{T}^i_r$), the outside world $\dot{T}_{outdoor}$ (proportional to the coefficient α^i_a), and with the other rooms of the house (the last sum term of Equation (4)).

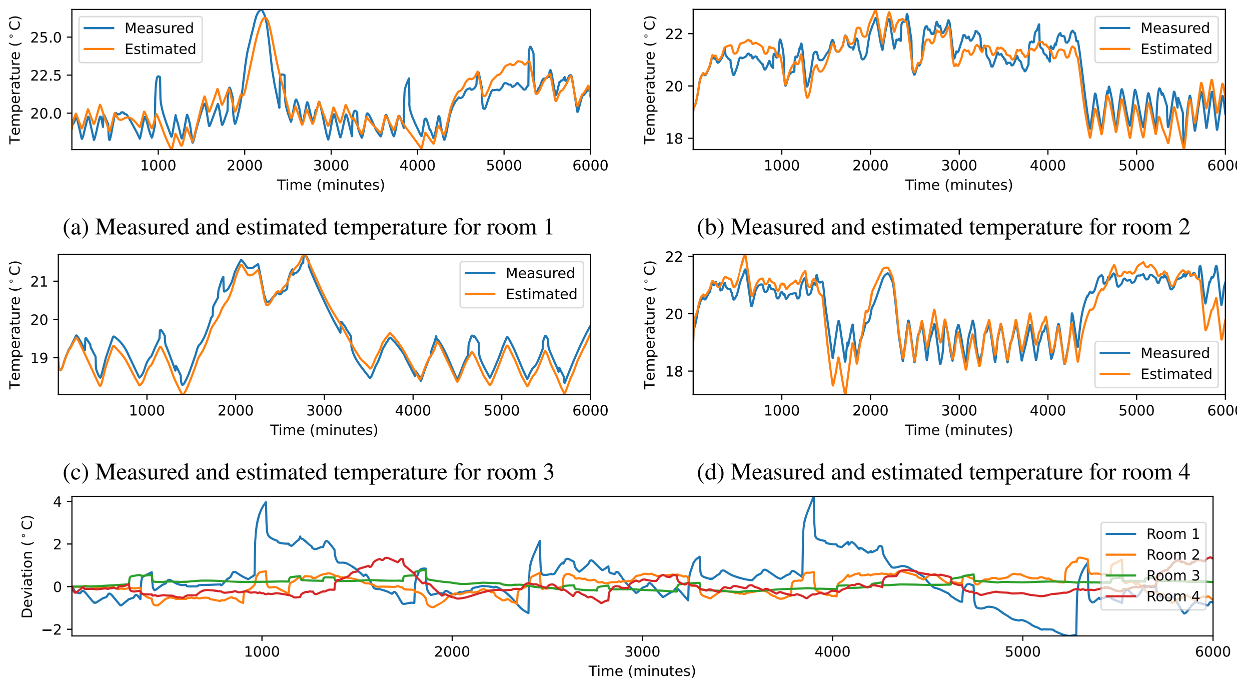
A legend of the variables used can be seen in Fig. 3a for data-variables and Fig. 3b for the transfer coefficients. Given a time-series of historical data of the variables presented in Fig. 3a (excluding $\tilde{T}^i_h$ and $\tilde{T}^i_e$), the CTSM-R software can estimate the heat transfer coefficients presented in Fig. 3b. The model is estimated s.t. the suggested coefficients allow us to predict $\tilde{T}^i_r$ with a high accuracy given known values for the massflow and temperature of the feed-in water and the environmental influences such as sun and outdoor temperature.

Decomposed grey-box modelling Notice here that the presented model only considers a single room. This implies that the identification can be done on a room-to-room basis without the explosion in the coefficients to be estimated. In general, this allows enables faster model identification. This implies repeating the same model-identification scheme four times but with different indexing. As we shall see later, four such identified models can be recomposed to form a complete model with high predictive power.

Evaluation of the estimations We shall now assess the quality of our proposed model-identification. The model is identified on a time-series consisting of data generated by the high-fidelity Dymola model using historical weather input from February 05, 2009, from Aalborg, Denmark and five days forward at a sample rate of 60 seconds.

We note that the model estimation for the values is completed in less than 7 minutes for each room. We remark here that the data used for the estimation contains some effects not captured by Equations (2)–(4); notably occupants and cooking activities, contributing to significant noise in the system.

To estimate the quality of the model, we compute a 6000 minutes time series from Dymola model data into CTSM-R and compare the predicted room temperatures of each of the identified room models to the reference temperatures. In Fig. 4 we see a head-to-head comparison for all the rooms where we can observe that the identified model demonstrates good predictive power. In particular, we note that no divergent behaviour is observed. The deviation of each room from measured data can be seen in Fig. 4e. Excluding Room 1, the largest average deviation observed for all rooms is less than 1 C°, and in general, kept below 0.5 C° of difference. However, in Room 1 in the following intervals 1000-1400, 2400-2900, 3800-4300,



(a) Measured and estimated temperature for room 1

(b) Measured and estimated temperature for room 2

(c) Measured and estimated temperature for room 3

(d) Measured and estimated temperature for room 4

(e) Deviation of estimated temperature from measured temperature for each room

Fig. 4. Predicted, measured indoor temperatures and the deviations between them.

and 4700-5400 minutes, significant deviation occurs, which is traced back (in the data) to the unaccounted contribution of heat from cooking activities which are expected to be unobservable in a real application—and not captured during the model identification. The starting points of these intervals approximately become 17:00, 16:00, 15:30, and 6:30 when converted to hours of the day. The heat contribution keeps growing for about one and a half hours during each interval, reflecting that some stove is continuously on. These hours probably refer to the occupants' dinner and breakfast cooking activities. We also observe a lower peak between 5000 to 6000 minutes which can be the outcome of several factors. The main factor is that CTSM-R tries to minimize the overall average error; when it finds some significant upper peaks in the measured data due to kitchen activities, it introduces lower peaks in the predicted trajectory to balance the error effect. This lower peak can also result from human behaviour, e.g., opening a window. The other possible factor is that the house thermodynamics model does not capture the solar radiation effect and some other heat losses.

Estimating return-water temperatures The model created so far is only concerned with the internal transfer of energy in the house; however, in Fig. 1 we can see that the return-water of the rooms is given as input to the heat-pump. We thus need to estimate a second model for predicting the return-water temperature. We here again adopt a data-driven approach and derive a model of this temperature-drop directly from data. We here assume a simple linear model; a room's temperature increase is assumed to be primarily due to energy dissipation from the heater. While this assumption is valid in a closed system (due to the principle of energy conservation), we know that the assumption is incorrect: external influences occur, such as energy loss and gain from neighboring rooms. This assumption allows us to establish the relationship described in Equations (5)–(6). In Equation (5), we state that the energy loss ($Energy^i_{loss}$) from the hot water pipes is equal to the difference of forward water ($\overline{T}_{forward}$) and floor temperature ($\tilde{T}^i_h$) for a specific room. The Equation (6) introduces the energy gain in a room ($Energy^i_{gain}$), which is the temperature difference between floor and indoor temperature. In Equation (7), we show the assumption that the difference between the forward and return water temperatures for individual rooms is equal to the sum of energy loss and gain. By using historical data for $\overline{T}_{forward}$, $\tilde{T}^i_h$ and $\tilde{T}^i_r$, we can estimate the coefficients α, β and intercept γ using a simple linear regression method in R. Again, these coefficients allow us to later predict the value of $\overline{T}^i_{return}$ for use in our full predictive model. Notice that prior to model-identification, we filter out data points where the water is not flowing (i.e. the local thermostat has shut off the supply). It is observed that while the return temperatures are all above 32 °C, the absolute mean error in estimating them is for all the rooms between 0.74 °C and 0.95 °C.

$$Energy^i_{loss} = \overline{T}_{forward} - \tilde{T}^i_h \tag{5}$$

$$Energy^i_{gain} = \tilde{T}^i_h - \tilde{T}^i_r \tag{6}$$

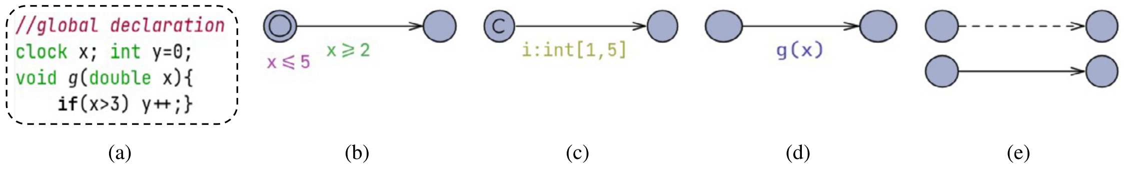


Fig. 5. Examples of UPPAAL primitives.

$$\overline{T}_{forward} - \overline{T}^{i}_{return} \approx \alpha \cdot Energy^{i}_{loss} + \beta \cdot Energy^{i}_{gain} + \gamma \tag{7}$$

4. Modelling in UPPAAL STRATEGO

We now create a complete predictive model given the two identified thermal models. For doing so, we use the UPPAAL tool suit [31–34] which has been successfully applied in many industrial projects for verification, performance analysis, and strategy synthesis. UPPAAL STRATEGO [7] is a branch of the tool that provides machine learning-based techniques for strategy synthesis and cost optimization of different controllers from Priced Timed MDPs. It has a rich modelling formalism for stochastic and hybrid games and control synthesis exploiting efficient reinforcement learning facilities. In UPPAAL, systems are modelled as networks of finite-state automata processes. In addition, STRATEGO provides C-library support [35] which offers a convenient way to construct complex interactions with other libraries and historical data, for instance, STRATEGO itself.

4.1. *Introduction to* UPPAAL STRATEGO

UPPAAL provides a graphical user interface to define system descriptions as networks of timed automata. It includes a description language, a simulator, and a model-checker as three main parts. The description language is the modeling language that describes the system behavior and provides the data variables for networks of timed automata. The automata network represents the processes that work in parallel; each process is instantiated using a parameterized template. The simulator and model-checker provide automated analysis and also help to enable early fault detection before verifying the system. To provide a basic understanding of the tool semantics, we explain some basic examples in Fig. 5. Further information about the tool can be found in [31,36].

Locations: The single circles in Fig. 5 are regular locations, and the double circle represents an initial location. The arrows between the locations are transitions/actions. The first location in Fig. 5c is committed. A committed state cannot be delayed, and the system in a committed state does not allow time to pass.

Clocks and variables: The real-valued clocks facility is available in the tool to capture critical timing aspects of a system. The clocks represent the time progress and handle the continuous expressions as differential equations. On the other hand, discrete variables in the tool are used to provide the data storage facility. In Fig. 5a, we globally declare the clocks, variables, and functions used in the examples. Anything defined in the global declarations is accessible by all the processes in the system. However, the local declarations limit access to only a specific process.

Invariant: It is an expression on the locations which uses clocks and integer variables to define the maximum allowed time of stay on that location. The invariant `x<=5` in Fig. 5b indicates that the system is allowed to stay here until the clock `x` reaches the value up to 5.

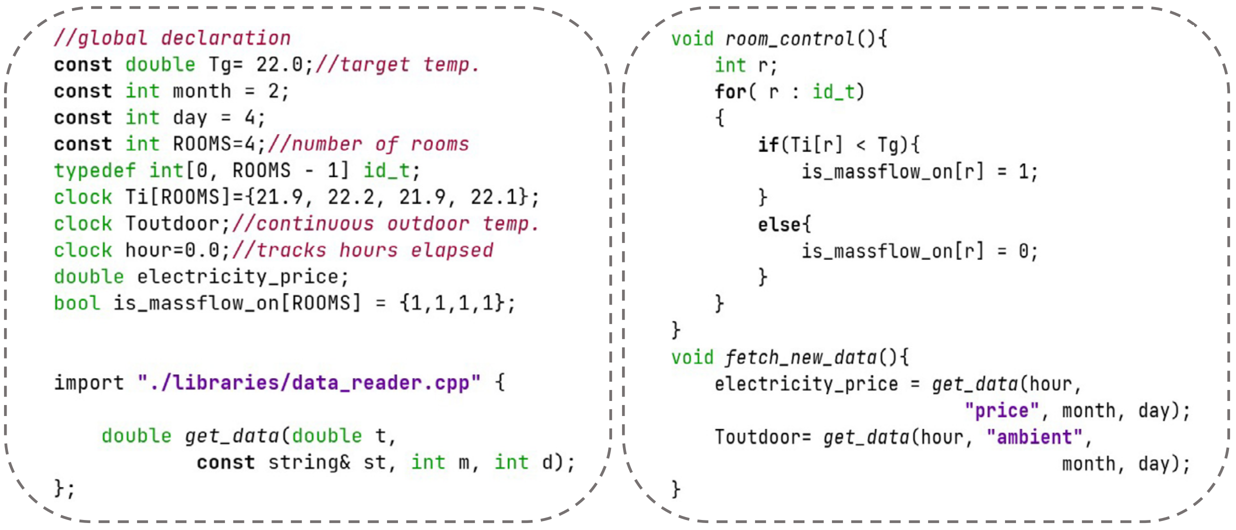
Guard: A guard is a condition that uses clocks, integers, and constants; it evaluates to a Boolean value. The transition in Fig. 5b can only be taken when the guard evaluates to true, i.e., clock `x` has at least a value 2.

Select: A select label generates a temporary variable accessible only during that particular edge/transition. The select statement in Fig. 5c is equivalent to $\mathtt{i} = 1, \ldots, 5$, meaning that five transitions with unique `i` values exist, and the controller can choose any transition that sets `i` to a value between 1 and 5.

Update: The update label is used to set, reset or update the values of variables directly or through functions just before leaving a transition. The functions can be from the system's description or some C libraries. The update label in Fig. 5d increments the value of `y` using function `g(x)` if clock `x` value is more than 3.

Controllable and uncontrollable transitions: In UPPAAL, the transitions can be controllable or uncontrollable. In Fig. 5e, the solid edge indicates controllable action, and the dashed line indicates uncontrollable (controlled by the environment) action. If two parallel running processes have controllable and uncontrollable actions enabled simultaneously, the environment has priority to take any transition or pass control to the controller. If the controller is given control, it decides the action according to its strategy.

Communication: In UPPAAL, the processes communicate with each other through channels or shared variables. The variables declared in global declarations are shared variables. The variable `y` is shared by all the processes and is responsible for exchanging information.



```
//global declaration
const double Tg= 22.0;//target temp.
const int month = 2;
const int day = 4;
const int ROOMS=4;//number of rooms
typedef int[0, ROOMS - 1] id_t;
clock Ti[ROOMS]={21.9, 22.2, 21.9, 22.1};
clock Toutdoor;//continuous outdoor temp.
clock hour=0.0;//tracks hours elapsed
double electricity_price;
bool is_massflow_on[ROOMS] = {1,1,1,1};


import "./libraries/data_reader.cpp" {

    double get_data(double t,
            const string& st, int m, int d);
};
```

(a) Global declarations

```
void room_control(){
    int r;
    for( r : id_t)
    {
        if(Ti[r] < Tg){
            is_massflow_on[r] = 1;
        }
        else{
            is_massflow_on[r] = 0;
        }
    }
}
void fetch_new_data(){
    electricity_price = get_data(hour,
                        "price", month, day);
    Toutdoor= get_data(hour, "ambient",
                                month, day);
}
```

(b) Function definitions

Fig. 6. Function definitions and related variable declarations.

4.2. *Case house model in* UPPAAL STRATEGO

Recall that our heating system has two levels of control: selection of the heat-pump intensity and opening/closing of the room thermostats. We control the heat-pump intensity with a bang-bang (BB) and a STRATEGO controller and compare their performance. However, for both BB and STRATEGO, each room's thermostat valve opens if the indoor temperature falls below 22 °C and closes if it rises over 22 °C. BB switches the heat-pump *on* and *off* based on the current room temperatures, unconcerned with electricity prices or weather conditions. STRATEGO, on the other hand, employs machine learning approaches to learn better future control decisions by considering the electricity price and stochastic weather forecast. It optimizes the *fitness* function to reduce the indoor and target temperature gap and minimizes the cost.

The overall composition of the system as a model in STRATEGO can be seen in Fig. 7. We design sub-parts of the system as separate templates (parameterizable automata). The dashed-line areas represent four templates of the model: *Room*, *HeatPumpController*, *DataReader* and *ObjectiveFunction*. In the *Room* template, we express the continuous variables $\tilde{T}_r^i$, $\tilde{T}_h^i$ and $\tilde{T}_e^i$ (developed in Equations (2)–(4) as real-time clocks which evolve with time, but with rate-expressions matching the identified models. The *DataReader* template reads the *weather* and *day-ahead electricity price* information from the data file into the model. The *ObjectiveFunction* template implements the *fitness/optimization* function (see Equation (1)). The *HeatPumpController* template implements the control mechanism of the controller where choices for using different energy levels for heat-pump are made.

To perform specific tasks, the templates in Fig. 7 use functions namely `room_control()`, `calculate_cop()`, `init()`, `set_massflow()` and `fetch_new_data()`. We here describe `room_control()` and `fetch_new_data()` functions (see Fig. 6b) and their related declarations (see Fig. 6a). However, the remaining functions are long and complex; hence their details are kept in the repeatability package [37]. The function `room_control()` implements the bang-bang controllers of the room thermostats, which decides whether to periodically open or close the valves leading to each room, this directly gives a way to compute the distribution of the ongoing flow by the `set_massflow()` function. The room temperature (`Ti[r]`) is compared with the target temperature (T_g) where r loops over `id_t` and `id_t` $= 0, \dots, 3$ as declared in Fig. 6a. The value of `is_massflow_on[r]` is set to 1 if room `r` has temperature below T_g and needs heat in the next interval and vice versa. To fetch electricity and weather information from CSV files, we import our customized function named `get_data()` from c-library (data_reader.cpp). For both weather and electricity price, the function `fetch_new_data()` provides `time`, `day`, and `month` information to `get_data()` periodically (once an hour) and gets updated data for that particular point in time.

STRATEGO handles this heating problem as a *stochastic hybrid game*. An (instantiated) set of templates constitutes a Network of (Hybrid Stochastic) Timed Game Automata (which semantically gives us an EMDP) where a state is given by a location vector (one for each template instance) and an assignment of concrete values to any variable. The system may then evolve by respecting `guards` and `invariants` – in particular, those that restrict the continuous development of clocks with respect to our house dynamics. In *HeatPumpController*, the initial location is committed (marked by a C in the locations) (see Fig. 7) forcing the model to initially call the `init()` function, initializing the *weather*, *electricity price* and *solar radiations* values. At the following location, the controller may take any of the two edges depending on the amount of energy it decides to operate the heat-pump in an interval; in the lower transition, the heat-pump is shut off while in the upper uses the *select* statement to compactly implement 11 different intensity levels of the controller (encoded in the type `intensity_t`). I.e. the controller assigns the temporary variable i a discrete value from `intensity_t` (range between 0 and 10) to reflect

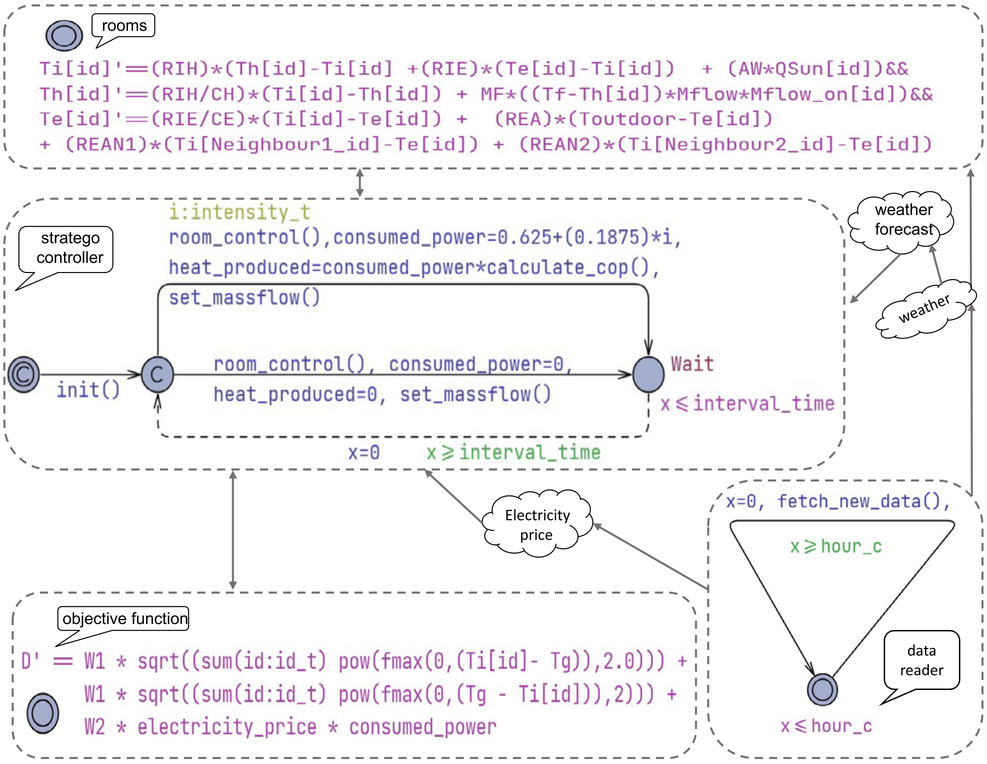

rooms
Ti[id]'==(RIH)*(Th[id]-Ti[id] +(RIE)*(Te[id]-Ti[id]) + (AW*QSun[id])&&
Th[id]'==(RIH/CH)*(Ti[id]-Th[id]) + MF*((Tf-Th[id])*Mflow*Mflow_on[id])&&
Te[id]'==(RIE/CE)*(Ti[id]-Te[id]) + (REA)*(Toutdoor-Te[id])
+ (REAN1)*(Ti[Neighbour1_id]-Te[id]) + (REAN2)*(Ti[Neighbour2_id]-Te[id])
stratego controller
i:intensity_t
room_control(),consumed_power=0.625+(0.1875)*i,
heat_produced=consumed_power*calculate_cop(),
set_massflow()
init()
C
room_control(), consumed_power=0,
heat_produced=0, set_massflow()
Wait
x≤interval_time
x=0
x≥interval_time
weather forecast
weather
Electricity price
objective function
D' = W1 * sqrt((sum(id:id_t) pow(fmax(0,(Ti[id]- Tg)),2.0))) +
W1 * sqrt((sum(id:id_t) pow(fmax(0,(Tg - Ti[id])),2))) +
W2 * electricity_price * consumed_power
x=0, fetch_new_data(),
x≥hour_c
data reader
x≤hour_c

Fig. 7. Overall system composition.

different intensity level choices to operate the heat-pump. The function `room_control()` is triggered to open or close the room thermostat valves. The variable `consumed_power` contains the amount of electricity selected by the controller for each intermediate interval during strategy synthesis while `heat_produced` is the produced heat for every interval. The function `calculate_cop()` calculates the *coefficient of performance* (COP) value used to measure the produced heat. Notice that the COP value, for a given return-water temperature and outdoor temperature, provides a gearing of input to output energy; i.e. at a COP value of 2, a single kWh of power yields 2 kWh of energy. Such gearing is normally found in the technical specifications of a heat-pump. This gearing, along with the pump intensity-setting, massflow and the return-water temperature, allows for the computation of the *forward* temperature and using the equations for return-water temperature. The return-water temperature is computed based on the previous time-step, and the relative changes in the indoor temperatures allow us to predict this value using the developed equations from Section 3. At the `wait` location, the invariant `x<=interval_time` bounds the system to stay there until the value of clock `x` (local clock) reaches `interval_time` i.e. 15 minutes. The constant `interval_time` is the control frequency of the heat-pump. The guard `x>=interval_time` prevents the system from making this transition before spending 15 minutes at the `wait` location, after which the clock `x` is reset to zero, allowing for timing a new interval.

Evaluation using STRATEGO We use the provided model for both learning and evaluation in our experiments. However, in the evaluation phase, the control choices of the STRATEGO controller template are restricted to follow those suggested by a call to an external library, implying that the controller is invoked for every simulated 15 minutes. When we evaluate the system under the control of a bang-bang controller, this call returns either full intensity or the 0 intensity value. When the STRATEGO itself is used as a controller instead, the call instantiates the above templates—but instead of evaluating, it will synthesize a controller by repeated sampling.

4.3. *Learning by* STRATEGO

To attain a predictive controller, STRATEGO trains on the instantiated model, using the initial state estimate by exploiting repeated sampling and Q-learning. This allows the tool to factor in temporal changes such as weather and price changes of the near future. However, four problems arise:

1. not all variables are observable, specifically only the room-temperatures ($\tilde{T}_r^i$), the weather forecast and the price projection can be observed,
2. controllers take time to compute, leading to a delay from observation to reaction,

3. the development of the real world (or evaluation model) can deviate over time, and
4. both weather forecasts and prices have a limited horizon.

The latter points we shall discuss in Section 4.4, and let us instead here address the first issue.

Initial state estimation In our three-state thermodynamics model, the changes in room temperature ($\tilde{T}_r^i$) are directly dependent on two hidden states, i.e., floor temperature ($\tilde{T}_h^i$) and envelope temperature ($\tilde{T}_e^i$). The states are unobservable and they intuitively track the abstract value of "energy stored in the heater" and "energy stored in the wall" respectively. This implies that the tool works on a partially observable model. To attain reasonable estimates, we can predict these hidden variables—and similarly the future value of observable values to accommodate for the observational delay. Assuming that a full state is known at τ_t and that a control choice has already been made, then the state at τ_{t+15} can be captured by forward simulation of the model. Experiments show that this method of approximating the hidden variables $\tilde{T}_e^i$ and $\tilde{T}_h^i$ allowed a drift of no more than 0.1 °C between the repeated estimates of the variables in the model used for learning and the model used for evaluation.

4.4. Online synthesis

Given that energy costs are only known 24 hours in advance and that weather forecasts are known to have degrading predictive power the further into the future they look, we utilize an online synthesis procedure. At the same time, as our model used for controller synthesis has hidden variables, a rapid recomputation of a strategy allows us to adjust for errors made in the initial state-estimation and potential discrepancies with the real house (or evaluation model in our case).

We thus propose a method where at each 15-minute interval, the volatile variables (energy price, weather and measurable variables of the house) are monitored and transferred to the controller. This allows the controller to instantiate the method of Section 4.3 with the most recent measurements. We can then let Stratego synthesize a controller on. An overview of this flow is given in Fig. 8. We divide each day into four 6-hour periods, each with 24 of 15-minute intervals. The reuse interval (k) is 24, which means a single strategy is used for k intervals. In the first interval, Stratego gets the current *room temperatures*, *weather forecast* and *day-ahead electricity price*, estimates the initial state and synthesizes a strategy that minimizes the fitness function F by sampling 12-hours (learning horizon) ahead in future. The controller saves the first strategy (*S1*). The heat-pump uses *S1* during intervals 2–25 and makes another strategy during interval 25, which is then used for the next 24 intervals (i.e. from 26–49) and so on.

Notice here that it is assumed that the synthesis procedure is *not* instantaneous, implying that a synthesized controller can only be applied in the sub-subsequent interval. To overcome this issue, we apply the initial state-estimation as described in Section 4.3 to estimate the house state at a time-point $\tau + 15$ from measurements obtained at τ. We set the training horizon of the method to be 12 hours, implying that the learning method will only sample the system up to a horizon of 12 hours from the starting measurement. This can aptly be done in Stratego using the query presented in Equation (8). We here see that the controller is trained to only react on the observable variables `minute_clock` (tracking real-time), `Ti [i]` (tracking indoor room temperatures), `Toutdoor` (tracking weather forecast) and `Tforward` (tracking the forward water temperature). The observable variables are input to the controller. The energy price is also an input, but it is not observed directly in the training as it follows a fixed pattern from historical data. The result of the synthesis procedure is a reactive strategy that for an assignment of the input variables yields a (near-)optimal action to take [3,8,27]

```
strategy S = minE (F) [minute_clock <= 12*60)]                          (8)
            {} -> {minute_clock, Ti[0], Ti[1], Ti[2], Ti[3],
                    Toutdoor, Tforward}: <> minute_clock==(12*60)
```

Notice also that Equation (8) defines a feature vector (observable variables of the state-space) to be only directly measurable variables of the original system. This makes the strategy directly applicable for several intervals (up to the learning horizon) in the actual system under control. We can exploit this to lower the overall computational effort of deploying our approach, which directly impacts the overall energy impact of running the smart heating system. Essentially, by reusing a strategy for several intervals, a single server can act as a controller for a group of houses, i.e. if a strategy only needs to be computed every six intervals, six houses can share the computational resource.

5. Experiments and evaluation

In this section, we first describe the experimental setup with all realistic conditions and assumptions. We then evaluate our approach by providing a series of experiments and also study different factors that affect the performance of Stratego. A reproducibility package for our experimentation is available in [37].

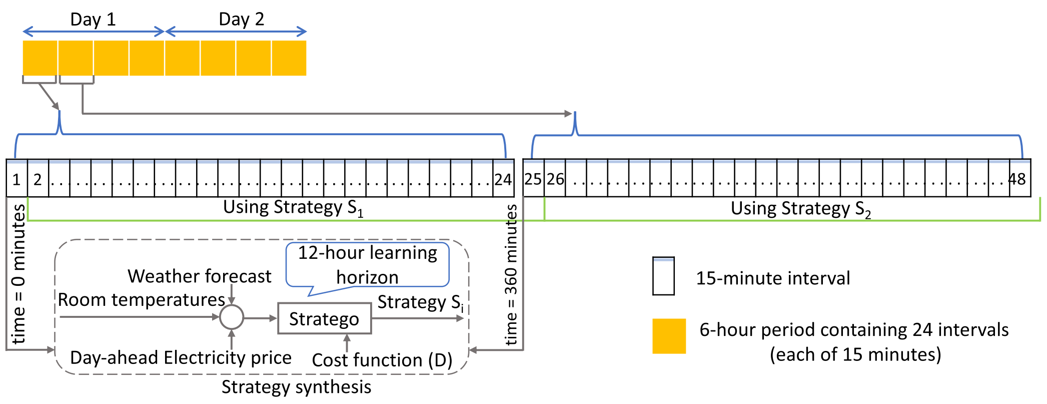


Fig. 8. Online synthesis approach shown on a two day period.

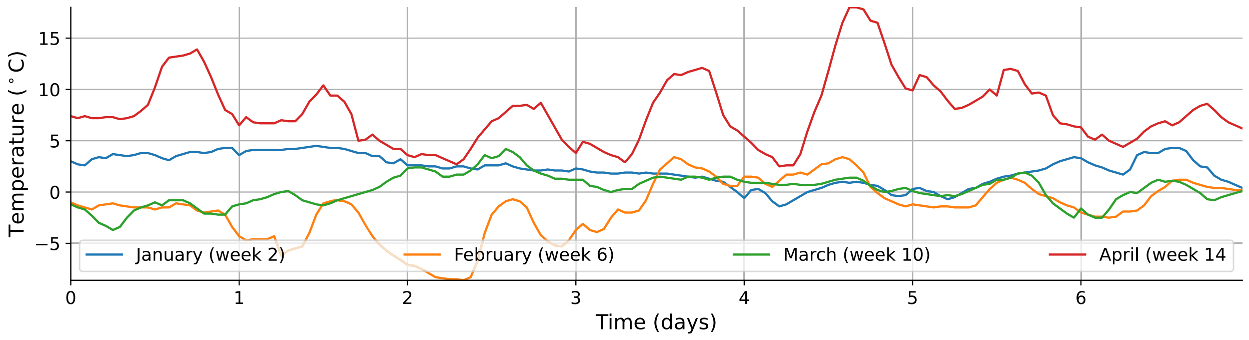


Fig. 9. Outside weather during the weeks controlled in our experiments.

5.1. Experimental setup

To validate our approach, we experiment with controlling the experimental house from the OpSys project for four individual weeks with weather conditions matching January (week 2), February (week 6), March (week 10), and April (week 14), 2018. As a reference, we use the estimated model presented in section 4. We utilize the recent energy prices (autumn 2022) from the Danish day-ahead electricity market in all the experiments presented in the rest of the section. The goal of controller is to minimize the parameterized cost-criteria defined in Equation (1) in section 2 for the period under control. In all experiments, we compare different variations of the online STRATEGO-based controller against a standard bang-bang (BB) controller. If the temperature in any room drops below 22 °C, the BB controller turns the heat-pump on; otherwise, it keeps the heat-pump off.

We conduct four series of experiments:

1. a series of evaluations with realistic observability and time,
2. a series of alternative control strategy approaches,
3. a study of sensitivity analysis of impact of measurement noise, and
4. a study of sensitivity analysis of learning parameters.

All experiments are conducted on an AMD EPYC 7551 processor and limited to a single core and 2GB of memory. We limit the controller to 15-minute control intervals which is sufficient for systems with as slow dynamic as floor-heating. For each reported configuration, the experiment is repeated 10 times and the mean value is reported with standard deviation intervals reported in bar-charts. In all series, we observe changes to the results when the weights of comfort ($0 \le W_{comf} \le 1$) changes; we modify these values in steps of 0.1. To make this weighting independent of the given week and to allow for this proportional weighting between two units (cost (DKK) and squared degrees Celsius), we compute a normalization-factor *norm* based on the BB controller. This normalization-factor is computed by estimating the performance of the BB controller on the week leading up to experimental week in terms of *cost* and *comfort*. The normalization-factor is then given directly by $norm = \frac{comfort}{cost}$.

The outside temperature for different weeks can be seen in Fig. 9. Week 6 is the coldest, and week 14 is the hottest week we control in our experiments. We consider the fact that a house and a controller cannot have the same weather information because the house experiences the real weather, and the controller has to work with weather forecasts. In our experimental

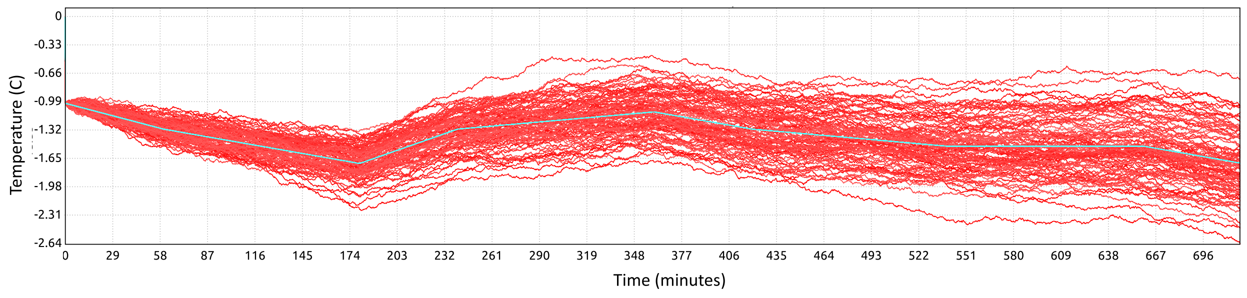
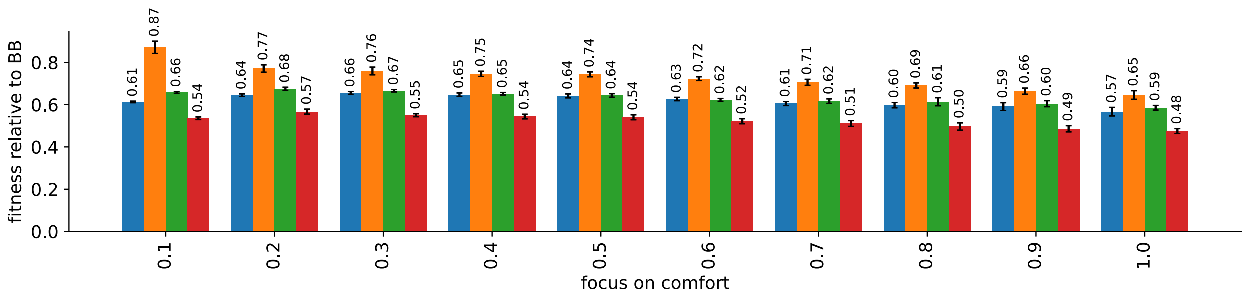
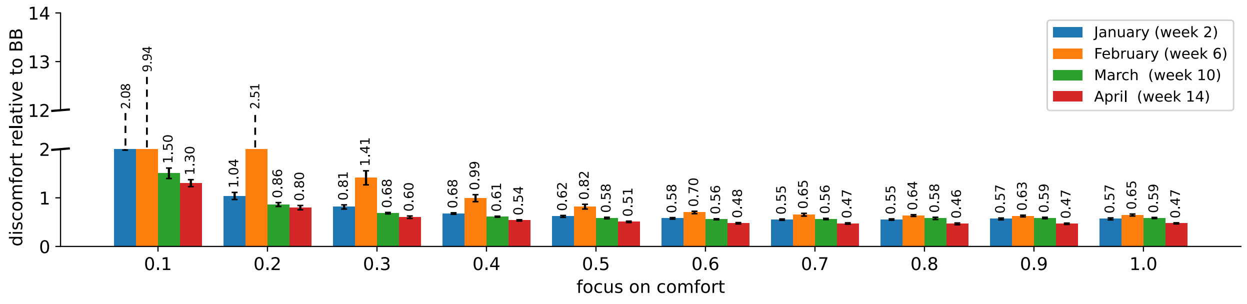


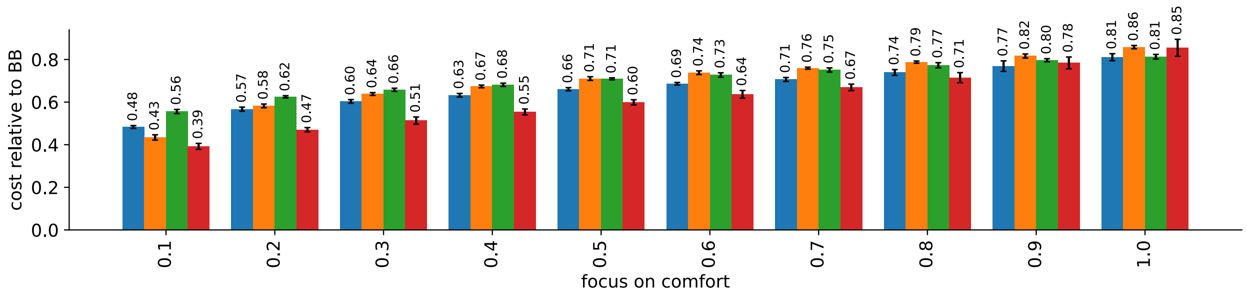
Fig. 10. Actual weather and random noise added trajectories over 100 simulations where the middle line is the actual weather, and the red lines represent the weather predictions.



(a) Fitness measured in different months against different values of W_{comf}



(b) Discomfort measured in different months against different values of W_{comf}



(c) Energy cost measured in different months against different values of W_{comf}

Fig. 11. Experimental results for four different weeks under *fixed-target* STRATEGO with realistic assumptions on observability and time.

setup, instead of providing the STRATEGO controller with the exact weather information, we add random noise to allow the controller to work under realistic conditions. The result of 100 simulations executed under 12 hours is presented in Fig. 10. It can be observed that the predicted weather (forecast) may differ from actual weather by approximately $\pm 1\,^\circ$C.

In our plots, we report *relative* performance to that of the BB controller. If the relative number in the bar-charts is below 1, a controller performs better than BB and vice versa. We compare the results with BB as the behavior of the BB controller in the model (when replayed with historical weather) is deterministic. We omit to report values for a pure emphasis on cost as this leads the controller to (as expected) turn off all heat.

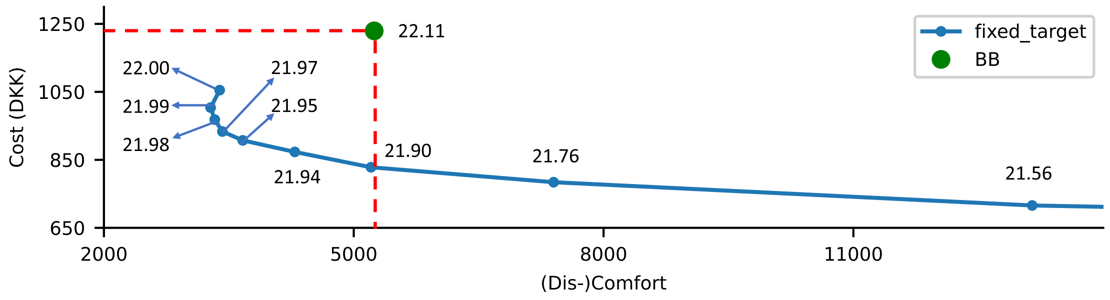


Fig. 12. The cost/comfort tradeoff with different W_{comf} settings (red lines separate the points on its left side where STRATEGO is outperforming BB w.r.t both comfort and cost).

5.2. *Evaluations with realistic observability and time*

We design a *fixed-target* STRATEGO controller for this series of experiments. In this case, all the rooms have a fixed target temperature (i.e., 22 °C). The fitness function (in Equation (1)) gets penalized when the room temperatures deviate from the set-point. In this series, we emulate a realistic setup; this implies that hidden variables are estimated, the weather forecast is stochastic, and the controller only can be applied with a 15 minutes delay from the variable observation-time, as described in Section 4.

We experiment with STRATEGO realistic observation setup and compare it with the standard BB controller in Fig. 11. The *fitness* and *cost* measures on vertical axes represent fitness function and energy cost values, respectively. However, the *discomfort* measure indirectly represents comfort; a reduction in *discomfort* improves *comfort*. Each measure is calculated by computing its values from both controllers in the formula $measure = \frac{Stratego}{BB}$ which means STRATEGO performs better than BB with a number lower than 1 and vice versa. We fix the learning parameters s.t. the wall-clock computation time is not exceeding 15 minutes on given hardware; namely 600 samples, and a reuse interval of 24 (6 hours) limiting the computation effort—named as *realistic* configuration. We present *fixed-target* STRATEGO experiments to evaluate its performance by controlling the heat-pump during four weeks (2, 6, 10, and 14) from other months.

First, we compare the results from both controllers for week 6, which is an interesting week as being the coldest. We see that the BB controller is dominated in all ways by the STRATEGO controller (see Fig. 11a). In particular, we see that at $W_{comf} = 0.4$ STRATEGO controller achieves *comparable comfort* (at least same comfort as BB) to the BB controller (Fig. 11b), but at a 33 % reduction in cost (Fig. 11c). In Fig. 11a, we observe that for any setting of the parameters, the STRATEGO-based controller dominates in the distance measure and more so with an increased comfort weight.

In Fig. 12, we have also annotated the average temperature experienced throughout the week for the *realistic* controller to provide a tangible perspective on the discomfort measure. A less than 0.10 °C deviation in the average temperature from the setpoint with $W_{comf} \geq 0.4$ again results in a 33 % reduction in cost. As the BB controller is incapable of adjusting to the time-varying cost function, the STRATEGO controller outperforms it with a focus on cost as expected. More interestingly, the benefit of using STRATEGO grows with an increased focus on comfort. Notice that Equation (1) penalizes overshooting. We hypothesize that the binary-mode of the BB entails periods of large overshooting of the target temperature and others with undershooting due to the 15-minute control interval.[1] The STRATEGO controller can instead compensate gradually and react more subtly. We observe that the *realistic* controller performs much better than BB.

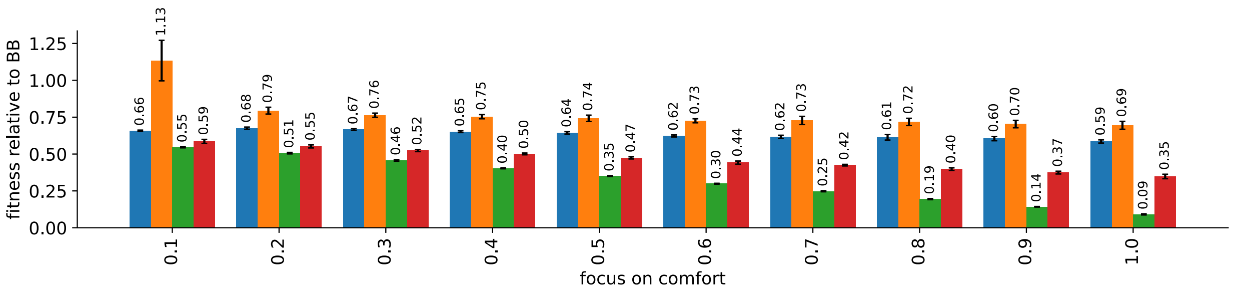
Now we discuss the performance analysis of the same controllers for the remaining weeks (2, 10, 14). The STRATEGO controller proves consistent performance improvement in all the W_{comf} settings, which can be witnessed in Fig. 11a. We can see that at $W_{comf} = 0.2$ the controller offers *comparable comfort* while providing cost reductions up to 38-53 % (see Figs. 11b, 11c). The average cost saving during the four weeks is more than 39 %. It is interesting to note that the controller responds to the change in W_{comf} values and significantly improves the comfort when we emphasize it more by increasing W_{comf}.

5.3. *Alternative control strategy approaches*

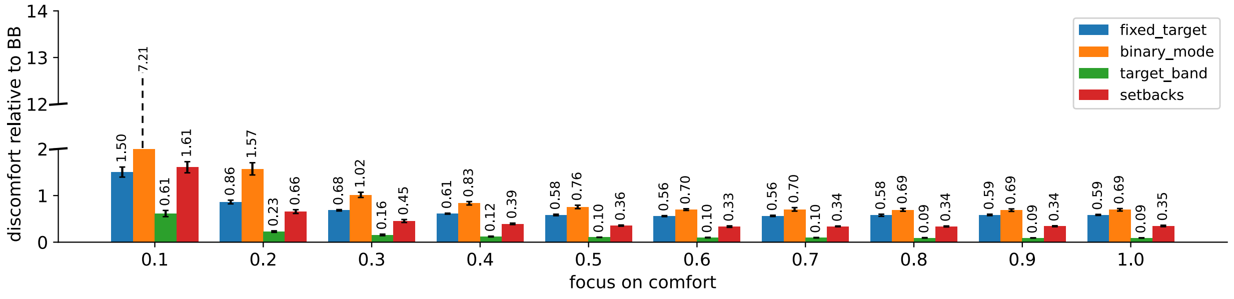
To investigate potential energy cost reduction options, we introduce several STRATEGO controllers based on various control strategy approaches. We describe the controllers in the following.

– *Target-band:* In this controller, we introduce a small range of acceptable set-points to save additional energy costs. In this case, we have a 1 °C target-band where ±0.5 °C deviation from the fixed-target is not penalized. The little flexibility in set-point provides a useful direction for the users to save significant energy while being less strict but still very close to their favorite set-point.

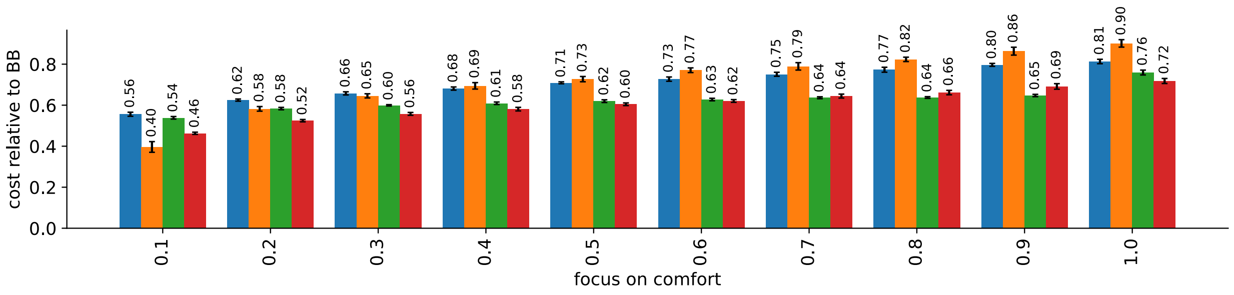
[1] Due to wear and tear on heat-pumps it is undesirable to change states too often.



(a) Fitness measured in different control strategy approaches against different values of W_{comf}



(b) Discomfort measured in different control strategy approaches against different values of W_{comf}



(c) Energy cost measured in different control strategy approaches against different values of W_{comf}

Fig. 13. March week experimental results for four different control strategies under realistic assumptions on observability and time.

- *Setbacks:* It is important to consider occupancy variations into account for better control mode. There are always some time slots when the house is unoccupied or the users prefer a much lower temperature than the normal set-point; this allows heat-pump to reduce heat output resulting in less energy use. Additionally, this approach helps in improving comfort in general. This low-temperature period is referred to as a setback. We investigate our system with two setback periods per day: the users' sleeping time (from 00:00 to 06:00) and the period of users' regular absence at home (from 08:00 to 15:00). For setback periods, we consider 19 °C instead of 22 °C.
- *Binary-mode:* The *fixed-target*, *target-band* and *setbacks* controllers operate the heap-pump under multiple intensity levels (11 in our case), the approach fits well with modern heat-pumps. However, this multi-level concept is not helpful for old heat-pump systems as they only work under the binary-mode, i.e., *on/off*. We expand the applicability of STRATEGO to attack the binary-mode heat-pump control problem and provide a reasonable set of related experiments.

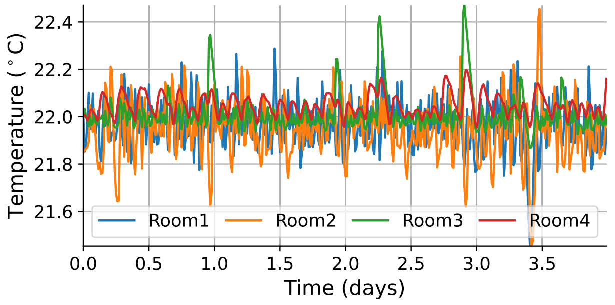
So far, we have witnessed significant energy savings with *fixed-target* controller. Now we present a series of alternative controllers (i.e., *binary-mode*, *target-band*, and *setbacks*) under realistic assumptions on observability and time. Fig. 13 shows the experimental results for week 10 controlled under four different control strategy approaches. All the controllers beat BB with all W_{comf} settings (see Fig. 13a). However, *binary-mode* controller performs a bit worse than BB only at $W_{comf} = 0.1$. With this setting, all the controllers emphasize 90 % on *cost* and only 10 % on *comfort*. Due to limited operational modes, the *binary-mode* controller has very low flexibility to choose near-optimal decisions compared to the other three controllers. The *fixed-target*, *binary-mode*, *target-band* and *setbacks* controllers gain *comparable comfort* at W_{comf} 0.2, 0.4, 0.1 and 0.2 respectively (see Fig. 13b, 13c). For achieving the *comparable comfort*, the *setbacks* controller offers the highest (47 %) and the *binary-mode* controller attains the lowest (30 %) energy cost reductions while the *target-band* controller (46 %) overperforms *fixed-target* controller (37 %) as expected.

Table 1
Money paid (in Danish Krone (DKK)) for heating during different weeks under different Stratego control approaches where the bold numbers represent Stratego saving energy cost while attaining at least *comparable comfort* to BB.

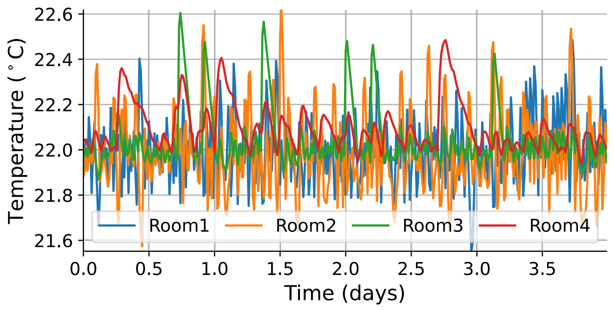
Control Method	W_{comf} **settings**									
	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
January (week 2): BB cost is 967 DKK										
fixed-target (cost)	468	549	**584**	**612**	**639**	**664**	**683**	**715**	**744**	**785**
———(discomfort)	2.08	1.04	**0.81**	**0.68**	**0.62**	**0.58**	**0.55**	**0.55**	**0.57**	**0.57**
binary-mode (cost)	283	486	555	**601**	**640**	**659**	**692**	**717**	**764**	**848**
———(discomfort)	10.6	2.05	1.13	**0.87**	**0.74**	**0.68**	**0.65**	**0.63**	**0.64**	**0.67**
target-band (cost)	462	**524**	**542**	**554**	**567**	**578**	**583**	**596**	**607**	**747**
———(discomfort)	1.06	**0.32**	**0.2**	**0.15**	**0.13**	**0.12**	**0.12**	**0.11**	**0.11**	**0.11**
setbacks (cost)	372	446	**487**	**510**	**535**	**554**	**581**	**608**	**638**	**689**
—-(discomfort)	2.63	1.03	**0.64**	**0.49**	**0.42**	**0.39**	**0.37**	**0.37**	**0.37**	**0.38**
February (week 6): BB cost is 1229 DKK										
fixed-target (cost)	534	716	784	**828**	**873**	**907**	**933**	**968**	**1004**	**1055**
———(discomfort)	9.94	2.51	1.41	**0.99**	**0.82**	**0.7**	**0.65**	**0.64**	**0.63**	**0.65**
binary-mode (cost)	194	434	631	776	873	930	**986**	**1056**	**1110**	**1197**
———(discomfort)	33.72	16.38	6.88	2.7	1.42	1.14	**0.95**	**0.81**	**0.79**	**0.8**
target-band (cost)	521	692	**748**	**786**	**808**	**827**	**835**	**854**	**863**	**1013**
———(discomfort)	8.77	1.54	**0.63**	**0.3**	**0.2**	**0.14**	**0.13**	**0.11**	**0.1**	**0.11**
setbacks (cost)	442	614	671	**699**	**727**	**756**	**791**	**826**	**868**	**934**
—-(discomfort)	9.93	2.18	1.12	**0.84**	**0.63**	**0.5**	**0.43**	**0.4**	**0.38**	**0.39**
March (week 10): BB cost is 1126 DKK										
fixed-target (cost)	626	**704**	**741**	**767**	**799**	**819**	**845**	**870**	**897**	**916**
———(discomfort)	1.5	**0.86**	**0.68**	**0.61**	**0.58**	**0.56**	**0.56**	**0.58**	**0.59**	**0.59**
binary-mode (cost)	446	656	727	**782**	**819**	**868**	**889**	**927**	**972**	**1015**
———(discomfort)	7.21	1.57	1.02	**0.83**	**0.76**	**0.7**	**0.7**	**0.69**	**0.69**	**0.69**
target-band (cost)	**606**	**657**	**675**	**686**	**698**	**707**	**717**	**718**	**729**	**855**
———(discomfort)	**0.61**	**0.23**	**0.16**	**0.12**	**0.1**	**0.1**	**0.1**	**0.09**	**0.09**	**0.09**
setbacks (cost)	521	**591**	**628**	**654**	**681**	**699**	**726**	**745**	**779**	**809**
—-(discomfort)	1.61	**0.66**	**0.45**	**0.39**	**0.36**	**0.33**	**0.34**	**0.34**	**0.34**	**0.35**
April (week 14): BB cost is 653 DKK										
fixed-target (cost)	256	**308**	**336**	**362**	**392**	**416**	**437**	**467**	**513**	**559**
———(discomfort)	1.3	**0.8**	**0.6**	**0.54**	**0.51**	**0.48**	**0.47**	**0.46**	**0.47**	**0.47**
binary-mode (cost)	214	**293**	**325**	**340**	**355**	**363**	**379**	**396**	**417**	**456**
———(discomfort)	1.96	**0.77**	**0.57**	**0.52**	**0.49**	**0.48**	**0.46**	**0.48**	**0.49**	**0.5**
target-band (cost)	**251**	**281**	**297**	**304**	**319**	**326**	**341**	**346**	**381**	**565**
———(discomfort)	**0.5**	**0.22**	**0.16**	**0.14**	**0.13**	**0.12**	**0.12**	**0.12**	**0.12**	**0.12**
setbacks (cost)	168	**221**	**248**	**270**	**297**	**320**	**348**	**386**	**425**	**474**
—-(discomfort)	1.59	**0.82**	**0.59**	**0.47**	**0.42**	**0.4**	**0.39**	**0.38**	**0.37**	**0.38**

After discussing the performance of various Stratego controllers for March week, we present the *cost* and *comfort* results from these controllers for four independent weeks in Table 1. From the table, it is clear that all the controllers (including BB) reflect the highest cost in week 6 and the lowest cost in week 14. The outside weather condition is the main reason behind the high energy use in week 6. As all the rooms have two walls open to the outside, the heating energy consumption is highly influenced by the outside weather. It means, the controllers consume more energy in week 6 to provide enough warm water to heat the rooms to a comfortable level. Moving from left to right in the table, we emphasize comfort, which is why the heating cost keeps rising and discomfort decreases. For all control strategies, discomfort suddenly drops when W_{comf} is between 0.1 and 0.3. At $W_{comf} = 0.1$, Stratego generally offers worse comfort than BB (as expected) because it emphasises energy cost (weight for cost is 0.9). By shifting W_{comf} from 0.1 to 0.2, we double the focus on comfort, but the focus on cost is reduced by only a few degrees (0.9 to 0.8). Similarly, when we move from 0.2 to 0.3, we increase focus on comfort by 50 %. As a result of these remarkable rises in comfort focus, discomfort significantly drops between 0.1 and 0.3. All the controllers are cost-wise always cheaper than BB, and with bold numbers, they are better in comfort as well. If we look at the cost in week 2, the *setbacks* controller spends 487 DKK (BB cost is 967) and *fixed-target*, *binary-mode* and *target-band* controllers spend 584, 601 and 524 DKK respectively. The *setbacks* approach achieves the overall best performance in all the weeks with average cost reductions of 49 %. The cost savings for *fixed-target*, *binary-mode* and *target-band* are 39 %, 33 % and 46 % respectively. The *setbacks* and *target-band* strategy approaches offer significant additional cost savings due to less energy demand than *fixed-target*. It is interesting to note that the *binary-mode* outperforms *fixed-target* in week 14. We conjecture that a *fixed-target* (multi-mode) controller needs more computational effort to beat a *binary-mode* controller in relatively hot weeks.

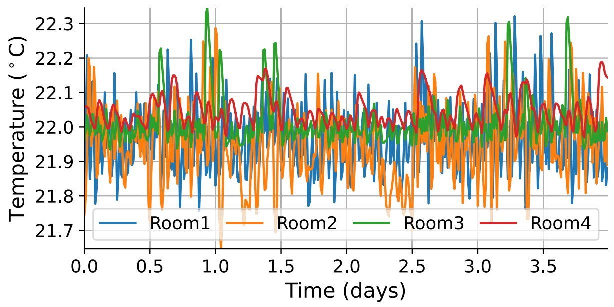
Fig. 14 reveals the visualization of the room temperatures samples recorded under the four implementend Stratego controller approaches. The temperatures for *fixed-target* controller are compact towards 22 °C due to a fixed set-point (Fig. 14a). However, we observe more over and undershooting in temperatures in case of *binary-mode* and *target-band* (Fig. 14b, Fig. 14c). This range of fluctuation is as expected due to less flexibility in adjusting the temperatures by *binary-*



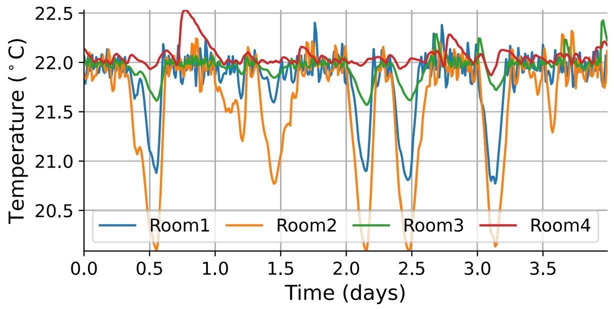
(a) Indoor temperatures for fixed-target (22 °C)



(b) Indoor temperatures for binary-mode



(c) Indoor temperatures for target-band (±0.5 °C from 22 °C)



(d) Indoor temperatures for setbacks

Fig. 14. Temperature trajectory samples from various STRATEGO controllers.

mode controller and a non-penalized 1 °C band in *target-band* controller. For the *setbacks* case, the controller drops the room temperatures according to the fixed setback time and set-point (Fig. 14d).

5.4. Sensitivity analysis of impact of measurement noise

As our experimental setup is virtual, we can study the impact of stochastic weather-prediction, hidden variables and delayed controller response; this comparison is seen in Fig. 15.

The experimental setup is similar to the *realistic* series apart from the specified changes. We experiment with four configurations:

1. full observability, allowing for delayed controller response, perfect weather-prediction and direct observation of hidden variables (i.e. an ideal scenario),
2. predicted $\tilde{T}_h^i$ & $\tilde{T}_e^i$, otherwise as full observability,
3. noisy weather, otherwise as full observability, and
4. partial observability, exactly the same setup as in the *realistic evaluation* series.

In Fig. 15, we observe that while a high emphasis on cost (which is a fully observable variable), the impact of noisy weather prediction and predicted unobservable variables has only a modest effect (2-3 %), however with more than 0.8 focus on comfort, we see the uncertainty of the weather significantly impacting the performance (up to 8 %), indicating a sensitivity of the controller towards accurate forecasts. We observe an anomaly with a 0.1 emphasis on comfort where the full observability configuration is significantly worse than the predicted scenario. While the discrepancy appears to be within measurement noise, it still warrants further investigation. We hypothesize that the squaring of the difference of the target and the actual temperature in Equation (1) leads to rare spikes in the response affecting the partition-refinement scheme deployed of STRATEGO.

5.5. Sensitivity analysis of learning parameters

STRATEGO is a sampling-based tool, and the performance (in terms of quality of the controller synthesized) is dependent on the number of samples provided. This is given directly in the number of runs (simulations) the engine is allowed to conduct.

We here experiment with reducing the overall computational effort needed for control. In Section 4.4, we introduced the *reuse* interval which allows for applying a given control strategy for an extended period. In Fig. 16a, we observe that a tighter re-computation cycle of 1 interval (every 15 minute) allows us to gain 7 % performance compared to our *realistic*

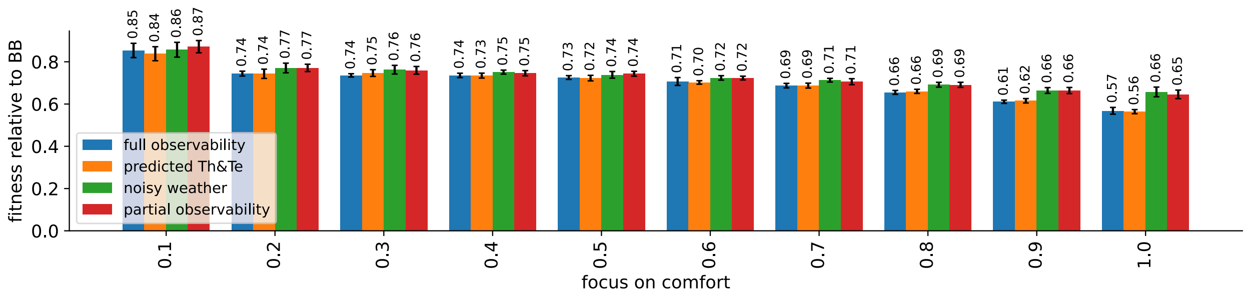
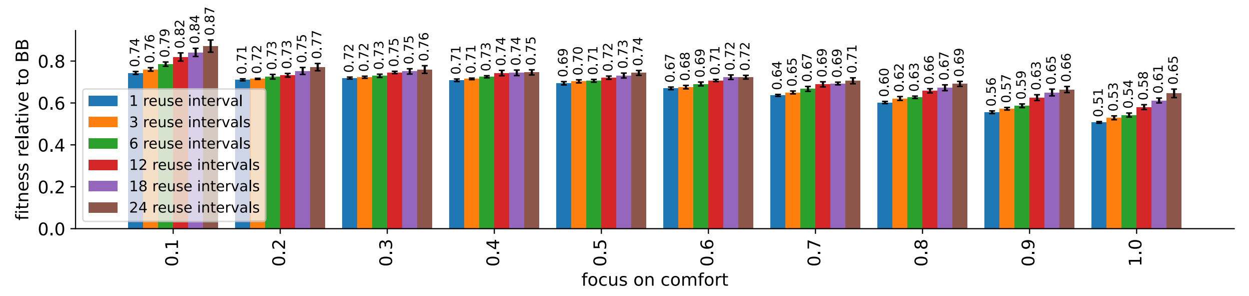
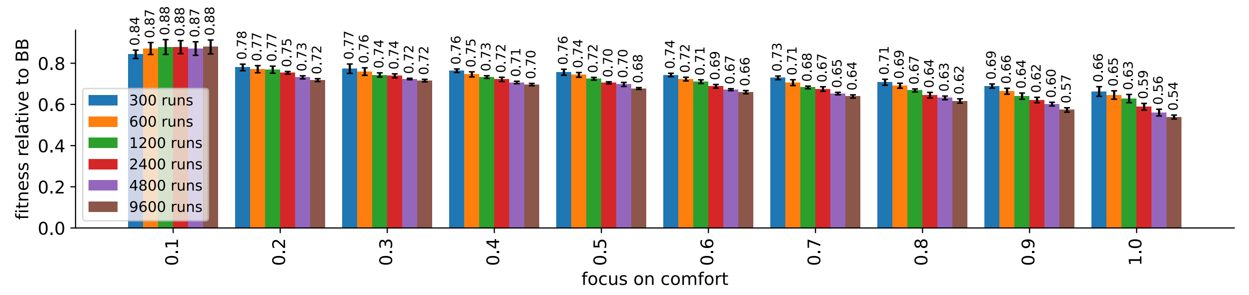


Fig. 15. February week control under different observability situations with 24 reuse intervals and 600 learning runs.



(a) February week with different reuse intervals and 600 learning runs where the number of intervals represents the number of 15-minute control intervals that use a single strategy

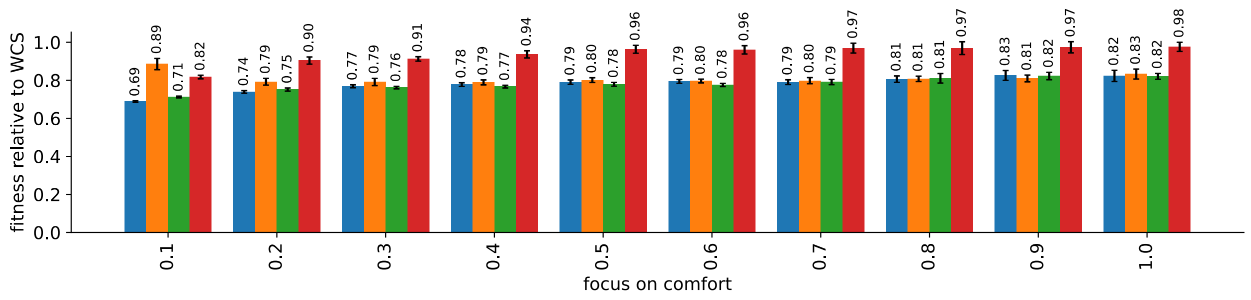


(b) February week with different number of runs and 24 reuse intervals

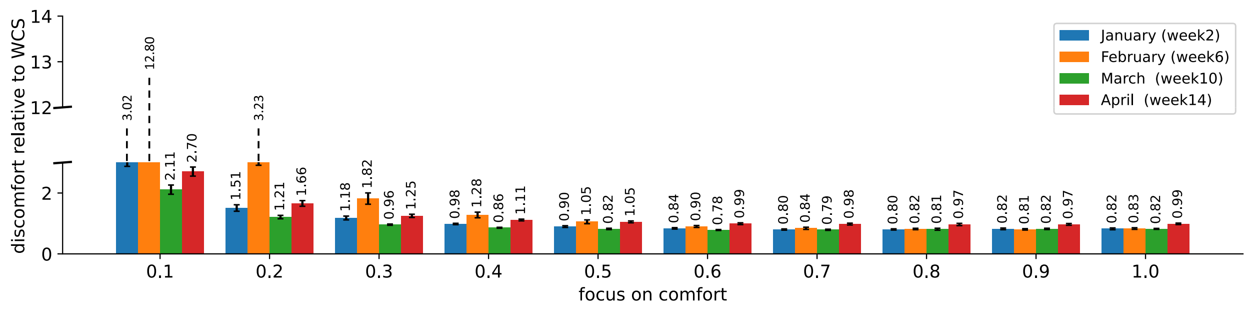
Fig. 16. Effect of choosing different sets of reuse intervals and number of runs.

series of experiments. We also observe that Stratego attains 11 % improved performance in some instances. Towards the reuse of 24 periods, we observe a drop in performance when the reuse interval is extended. On the other hand, with reuse interval 1, the controller uses the computational power 24 times more frequently than 24 reuse intervals. In Fig. 16b, we see that an increase in the number of runs (in general) improves the performance. However, with a doubled effort of 1200 runs (approx 30 minutes of computation pr. strategy), the performance in general improved by 2 % and in a rare case by 3 %. We can see a similar improvement in performance with an increase from 300 to 600 runs. To explore the edge points where the performance improvement becomes constant, and Stratego offers no further benefits, we allow the controller to synthesize the strategies with an increased budget of up to 9600 runs. We can see that Stratego constantly keeps improving the performance with a similar trend. The results reflect that the edge point still needs to be explored, even with 9600 runs, and the controller is still eager to improve the performance. Moving from 600 to 9600 runs, we increase the computational effort of simulations 16 times and get 6 % performance improvement in general and, in some cases, above 9 %. On the other hand, the time to compute a single strategy reaches to approx 240 minutes. However, a similar anomaly is observed with an 0.1 emphasis on comfort as was found in the *Measurement Noise*-series of experiments. We conjecture that this effect manifests to a higher degree with an increased learning effort.

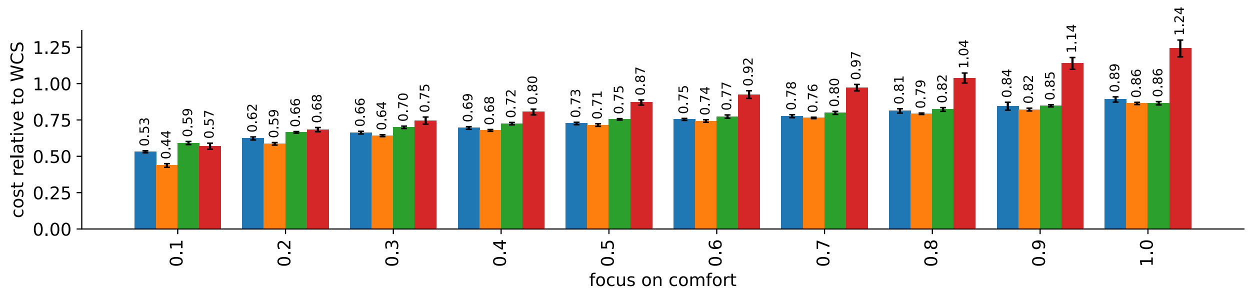
In conclusion, increasing the number of learning runs or frequency of synthesizing a strategy improves performance. However, this improvement is less significant compared to the increased computational effort and time. Thus, our *realistic* evaluation setup with 600 learning runs and 24 reuse intervals produces good results close to the best achievable performance while limiting real-time computation time below 15 minutes.



(a) Fitness measured against different values of W_{comf}



(b) Discomfort measured against different values of W_{comf}



(c) Energy cost measured against different values of W_{comf}

Fig. 17. Experimental results for four different weeks under *fixed-target* STRATEGO relative to WCS.

5.6. Comparison with weather compensation strategy

To further investigate the performance stability of STRATEGO, we implement a control strategy using the reference curve from the product sheet presented in [38]. The strategy is an industry standard and is named weather compensation strategy (WCS). In WCS, the controller obtains the outside temperature (drops/rises) information from the outdoor sensor and operates the heat-pump accordingly at a lower/higher forward water temperature to maintain the indoor room temperature. We fine-tuned the reference curve by trying different settings to suit the building construction type. In Fig. 17, we compare *fixed-target* STRATEGO (under realistic assumptions described in Section 5.2) with WCS where each measure on the vertical axis is calculated by computing its values from both controllers in the formula $measure = \frac{Stratego}{WCS}$. STRATEGO fitness-wise outperforms WCS every month, and with $W_{comf} = 0.6$, it saves (8-26 %) energy cost while improving the comfort between 1-16 %. In general, STRATEGO obtains 23-26 % cost savings in colder months and 8 % for the warmer month (where the potential cost saving is low).

6. Conclusion

We presented a tool-chain for controlling a heat-pump system in a floor-heating case study. The tool-chain offers an end-to-end solution for floor-heating applications by establishing an automatic procedure for the identification of house thermodynamics and designing UPPAAL STRATEGO controller. We compare the performance of STRATEGO controller against the traditionally used bang-bang controller. We experiment with four individual weeks having different outside weather conditions. Experimental results show that our *fixed-target* controller offers significant improvements in both user *comfort* as well as energy *cost*, even when realistic limitations on computation effort are taken into account. In particular, Stratego saves 39 % energy cost while preserving the same *comfort* as the standard bang-bang controller. We also analyze the *cost-comfort* trade-off paradigm, which shows that we can save energy costs by slightly compromising comfort.

This study also aimed to identify the impact of various control strategies on heating *cost* and *comfort* in residential buildings. To do so, we introduced *target-band* and *setbacks* strategy approaches. The approaches indicated a significant impact on heating energy consumption. We report 7 % and 10 % additional cost savings (as compared to *fixed-target*) by implementing *target-band* and *setbacks* concepts, respectively. Moreover, we design a *binary-mode* controller to examine the capability of Stratego to control the old heat-pump systems. The controller does not perform as good as a *fixed-target* multi-mode controller, but the average energy cost reduction is still quite good (33 %). We also compare the performance of *fixed-target* Stratego with an industry-standard control strategy WCS. The results show that Stratego outperforms WCS and offers up to 8-26 % cost reductions with 1-16 % improvement in comfort.

Apart from the controllers with realistic limitations on time and observability, this paper also presents a sensitivity analysis of the impact of measurement noise and learning parameters. As the *full observability* setup considers exact weather and measured hidden variables ($\tilde{T}_e^i$, $\tilde{T}_h^i$), it gains 2-3 % additional performance for fitness function in general and at some points 8 % with more focus on *comfort*. However, by using the maximum needed computational power and making a strategy in every interval (15 minutes), we can improve the controller's performance by 7-11 %. Increasing the number of learning runs is another aspect of improving performance. Using 9600 simulation runs, we increase the learning time and effort by 16 times but achieve less significant performance improvement of 6 %.

The current approach is scalable but needs some manual modifications when applying it to a different building. We need the building layout to know how the walls are shared among the rooms and outside in order to instantiate the thermodynamic equations to reflect the building. We also need one week of historical data from the building to instantiate each room's model identification process in CTSM-R. In the last step, we need to modify a few variables (representing the number of rooms) to employ the identified model in Stratego. All this, in principle, can be automated.

We believe that the results can be further improved by introducing a heat buffer and that the computational effort can be reduced by techniques such as ensemble learning. The performance of the heating system can also be improved by using users' GPS location, especially in the case of *setbacks* control strategy.

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.

Acknowledgements

We would like to thank Simon Thorsteinsson for his extensive help in acquiring base data from Dymola model for model identification. This research is partly funded by the ERC Advanced Grant LASSO, the Villum Investigator Grant S4OS, as well as DIREC: Digital Research Centre Denmark.

References

[1] M.K. Dixit, J.L. Fernández-Solís, S. Lavy, C.H. Culp, Identification of parameters for embodied energy measurement: A literature review, Energy Build. 42 (2010) 1238–1247.
[2] M.K. Agesen, K.G. Larsen, M. Mikučionis, M. Muñiz, P. Olsen, T. Pedersen, J. Srba, A. Skou, Toolchain for user-centered intelligent floor heating control, in: IECON 2016 - 42nd Annual Conference of the, IEEE Industrial Electronics Society, 2016, pp. 5296–5301.
[3] K.G. Larsen, M. Mikučionis, M. Muñiz, J. Srba, J.H. Taankvist, Online and compositional learning of controllers with application to floor heating, in: M. Chechik, J.-F. Raskin (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer, Berlin, Heidelberg, 2016, pp. 244–259.
[4] P. Vogler-Finck, R. Wisniewski, P. Popovski, Reducing the carbon footprint of house heating through model predictive control – A simulation study in Danish conditions, Sustain. Cities Soc. 42 (2018) 558–573.
[5] R. Juhl, J.K. Møller, H. Madsen, CTSMR - Continuous Time Stochastic Modeling in R, 2016, arXiv.
[6] L. Ljung, Matlab system identification toolbox—getting started guide r2016a, Mathworks (Ed.), Mathworks (2016).
[7] A. David, P.G. Jensen, K.G. Larsen, M. Mikučionis, J.H. Taankvist, Uppaal stratego, in: International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2015, pp. 206–211.
[8] A. David, P.G. Jensen, K.G. Larsen, A. Legay, D. Lime, M.G. Sørensen, J.H. Taankvist, On time with minimal expected cost!, in: International Symposium on Automated Technology for Verification and Analysis, Springer, 2014, pp. 129–145.
[9] R. Hermansen, K. Smith, J.E. Thorsen, J. Wang, Y. Zong, Model predictive control for a heat booster substation in ultra low temperature district heating systems, Energy 238 (2022) 121631.
[10] S. Privara, J. Cigler, Z. Váňa, F. Oldewurtel, C. Sagerschnig, E. Žáčeková, Building modeling as a crucial part for building predictive control, Energy Build. 56 (2013) 8–22.
[11] F. Ferracuti, A. Fonti, L. Ciabattoni, S. Pizzuti, A. Arteconi, L. Helsen, G. Comodi, Data-driven models for short-term thermal behaviour prediction in real buildings, Appl. Energy 204 (2017) 1375–1387.
[12] A. Fonti, G. Comodi, S. Pizzuti, A. Arteconi, L. Helsen, Low order grey-box models for short-term thermal behavior prediction in buildings, Energy Proc. 105 (2017) 2107–2112.
[13] G. Reynders, J. Diriken, D. Saelens, Quality of grey-box models and identified parameters as function of the accuracy of input and observation signals, Energy Build. 82 (2014) 263–274.

[14] K. Vinther, T. Green, S.Ø. Jensen, J.D. Bendtsen, Predictive control of hydronic floor heating systems using neural networks and genetic algorithms, IFAC-PapersOnLine 50 (2017) 7381–7388.
[15] N. Nassif, Modeling and optimization of hvac systems using artificial neural network and genetic algorithm, Build. Simul. 7 (3) (2014) 237–245.
[16] S. Harasty, S. Lambeck, A. Cavaterra, Model predictive control for preventive conservation using artificial neural networks, in: 12th REHVA World Congress, Aalborg, Denmark, 2016.
[17] I.R. Hasrat, P.G. Jensen, K.G. Larsen, J. Srba, End-to-end heat-pump control using continuous time stochastic modelling and uppaal stratego, in: Y. Aït-Ameur, F. Crăciun (Eds.), Theoretical Aspects of Software Engineering, Springer International Publishing, Cham, 2022, pp. 363–380.
[18] H. Golmohamadi, K. Guldstrand Larsen, P. Gjøl Jensen, I. Riaz Hasrat, Optimization of power-to-heat flexibility for residential buildings in response to day-ahead electricity price, Energy Build. 232 (2021) 110665.
[19] S. Williams, M. Short, T. Crosbie, On the use of thermal inertia in building stock to leverage decentralised demand side frequency regulation services, Appl. Therm. Eng. 133 (2018) 97–106.
[20] P. Ponnaganti, J.R. Pillai, B. Bak-Jensen, P. Vogler-Finck, Intelligent operation of thermal storage systems based heat pump pool for cost efficiency, in: 2022 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), IEEE, 2022, pp. 01–05.
[21] J.W. Moon, S.-H. Han, Thermostat strategies impact on energy consumption in residential buildings, Energy Build. 43 (2011) 338–346.
[22] J.W. Moon, S.K. Jung, Algorithm for optimal application of the setback moment in the heating season using an artificial neural network model, Energy Build. 127 (2016) 859–869.
[23] M. Kersken, H. Sinnesbichler, Simulation study on the energy saving potential of a heating control system featuring presence detection and weather forecasting, Fraunhofer IBP Rapport 527 (2013) 40.
[24] S. Østergaard Jensen, OPSYS tools for investigating energy flexibility in houses with heat pumps, https://www.annex67.org/media/1838/report-opsys-flexibilitet.pdf, 2018.
[25] Dayssault systems. dymola (dynamic modeling laboratory) systems engineering, https://www.3ds.com/products-services/catia/products/dymola/, October 2022.
[26] R. Juhl, N.R. Kristensen, P. Bacher, J. Kloppenborg, H. Madsen, Grey-box modeling of the heat dynamics of a building with CTSM-R, http://ctsm.info/pdfs/examples/building2.pdf, 2017.
[27] M. Jaeger, G. Bacci, G. Bacci, K.G. Larsen, P.G. Jensen, Approximating euclidean by imprecise Markov decision processes, in: International Symposium on Leveraging Applications of Formal Methods, Springer, 2020, pp. 275–289.
[28] E. Carrascal, I. Garrido, A.J. Garrido, J.M. Sala, Optimization of the heating system use in aged public buildings via model predictive control, Energies 9 (2016) 251.
[29] C.A. Thilker, H.G. Bergsteinsson, P. Bacher, H. Madsen, D. Calì, R.G. Junker, Non-linear model predictive control for smart heating of buildings, E3S Web of Conferences, vol. 246, EDP Sciences, 2021, p. 09005.
[30] X. Yu, S. You, H. Cai, L. Georges, P. Bacher, Data-driven modelling and optimal control of domestic electric water heaters for demand response, in: The International Symposium on Heating, Ventilation and Air Conditioning, Springer, 2020, pp. 77–86.
[31] K.G. Larsen, P. Pettersson, W. Yi, Uppaal in a nutshell, Int. J. Softw. Tools Technol. Transf. 1 (1997) 134–152.
[32] G. Behrmann, A. David, K.G. Larsen, J. Håkansson, P. Pettersson, W. Yi, M. Hendriks, Uppaal 4.0, IEEE Computer Society, 2006.
[33] P. Bulychev, A. David, K.G. Larsen, M. Mikučionis, D.B. Poulsen, A. Legay, Z. Wang, Uppaal-SMC: Statistical model checking for priced timed automata, arXiv preprint, arXiv:1207.1272, 2012.
[34] G. Behrmann, A. Cougnard, A. David, E. Fleury, K.G. Larsen, D. Lime, Uppaal-tiga: Time for playing games!, in: International Conference on Computer Aided Verification, Springer, 2007, pp. 121–125.
[35] P.G. Jensen, K.G. Larsen, A. Legay, U. Nyman, Integrating tools: Co-simulation in uppaal using fmi-fmu, in: 22nd International Conference on Engineering of Complex Computer Systems (ICECCS), IEEE, 2017, pp. 11–19.
[36] G. Behrmann, A. David, K.G. Larsen, A tutorial on uppaal, in: Formal methods for the design of real-time systems, Springer, 2004, pp. 200–236.
[37] I.R. Hasrat, P.G. Jensen, K.G. Larsen, J. Srba, Artefact for "A Toolchain for Domestic Heat-Pump Control Using Uppaal Stratego", https://doi.org/10.5281/zenodo.7413575, 2022.
[38] Control technology: weather compensated controls (Viessmann: climate of innovation), https://viessmanndirect.co.uk/files//8e57dbc7-8a10-4065-bcc6-a27700ee752a/weather_comp.pdf, 2023.