Delft University of Technology

Delft Center for Systems and Control

Technical report 16-028

Reinforcement learning applied to an electric water heater: From theory to practice*

F. Ruelens, B.J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans

If you want to cite this report, please use the following reference instead:

F. Ruelens, B.J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans, "Reinforcement learning applied to an electric water heater: From theory to practice," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3792–3800, 2018. doi:10.1109/TSG.2016.2640184

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/16_028.html

Reinforcement Learning Applied to an Electric Water Heater: From Theory to Practice

F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans

***Abstract*—Electric water heaters have the ability to store energy in their water buffer without impacting the comfort of the end user. This feature makes them a prime candidate for residential demand response. However, the stochastic and non-linear dynamics of electric water heaters, makes it challenging to harness their flexibility. Driven by this challenge, this paper formulates the underlying sequential decision-making problem as a Markov decision process and uses techniques from reinforcement learning. Specifically, we apply an auto-encoder network to find a compact feature representation of the sensor measurements, which helps to mitigate the curse of dimensionality. A well-known batch reinforcement learning technique, fitted Q-iteration, is used to find a control policy, given this feature representation. In a simulation-based experiment using an electric water heater with 50 temperature sensors, the proposed method was able to achieve good policies much faster than when using the full state information. In a lab experiment, we apply fitted Q-iteration to an electric water heater with eight temperature sensors. Further reducing the state vector did not improve the results of fitted Q-iteration. The results of the lab experiment, spanning 40 days, indicate that compared to a thermostat controller, the presented approach was able to reduce the total cost of energy consumption of the electric water heater by 15%.**

***Index Terms*—Auto-encoder network, demand response, electric water heater, fitted Q-iteration, machine learning, reinforcement learning.**

I. Introduction

THE share of renewable energy sources is expected to reach 25% of the global power generation portfolio by 2020 [1]. The intermittent and stochastic nature of most renewable energy sources, however, makes it challenging to integrate these sources into the power grid. Successful integration of these sources requires flexibility on the demand side through demand response programs. Demand response programs enable end users with flexible loads to adapt their consumption profile in response to an external signal. Dynamic pricing, reflecting the dynamic nature of the underlying cost of electricity, is one way to engage end users [2]. This way, end users are incentivized to modify their demand pattern in order to achieve a more efficient power system in which the operation of renewable energy is integrated [3].

A prominent example of flexible loads are electric water heaters with a hot water storage tank [4], [5]. These loads have the ability to store energy in their water buffer without impacting the comfort level of the end user. In addition to having significant flexibility, electric water heaters can consume about 2 MWh per year for a household with a daily hot water demand of 100 liters [6]. As a result, electric water heaters are a prime candidate for residential demand response programs. Previously, the flexibility offered by electric water heaters has been used for frequency control [7], local voltage control [8], and energy arbitrage [9], [10]. Amongst others, two prominent control paradigms in the demand response literature on electric water heaters are model-based approaches and reinforcement learning.

Perhaps the most researched control paradigm applied to demand response are model-based approaches, such as Model Predictive Control (MPC) [9], [10], [11]. Most MPC strategies use a gray-box model, based on general expert knowledge of the underlying system dynamics, requiring a system identification step. Given this mathematical model, an optimal control action can be found by solving a receding horizon problem [12]. In general, the implementation of MPC consists of several critical steps, namely, selecting accurate models, estimating the model parameters, estimating the state of the system, and forecasting of the exogenous variables. All these steps make MPC an expensive technique, the cost of which needs to be balanced out by the possible financial gains [13]. Moreover, possible model errors resulting from an inaccurate model or forecast, can effect the stability of the MPC controller [14], [15].

In contrast to MPC, Reinforcement Learning (RL) techniques [16] do not require expert knowledge and consider their environment as a black-box. RL techniques enable an agent to learn a control policy by interacting with its environment, without the need to use modeling and system identification techniques. In [17], Ernst *et al.* state that the trade-off in applying MPC and RL mainly depends on the quality of the expert knowledge about the system dynamics that could be exploited in the context of system identification. In most residential demand response applications, however, expert knowledge about the system dynamics or future disturbances might be unavailable or might be too expensive to obtain relative to the expected financial gain. In this context, RL techniques are an excellent candidate to build a general purpose agent that can be applied to any demand response application.

This paper proposes a learning agent that minimizes the cost of energy consumption of an electric water heater. The agent measures the state of its environment through a set of sensors that are connected along the water buffer of the electric water heater. However, learning in a high-dimension

F. Ruelens and R. Belmans are with the Department of Electrical Engineering, KU Leuven/EnergyVille, Belgium (frederik.ruelens@esat.kuleuven.be).

B. J. Claessens is with the Energy Department of Vito/EnergyVille, Belgium (bert.claessens@vito.be).

S. Quaiyum is with the Department of Electrical Engineering, Uppsala University, Sweden.

B. De Schutter and R. Babuška are with the Delft Center for Systems and Control, Delft University of Technology, The Netherlands.

state space can significantly impact the learning rate of the RL algorithm. This is known as the "curse of dimensionality". A popular approach to mitigate its effects is to reduce the dimensionality of the state space during a pre-processing step. Inspired by the work of [18], this paper applies an auto-encoder network to reduce the dimensionality of the state vector. By so doing, this paper makes following contributions: (1) we demonstrate how a well-established RL technique, fitted Q-iteration, can be combined with an auto-encoder network to minimize the cost of energy consumption of an electric water heater; (2) in a simulation-based experiment, we assess the performance of different state representations and batch sizes; (3) we successfully apply an RL agent to an electric water heater equipped with eight temperature sensors in a lab environment (Fig. 1).

Although, the cost of obtaining expert knowledge would reduce for mass-produced products, such as an electric water heater, it is our intention to set a generic example in the context of demand response that can be extended to other flexible loads.

The remainder of this paper is organized as follows. Section II gives a non-exhaustive literature overview of RL related to demand response. Section III maps the considered demand response problem to a Markov decision process. Section IV describes how an auto-encoder network can be used to find a low-dimensional state representation, followed by a description of the fitted Q-iteration algorithm in Section V. Section VI presents the results of the simulation-based experiments and Section VII presents the results of the lab experiment. Finally, Section VIII draws conclusions and discusses future work.

II. Reinforcement learning

This section gives a non-exhaustive overview of recent developments related to Reinforcement Learning (RL) and demand response. Perhaps the most widely used model-free RL technique applied to a demand response setting is standard Q-learning [19], [20], [21], [22]. After each interaction with the environment, Q-learning uses temporal difference learning [16] to update its state-action value function or Q-function. A major drawback of Q-learning is that the given observation is discarded after each update. As a result, more interactions are needed to spread already known information through the state space. This inefficient use of information limits the application of Q-learning to real-world applications.

In contrast to Q-learning, batch RL techniques [23], [24], [25] are more data efficient, since they store and reuse past interactions. As a result, batch RL techniques require less interactions with their environment, which makes them more practical for real-world applications, such as demand response. Perhaps the most popular batch RL technique which has been applied to a wide range of applications [18], [26], [27] is fitted Q-iteration developed by Ernst *et al.* [23]. Fitted Q-iteration iteratively estimates the Q-function given a fixed batch of past interactions. An online version that uses a neural network, neural fitted Q-iteration, has been proposed by Riedmiller *et al.* in [24]. Finally, an interesting alternative is to combine experience replay to an incremental RL technique such as Q-learning or SARSA [28]. In [29], the authors demonstrate how fitted Q-iteration can be used to control a cluster of electric water heaters. The results indicate that fitted Q-iteration was able to reduce the cost of energy consumption of a cluster of 100 electric water heaters after a learning period of 40 days. In addition, [30] shows how fitted Q-iteration can be extended to reduce the cost of energy consumption of a heat-pump thermostat given that a forecast of the outside temperature is provided.

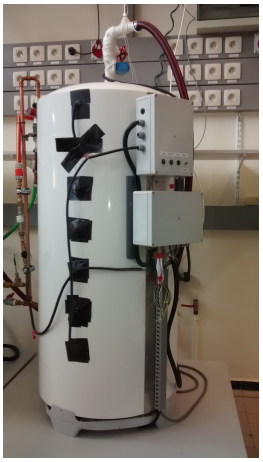

Fig. 1. Setup of the electric water heater used during the lab experiment. Eight temperature sensors were placed under the insulation material of the buffer tank.

A promising alternative to the previously mentioned model-free techniques are model-based or model-assisted RL techniques. For example, the authors of [31] present a model-based policy search method that learns a Gaussian process to model uncertainties. In addition, inspired by [32], the authors of [33] demonstrate how a model-assisted batch RL technique can be applied to control a building heating system.

III. Problem formulation

The aim of this work is to develop a controller or agent that minimizes the cost of energy consumption of an electric water heater, given an external price profile. This price profile is provided to the agent at the start of each day. The agent can measure the temperature of the water buffer through a set of temperature sensors that are connected along the hull of the buffer tank. Following the approach presented in [30], the electric water heater is equipped with a backup controller that overrules the control action from the agent when the safety or comfort constraints of the end user are violated. A challenge in developing such an agent is that the dynamics of the electric water heater, the future hot water demand and the settings of the backup controller are unknown to the agent. To overcome this challenge, this paper leverages on the previous work of [18], [23], [30] and applies techniques from RL.

A. Markov decision process framework

To apply RL, this paper formulates the underlying sequential decision-making problem of the learning agent as a Markov decision process formulation. The Markov decision process formulation is defined by its d-dimensional state space

$X \subset \mathbb{R}^d$, its action space $U \subset \mathbb{R}$, its stochastic discrete-time transition function f, and its cost function ρ. The optimization horizon is considered finite, comprising $T \in \mathbb{N} \setminus \{0\}$ steps, where at each discrete time step k, the state evolves following:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, u_k, \boldsymbol{w}_k) \quad \forall k \in \{1, ..., T-1\}, \tag{1}$$

where a random disturbance $\boldsymbol{w}_k$ is drawn from a conditional distribution $p_{\mathcal{W}}(\cdot|\boldsymbol{x}_k)$, $u_k \in U$ is the control action and $\boldsymbol{x}_k \in X$ the state. Associated with each state transition, a cost c_k is given by:

$$c_k = \rho(\boldsymbol{x}_k, u_k, \boldsymbol{w}_k) \quad \forall k \in \{1, ..., T\}. \tag{2}$$

The goal of the learning agent is to find an optimal control policy $h^* : X \to U$ that minimizes the expected T-stage return for any state in the state space. The expected T-stage return J_T^h starting from $\boldsymbol{x}_1$ and following a policy h is defined as follows:

$$J_T^h(\boldsymbol{x}_1) = \mathop{\mathbf{E}}_{p_{\mathcal{W}}(\cdot|\boldsymbol{x}_k)} \left[\sum_{k=1}^{T} \rho(\boldsymbol{x}_k, h(\boldsymbol{x}_k), \boldsymbol{w}_k) \right], \tag{3}$$

where $\mathbf{E}$ denotes the expectation operator over all possible stochastic realizations.

A more convenient way to characterize a policy is by using a state-action value function or Q-function:

$$Q^h(\boldsymbol{x}, u) = \mathop{\mathbf{E}}_{p_{\mathcal{W}}(\cdot|\boldsymbol{x})} \left[\rho(\boldsymbol{x}, u, \boldsymbol{w}) + J_T^h(f(\boldsymbol{x}, h(\boldsymbol{x}), \boldsymbol{w})) \right], \tag{4}$$

which indicates the cumulative return starting from state $\boldsymbol{x}$ and by taking action u and following h thereafter.

The optimal Q-function corresponds the best Q-function that can be obtained by any policy:

$$Q^*(\boldsymbol{x}, u) = \min_h Q^h(\boldsymbol{x}, u). \tag{5}$$

Starting from an optimal Q-function for every state-action pair, the optimal policy h^* is calculated as follows:

$$h^*(\boldsymbol{x}) \in \arg\min_{u \in U} Q^*(\boldsymbol{x}, u), \tag{6}$$

where Q^* satisfies the Bellman optimality equation [34]:

$$Q^*(\boldsymbol{x}, u) = \mathop{\mathbf{E}}_{w \sim p_{\mathcal{W}}(\cdot|x)} \left[\rho(\boldsymbol{x}, u, \boldsymbol{w}) + \min_{u' \in U} Q^*(f(\boldsymbol{x}, u, \boldsymbol{w}), u') \right]. \tag{7}$$

Following the notation introduced in [30], the next three paragraphs give a description of the state, the action, and the cost function tailored to an electric water heater.

B. Observable state vector

The observable state vector of an electric water heater contains a time-related component and a controllable component. The time-related component $\boldsymbol{x}^{\mathrm{t}}$ describes the part of the state related to timing, which is relevant for the dynamics of the system. Specifically, the tap water demand of the end user is considered to have diurnal and weekly patterns. As such, the time-related component contains the day of the week and the quarter in the day. The controllable component $\boldsymbol{x}^{\mathrm{ph}}$ represents physical state information that is measured locally and is influenced by the control action. The controllable component contains the temperature measurements of the n_{s} sensors that are connected along the hull of the storage tank. The observable state vector is given by:

$$\boldsymbol{x}_k = (\underbrace{d, t}_{\boldsymbol{x}_k^{\mathrm{t}}}, \underbrace{T_k^1, \ldots, T_k^i, \ldots, T_k^{n_{\mathrm{s}}}}_{\boldsymbol{x}_k^{\mathrm{ph}}}), \tag{8}$$

where $d \in \{1, \ldots, 7\}$ is the current day of the week, $t \in \{1, \ldots, 96\}$ is the quarter in the day and T_k^i denotes the temperature measurements of sensor i at time step k.

C. Control action

The learning agent can control the heating element of the electric water heater with a binary control action $u_k \in \{0, 1\}$, where 0 indicates off and 1 on. However, the backup mechanism, which enacts the comfort and safety constraints of the end user, can overrule this control action of the learning agent. The function $B : X \times U \to U^{\mathrm{ph}}$ maps the control action $u_k \in U$ to a physical action $u_k^{\mathrm{ph}} \in U^{\mathrm{ph}}$ according to:

$$u_k^{\mathrm{ph}} = B(x_k, u_k, \boldsymbol{\theta}), \tag{9}$$

where the vector $\boldsymbol{\theta}$ defines the safety and user-defined comfort settings of the backup controller. In order to have a generic approach we assume that the logic of the backup controller is unknown to the learning agent. However, the learning agent can measure the physical action u_k^{ph} enforced by the backup controller (see Fig. 2), which is required to calculate the cost.

The logic of the backup controller of the electric water heater is defined as:

$$B(\boldsymbol{x}, u, \boldsymbol{\theta}) = \begin{cases} P^{\mathrm{e}} & \text{if} \quad x_{\mathrm{soc}}(\boldsymbol{x}, \boldsymbol{\theta}) \leq \underline{x}_{\mathrm{soc}}(\boldsymbol{\theta}) \\ uP^{\mathrm{e}} & \text{if} \quad \underline{x}_{\mathrm{soc}}(\boldsymbol{\theta}) < x_{\mathrm{soc}}(\boldsymbol{x}, \boldsymbol{\theta}) < \bar{x}_{\mathrm{soc}}(\boldsymbol{\theta}), \\ 0 & \text{if} \quad x_{\mathrm{soc}}(\boldsymbol{x}, \boldsymbol{\theta}) \geq \bar{x}_{\mathrm{soc}}(\boldsymbol{\theta}) \end{cases} \tag{10}$$

where P^{e} is the electrical power rating of the heating element, $x_{\mathrm{soc}}(\boldsymbol{x}, \boldsymbol{\theta})$ is the current state of charge and $\underline{x}_{\mathrm{soc}}(\boldsymbol{\theta})$ and $\bar{x}_{\mathrm{soc}}(\boldsymbol{\theta})$ are the upper and lower bounds for the state of charge. A detailed description of how the state of charge is calculated can by found in [5].

D. Cost function

At the start of each optimization period $T\Delta t$, the learning agent receives a price vector $\boldsymbol{\lambda} = \{\lambda_k\}_{k=1}^T$ for the next T time steps. At each time step, the agent receives a cost c_k according to:

$$c_k = u_k^{\mathrm{ph}} \lambda_k \Delta t, \tag{11}$$

where λ_k is the electricity price during time step k, and Δt the length of one control period.

IV. Batch of four-tuples

Generally, batch RL techniques estimate the Q-function based on a batch of four-tuples $(\boldsymbol{x}_l, u_l, \boldsymbol{x}'_l, c_l)$. This paper, however, considers the following batch of four-tuples:

$$\mathcal{F} = \{(\boldsymbol{x}_l, u_l, \boldsymbol{x}'_l, u_l^{\mathrm{ph}}), l = 1, \ldots, \#\mathcal{F}\}, \tag{12}$$

where for each l, the next state $\boldsymbol{x}'_l$, and the physical action u_l^{ph} have been obtained as a result of taking control action u_l in

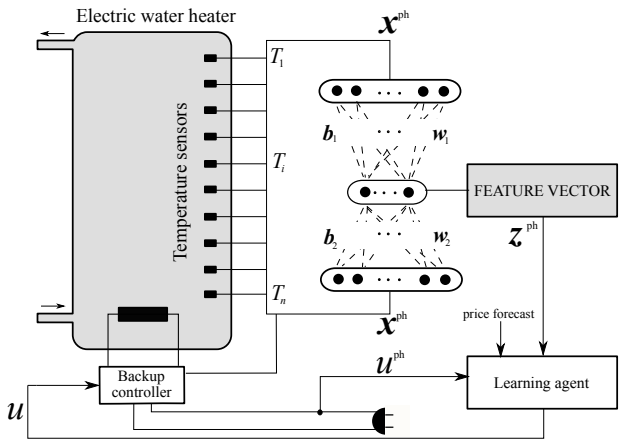

Fig. 2. Setup of the simulation-based experiment. An auto-encoder network is used to find a compact representation of the temperature measurements.

the state $\boldsymbol{x}_l$. Note that, $\mathcal{F}$ does not include the observed cost c_l, since the cost depends on the price vector that is provided to the learning agent at the start of each day.

As defined by (8), $\boldsymbol{x}_l$ contains all temperature measurements of the sensors connected to the hull of the water buffer. Learning in a high-dimensional state space requires more observations from the environment to estimate the Q-function, as more tuples are needed to cover the state-action space. This is known as the "curse of dimensionality". This curse becomes even more pronounced in practical applications, where each observation corresponds to a "real" interation with the environment.

A pre-processing step can be used to find a compact and more efficient representation of the state space and can help to converge to a good policy much faster [35]. A popular technique to find a compact representation is to use a handcrafted feature vector based on insights of the considered control problem [36]. Alternative approaches that do not require prior knowledge are unsupervised feature learning algorithms, such as auto-encoders [18] or a principal component analysis [35].

As illustrated in Fig. 2, this paper demonstrates how an auto-encoder can be used to find a compact representation of the sensory input data. An auto-encoder network is an artificial neural network, commonly used in deep learning [37], for learning efficient features by mapping its output back to its input. By selecting a lower number of neurons in the middle hidden layer than in the input layer $p < d$, the auto-encoder can be used to reduce the dimensionality of the input data. The reduced feature vector $\boldsymbol{z}_l \in Z \subset \mathbb{R}^p$ can be calculated as follows:

$$\boldsymbol{z}_l = (\boldsymbol{x}_l^{\mathrm{t}}, \Phi_{\mathrm{enc}}(\boldsymbol{x}_l^{\mathrm{ph}}, \boldsymbol{w}, \boldsymbol{b})), \tag{13}$$

where $\boldsymbol{w}$ and $\boldsymbol{b}$ denote the weights and the biases that connect the input layer with the middle hidden layer of the auto-encoder network. The function $\Phi_{\mathrm{enc}} : X \to Z$ is an encoder function and maps the observed state vector $\boldsymbol{x}_l$ to the feature vector $\boldsymbol{z}_l$. To train the weights of the auto-encoder, a conjugate gradient descent algorithm is used as presented in [38].

In the next section, fitted Q-iteration is used to find a policy $h : Z \to U$ that maps every feature vector to a control action

Algorithm 1 Fitted Q-iteration [23] using feature vectors.

Input: $\mathcal{R} = \left\{(\boldsymbol{z}_l, u_l, \boldsymbol{z}'_l, u_l^{\mathrm{ph}}), l = 1, \ldots, \#\mathcal{R}\right\}, \{\lambda_t\}_{t=1}^{T}$

1: initialize $\widehat{Q}_0$ to zero
2: **for** $N = 1, \ldots, T$ **do**
3: **for** $l = 1, \ldots, \#\mathcal{R}$ **do**
4: $c_l \leftarrow u_l^{\mathrm{ph}} \lambda_t$ ▷ *where t is to the quarter in the day of the time-related component $\boldsymbol{x}_l^t = (d, t)$ of state $\boldsymbol{z}_l$*
5: $Q_{N,l} \leftarrow c_l + \min_{u \in U} \widehat{Q}_{N-1}(\boldsymbol{z}'_l, u)$
6: **end for**
7: use a regression technique to obtain $\widehat{Q}_N$ from $\mathcal{T}_{\mathrm{reg}} = \left\{\left((\boldsymbol{z}_l, u_l), Q_{N,l}\right), l = 1, \ldots, \#\mathcal{R}\right\}$.
8: **end for**

Output: $\widehat{Q}^* = \widehat{Q}_T$

using batch $\mathcal{R}$:

$$\mathcal{R} = \{(\boldsymbol{z}_l, u_l, \boldsymbol{z}'_l, u_l^{\mathrm{ph}}), l = 1, \ldots, \#\mathcal{R}\}, \tag{14}$$

which consists of feature vectors with a dimensionality p.

Since we apply the auto-encoder on the input data of the supervised learning algorithm, we assume that all input data is equally important. As such, it is possible that we ignore low-variance yet potentially useful components during the learning process. A possible route of future work would be to add a regularization term to the regression algorithm of the supervised learning algorithm to prevent the risk of overfitting without the risk of ignoring potentially important data.

V. Fitted Q-iteration

This section describes the learning algorithm and the exploration strategy of the agent based on the batch of feature vectors $\mathcal{R}$ presented in the previous section.

A. Fitted Q-iteration

Fitted Q-iteration iteratively builds a training set $\mathcal{T}_{\mathrm{reg}}$ with all state-action pairs $(\boldsymbol{z}, u)$ in $\mathcal{R}$ as the input. The target values consist of the corresponding Q-values, based on the approximation of the Q-function of the previous iteration. In the first iteration ($N = 1$), the Q-values approximate the expected cost (line 5 in Algorithm 1). In the subsequent iterations, Q-values are updated using an approximation of the Q-function $\widehat{Q}_{N-1}$ of the previous iteration $N - 1$ and information about the current cost c_l and successor state $\boldsymbol{z}'_l$ in each tuple. As a result, Algorithm 1 needs T iterations until the Q-function contains all information about the successor states. Note that, the cost corresponding to each tuple is recalculated using price vector $\boldsymbol{\lambda}$ that is provided at the start of the day (line 4 in Algorithm 1). As a result, the algorithm can reuse past experiences to find a control policy for the next day. Following [23], Algorithm 1 applies an ensemble of extremely randomized trees as a regression algorithm to estimate the Q-function. An empirical study of the accuracy and convergence properties of extremely randomized trees can be found in [23]. However, in principle, any regression algorithm, such as neural networks [25], [26], can be used to estimate the Q-function.

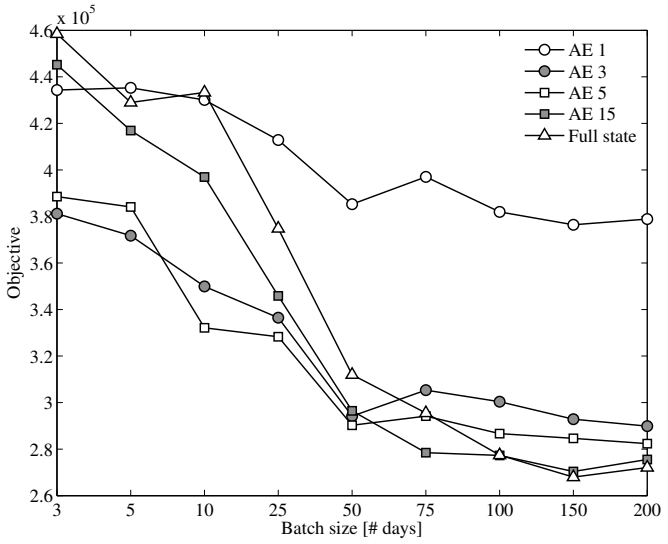

Fig. 3. Simulation-based results of fitted Q-iteration using five state representations and different batch sizes. The full state contains 50 temperature measurements. A non-linear dimensionality reduction with Auto-Encoder (AE) is used to find a compact representation of the temperature measurements. Each marker point represents the average result of 100 simulation runs.

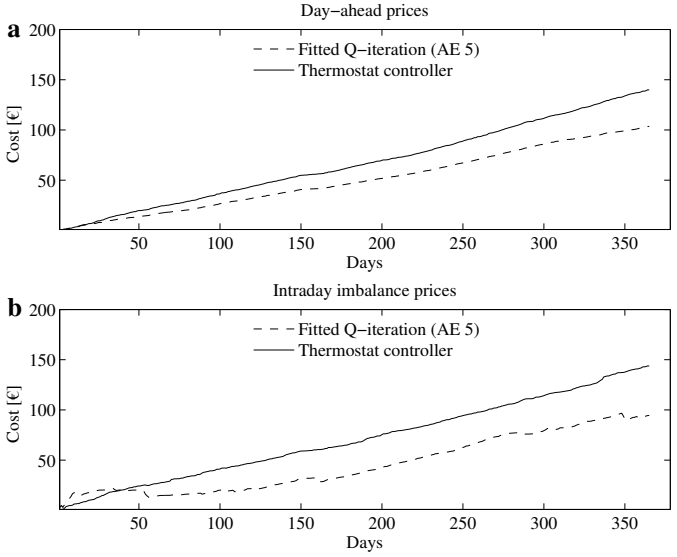

Fig. 4. Cumulative energy cost of fitted Q-iteration with a non-linear dimensionality reduction and of the thermostat controller. Results for one year using day-ahead prices (**a**) and imbalance prices (**b**).

B. Boltzmann exploration

During the day, the learning agent uses a Boltzmann exploration strategy [39] and selects an action with the following probability:

$$P\left(u|\boldsymbol{z}\right) = \frac{e^{\widehat{Q}^*(\boldsymbol{z},u)/\tau_d}}{\Sigma_{u' \in U} e^{\widehat{Q}^*(\boldsymbol{z},u')/\tau_d}}, \tag{15}$$

where τ_d is the Boltzmann temperature at day d, $\widehat{Q}^*$ is the Q-function from Algorithm 1 and $\boldsymbol{z}$ is the current feature vector measured by the learning agent. If $\tau_d \to 0$, the exploration will decrease and the policy will become greedier. Thus by starting with a high τ_d the exploration starts completely random, however as τ_d decreases the policy directs itself to the most interesting state-action pairs. In the evaluation experiments, $\widehat{Q}^*$ in (15) is linearly scaled between $[0, 100]$ and the τ_1 is set to 100 at the start of the experiment, which will result in an equal probability for all actions. The Boltzmann temperature is updated as follows $\tau_d = \tau_{d-1} - \Delta\tau$, which increases the probability of selecting higher valued actions.

VI. Simulation-based Results

This section describes the results of the simulation-based experiments, which use a non-linear stratified tank model with 50 temperature layers. A detailed description of the stratified tank model can be found in [5]. The specifications of the electric water heater are chosen in correspondence with the electric water heater used during the lab experiment (see Section VII). The simulated electric water heater has a power rating of 2.36kW and has a water buffer of 200 liter. The experiments use realistic hot water profiles with a mean daily consumption of 120 liter [40] and use price information from the Belgian day-ahead [41] and balancing market [42]. The learning agent can measure the temperature of the 50 temperature layers obtained with the simulation model. Two comfort settings are applicable: $T_{\min} = 45°\text{C}$ and $T_{\max} = 65°\text{C}$, where $T_{\min}$ is the minimum temperature at which water is allowed to leave the buffer and $T_{\max}$ is the maximum temperature for the water in the buffer. To guarantee a minimum comfort reserve, the minimum and maximum state of charge in (10) are set to 25% and 100%. If the state of charge drops below 25%, the backup controller in (10) is activated, which forces the buffer to build up a sufficient reserve to safeguard the comfort of the end user. The aim of the first simulation-based experiment is to find a compact state representation using an auto-encoder network and to assess the impact of the state representation on the performance of fitted Q-iteration. The second simulation-based experiment compares the result of fitted Q-iteration with the default thermostat controller.

A. Step 1: feature selection

This experiment compares the performance of fitted Q-iteration combined with different feature representations for different fixed batch sizes. An auto-encoder (AE) network that reduces the original sensory input vector (50 temperature sensors) to 5 dimensions is denoted by AE 5. The simulations are repeated for 100 simulation days. The average cost of energy consumption of these 100 simulations is depicted in Fig. 3. As can be seen in Fig. 3, the performance of fitted Q-iteration combined with a specific state representation depends on the number of tuples in the batch. For example for a batch size of 10 days, AE 3 results in a lower cost than AE 15, while after 75 days, AE 15 will result in a lower cost than AE 3. In addition, as can be seen from Fig. 3, AE 1 resulted in a relatively bad policy, independent of the batch size.

In general, it can be concluded that for a batch of limited size, fitted Q-iteration with a low-dimensional feature vector will outperform fitted Q-iteration using the full state

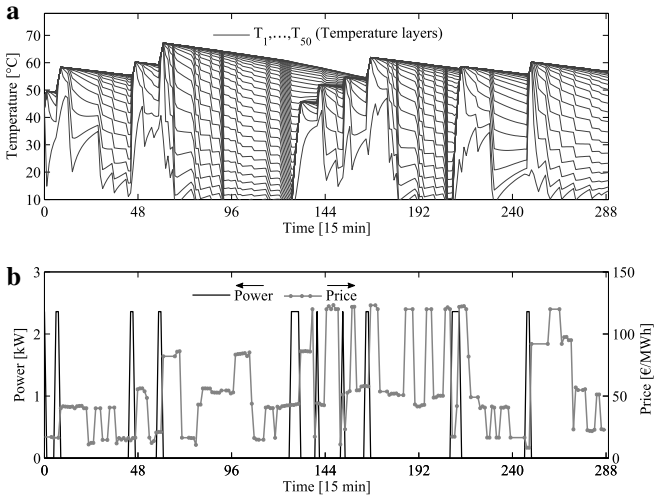

Fig. 5. Simulation-based results of a mature agent using fitted Q-iteration with a non-linear dimensionality reduction. **a**, Temperature profiles of the 50 simulation layers. **b**, Power consumption (black) and imbalance prices (dotted).

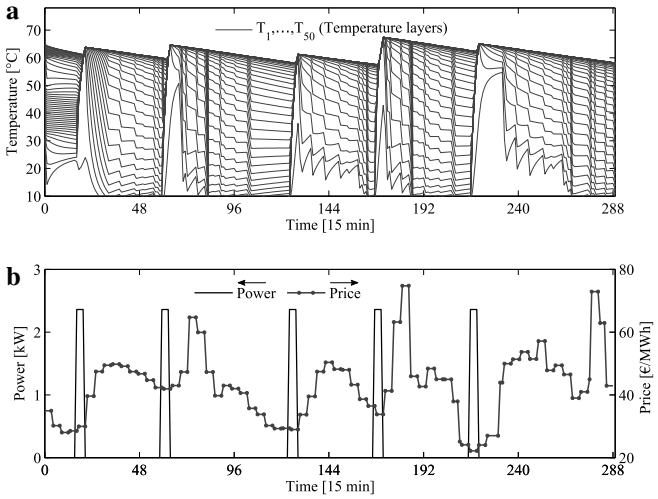

Fig. 6. Simulation-based results of a mature agent using fitted Q-iteration with a non-linear dimensionality reduction. **a**, Temperature profiles of the 50 simulation layers. **b**, Power consumption (black) and day-ahead prices (dotted).

information, i.e. 50 temperature measurements. By learning in a low-dimensional state space, it is possible to learn with a smaller and more efficient representation. As a result, the agent requires less observations to converge to a better control policy than when the full state information is used. In addition, as more observations will result in a more efficient coverage of the state-action space, it can be seen from Fig. 3 that the result of fitted Q-iteration with the full state improves significantly as the batch size increases. In the following subsection, we present the results of AE 5 in more detail. A method for selecting an appropriate feature representation during the learning process will be part of our future work (Section VIII).

B. Step 2: evaluation

Fig. 4 compares the total cost of energy consumption using fitted Q-iteration combined with AE 5 against the default thermostat controller for two relevant price profiles, i.e. day-ahead prices (top plot) and imbalance prices (bottom plot). The default thermostat controller enables the heating element when the state-of-charge drops below its minimum threshold and stays enabled until the state-of-charge reaches 100%. Note that in contrast to the learning agent, the default controller is agnostic about the price profile.

The experiment starts with an empty batch and the tuples of the current day are added to the given batch at the end of each day. At the start of each day, the auto-encoder is trained to find a batch of compact feature vectors, which are then used by fitted Q-iteration to estimate the Q-function for the next day. Online, the learning agent uses a Boltzmann exploration strategy with $\Delta\tau$ set to 10, which results in 10 days of exploration.

The results of the experiment indicate that fitted Q-iteration was able to reduce the total cost of energy consumption by 24% for the day-head prices and by 34% for the imbalance prices compared to the default strategy. Note, imbalance prices are generally more volatile than the day-ahead prices, as they reflect real-time imbalances due to forecasting errors of renewable energy sources, such as wind and solar, which were not foreseen in the day-ahead market. The standard deviation of the day-ahead prices is 12.7, while it is 33.4 for the imbalance prices.

The temperature profiles of the simulation layers and power profiles of a "mature" learning agent (batch size of 100 days) for the day-ahead and imbalance prices are depicted in Fig. 5 and Fig. 6. It can be seen in the bottom plot of both figures that the mature learning agent reduces the cost of energy consumption by consuming energy during low price moments.

A comparison between the presented model-free method and a model-based approach can be found in the Appendix section of this paper.

VII. Lab Results

The aim of our lab experiment is to demonstrate that fitted Q-iteration can be successfully applied to minimize the cost of energy consumption of a real-world electric water heater.

A. Lab setup

The setup used in the lab experiment was part of a pilot project on residential demand response in Belgium [43], where a cluster of 10 electric water heaters was used for direct load control. Fig. 1 shows the electric water heater used during the lab experiment. The electric water heater is a standard unit that was equipped with eight temperature sensors and a controllable power relay. A controllable valve connected to the outlet of the buffer tank is used to simulate the hot water demand of a household with a mean daily flow volume of 120liter [40]. An Arduino prototyping platform with a JSON/RPC 2.0 interface is used to communicate with a

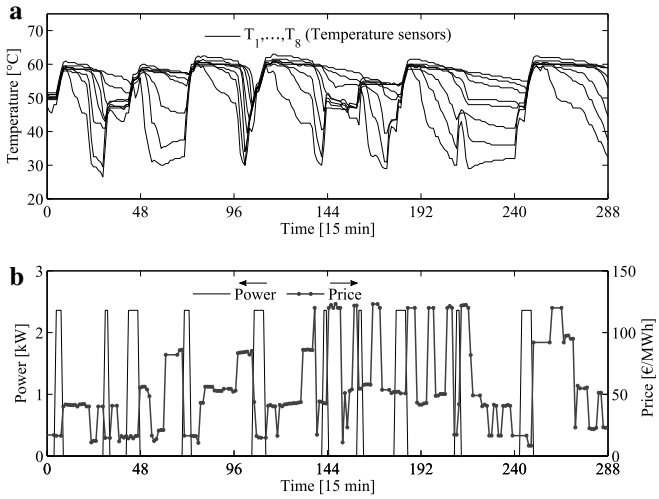

Fig. 7. Lab results of a mature agent using fitted Q-iteration with the full state (eight temperature measurements). **a**, Measurements of the temperature sensors. **b**, Power consumption (black) and imbalance prices (dotted).

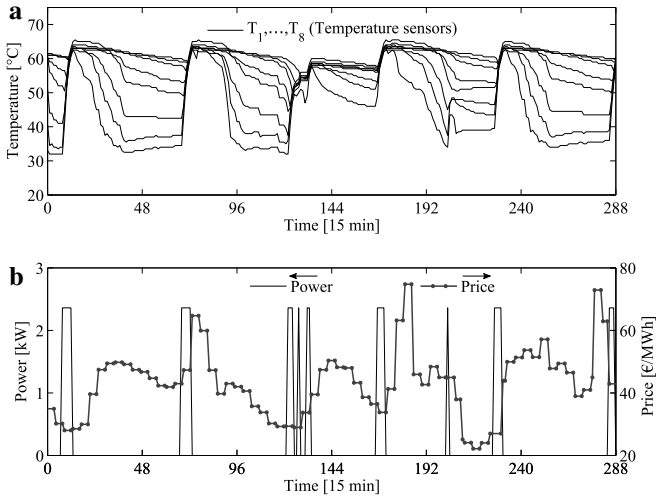

Fig. 8. Lab results of a mature agent using fitted Q-iteration with the full state (eight temperature measurements). **a**, Measurements of the temperature sensors. **b**, Power consumption (black) and day-ahead prices (dotted).

computer in the lab[1], which runs the learning agent that uses fitted Q-iteration. Fitted Q-iteration is implemented in Python and Scikit-learn [44] is used to estimate the Q-function, using an ensemble of extremely randomized trees [23].

Similar as in the previous simulation-based experiments, it is assumed that the learning agent is provided with a deterministic external price profile for the following day. The learning agent uses a Boltzmann exploration strategy with $\Delta\tau$ set to 10, which results in 10 days of exploration. In order to compare the performance of the lab experiment with the performance of the simulation-based experiments, we used both day-ahead prices and imbalance prices.

B. Evaluation

The performance of the learning agent was evaluated using different feature vectors as presented in Section VI. The best performance, however, was obtained by including the eight temperature measurements in the observable state vector.

Using identical price profiles and tap water demands, Fig. 7 and Fig. 8 show the temperature measurements and the power profiles of the mature learning agent using imbalance prices and day-ahead prices. As can be seen, the learning agent was able to successfully minimize the cost of energy consumption by consuming during low price moments.

Fig. 9 depicts the experimental results, spanning 40 days, of fitted Q-iteration and the default thermostat controller. The top plot of this figure indicates the cumulative costs of energy consumption and the bottom plot indicates the daily costs of energy consumption. After 40 days, fitted Q-iteration was able to reduce the cost of energy consumption by 15% compared to the default thermostat controller. Furthermore, by excluding the first ten exploration days, this reduction increases to 28%.

[1]Intel Core i5 - 4GB Memory

VIII. Conclusions and Future Work

This paper has demonstrated how an auto-encoder network can be used in combination with a well-established batch reinforcement learning algorithm, called fitted Q-iteration, to reduce the cost of energy consumption of an electric water heater. The auto-encoder network was used to find a compact representation of the state vector. In a series of simulation-based experiments using an electric water heater with 50 temperature sensors, the proposed method was able to converge to good policies much faster than when using the full state information. Compared to a default thermostat controller, the presented approach has reduced the cost of energy consumption by 24% using day-ahead prices and by 34% using imbalance prices.

In a lab experiment, fitted Q-iteration has been successfully applied to an electric water heater with eight temperature sensors. A reduction of the state vector did not improve the performance of fitted Q-iteration. Compared to the thermostat controller, fitted Q-iteration was able to reduce the total cost of energy consumption by 15% within 40 days of operation.

Based on the results of both experiments the following four conclusions can be drawn: (1) learning in a compact feature space can improve the quality of the control policy when the number of observations is relatively small (25 days); (2) when the number of observations increases it is advisable to switch to higher state-space representation; (3) when only a limited number of temperature sensors is available, i.e. 1-10 sensors, it is recommended to use the full state vector; (4) when applied to a real-world stetting, fitted Q-iteration was able to obtain good control policies within 20 days of operation.

In our future research, we aim at developing a method for selecting an appropriate state representation during the learning process. A promising route is to construct experts, where each expert combines a learning algorithm with a different feature representation. A metric based on the performance of each expert, as presented in [45], could then be used to select

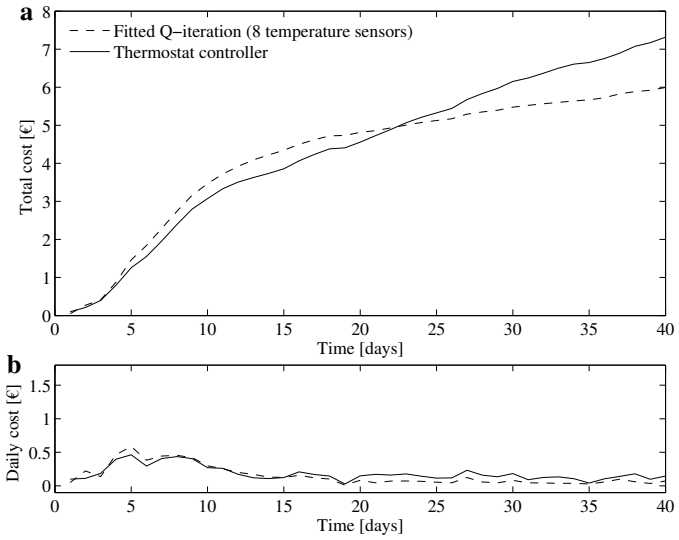

Fig. 9. Lab results of the learning agent using fitted Q-iteration (dashed line) and of the default thermostat (solid line) during 40 days. **a**, Cumulative energy cost. **b**, Daily energy cost.

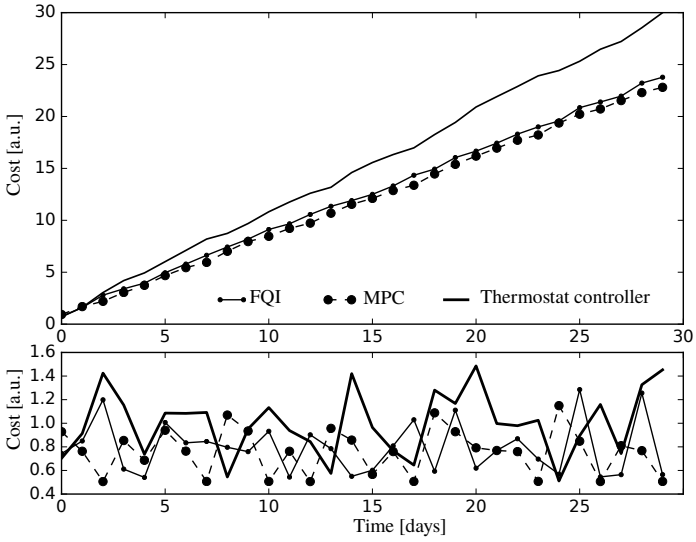

Fig. 10. Simulation-based results. Cumulative (top) and daily (bottom) electricity cost using Model Predictive Control (MPC), Fitted Q-iteration (FQI) and the default thermostat.

the expert with the highest metric as described in [46]. In addition, it would be interesting to investigate the impact of computing the control policy at every time step, using a new price forecast, instead of after each T time steps.

Appendix: Benchmark

In this section, we apply a Model Predictive Control (MPC) approach to the considered control problem. Similar as in [47], [48], the MPC approach uses a linear model with a deterministic forecast of the tap water profile of the next day. The average temperature based on the eight sensors is used to keep the transition function linear. However, the nonlinear mixing problem, as presented in [49], is used to update the states of the simulator. At each control step t, an optimal sequence on the prediction horizon T is found by solving the following MPC problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=t}^{t+T-1} u_k^{\text{ph}} \lambda_k \Delta t \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & \underline{T}_{\text{av}} \leq T_{\text{av},k} \leq \overline{T}_{\text{av}}, \\ & u_t^{\text{ph}} \in \{0, P^{\max}\}, \end{aligned}$$

with x_t the state of the system defined by $[T_{\text{av},k}, T_{\text{amb}}, T_{\text{in},k}]$, u_t the action defined by $[u_k^{\text{ph}}, d_{\text{tap},k}]$ and $\underline{T}_{\text{av}}$ and $\overline{T}_{\text{av}}$ are the minimum and maximum average temperature. To ensure feasibility, the temperature constraints are defined as soft constraints using a slack variable as presented by [50]. The first control action u_k^{ph} of the solution is implemented on the process and the procedure is repeated at the next control step. The overall MPC problem is modeled using CVXPY [51] and is solved using Gurobi [52]. Fig. 10 depicts the daily and cumulative electricity cost of FQI and MPC for a horizon of 30 days. The matrices A an B were estimated using an ordinary least squares method [44]. It can be seen from this figure that both FQI and MPC clearly outperform the thermostat controller and that MPC achieves a slightly better results than FQI (2.8%). It should be noted that this is a reasonable result since the MPC approach used prescient knowledge about the future tap demand, whereas FQI had no information about future tap demands.

Acknowledgment

The authors would like to thank Davy Geysen, Geert Jacobs, Koen Vanthournout, and Jef Verbeeck from Vito for providing us with the lab setup. This work was supported by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) and by Stable MultI-agent LEarnIng for neTworks (SMILE-IT).

References

[1] F. Birol *et al.*, "World Energy Outlook 2013: Renewable Energy Outlook, An annual report released by the International Energy Agency," http://www.worldenergyoutlook.org/media/weowebsite/2013, Paris, France, [Online: accessed July 21, 2015].

[2] M. Albadi and E. El-Saadany, "Demand response in electricity markets: An overview," in *IEEE Proc. Power Engineering Society General Meeting*, June 2007, pp. 1–5.

[3] B. DUPONT, "Dresidential demand response based on dynamic electricity pricing: Theory and practice," *PhD Thesis*, 2015.

[4] B. Hastings, "Ten years of operating experience with a remote controlled water heater load management system at detroit edison," *IEEE Trans. on Power Apparatus and Syst.*, no. 4, pp. 1437–1441, 1980.

[5] K. Vanthournout, R. D'hulst, D. Geysen, and G. Jacobs, "A smart domestic hot water buffer," *IEEE Trans. on Smart Grid*, vol. 3, no. 4, pp. 2121–2127, Dec. 2012.

[6] U.S. Department of Energy, "Energy cost calculator for electric and gas water heaters," http://energy.gov/eere/femp/energy-cost-calculator-electric-and-gas-water-heaters-0#output, [Online: accessed November 10, 2015].

[7] R. Diao, S. Lu, M. Elizondo, E. Mayhorn, Y. Zhang, and N. Samaan, "Electric water heater modeling and control strategies for demand response," in *Proc. 2012 IEEE Power and Energy Society General Meeting,*, 2012, pp. 1–8.

[8] S. Iacovella, K. Lemkens, F. Geth, P. Vingerhoets, G. Deconinck, R. D'Hulst, and K. Vanthournout, "Distributed voltage control mechanism in low-voltage distribution grid field test," in *Proc. 4th IEEE PES Innov. Smart Grid Technol. Conf. (ISGT Europe)*, Oct 2013, pp. 1–5.

[9] S. Koch, J. L. Mathieu, and D. S. Callaway, "Modeling and control of aggregated heterogeneous thermostatically controlled loads for ancillary services," in *Proc. 17th IEEE Power Sys. Comput. Conf. (PSCC)*, Stockholm, Sweden, Aug. 2011, pp. 1–7.

[10] J. Mathieu and D. Callaway, "State estimation and control of heterogeneous thermostatically controlled loads for load following," in *Proc. 45th Int. Conf. on System Science*, Maui, HI, US, Jan. 2012, pp. 2002–2011.

[11] F. Sossan, A. M. Kosek, S. Martinenas, M. Marinelli, and H. Bindner, "Scheduling of domestic water heater power demand for maximizing PV self-consumption using model predictive control," in *Proc. 4th IEEE PES Innov. Smart Grid Technol. Conf. (ISGT Europe)*, Oct 2013, pp. 1–5.

[12] E. F. Camacho and C. Bordons, *Model Predictive Control*, 2nd ed. London, UK: Springer London, 2004.

[13] J. Cigler, D. Gyalistras, J. Širokỳ, V. Tiet, and L. Ferkl, "Beyond theory: the challenge of implementing model predictive control in buildings," in *Proc. 11th REHVA World Congress*, Czech Republic, Prague, 2013.

[14] Y. Ma, "Model predictive control for energy efficient buildings," Ph.D. dissertation, University of California Berkeley, Mechanical Engineering, Berkeley, CA, 2012.

[15] M. Maasoumy, M. Razmara, M. Shahbakhti, and A. Sangiovanni Vincentelli, "Selecting building predictive control based on model uncertainty," in *Proc. American Control Conference (ACC)*, Portland, OR, June 2014, pp. 404–411.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[17] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, "Reinforcement learning versus model predictive control: a comparison on a power system problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 39, no. 2, pp. 517–529, 2009.

[18] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Proc. IEEE 2010 Int. Joint Conf. on Neural Networks (IJCNN)*, Barcelona, Spain, July 2010, pp. 1–8.

[19] E. C. Kara, M. Berges, B. Krogh, and S. Kar, "Using smart devices for system-level management and control in the smart grid: A reinforcement learning framework," in *Proc. 3rd IEEE Int. Conf. on Smart Grid Commun. (SmartGridComm)*, Tainan, Taiwan, Nov. 2012, pp. 85–90.

[20] G. P. Henze and J. Schoenmann, "Evaluation of reinforcement learning control for thermal energy storage systems," *HVAC&R Research*, vol. 9, no. 3, pp. 259–275, 2003.

[21] Z. Wen, D. O'Neill, and H. Maei, "Optimal demand response using device-based reinforcement learning," *IEEE Trans. on Smart Grid*, vol. 6, no. 5, pp. 2312–2324, Sept 2015.

[22] M. González, R. Luis Briones, and G. Andersson, "Optimal bidding of plug-in electric vehicles in a market-based control setup," in *Proc. 18th IEEE Power Sys. Comput. Conf. (PSCC)*, Wroclaw, Poland, 2014, pp. 1–7.

[23] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, pp. 503–556, 2005.

[24] M. Riedmiller, "Neural fitted Q-iteration–first experiences with a data efficient neural reinforcement learning method," in *Proc. 16th European Conference on Machine Learning (ECML)*, vol. 3720. Porto, Portugal: Springer, Oct. 2005, p. 317.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, vol. 27, no. 1, pp. 55–73, 2009.

[27] R. Fonteneau, L. Wehenkel, and D. Ernst, "Variable selection for dynamic treatment regimes: a reinforcement learning approach," in *Proc. European Workshop on Reinforcement Learning (EWRL)*, Villeneuve d'Ascq, France, 2008.

[28] S. Adam, L. Busoniu, and R. Babuška, "Experience replay for real-time reinforcement learning control," *IEEE Trans. on Syst., Man, and Cybern., Part C: Applications and Reviews*, vol. 42, no. 2, pp. 201–212, 2012.

[29] F. Ruelens, B. J. Claessens, S. Vandael, S. Iacovella, P. Vingerhoets, and R. Belmans, "Demand response of a heterogeneous cluster of electric water heaters using batch reinforcement learning," in *Proc. 18th IEEE Power Sys. Comput. Conf. (PSCC)*, Wrocław, Poland, 2014, pp. 1–8.

[30] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuska, and R. Belmans, "Residential demand response of thermostatically controlled loads using batch reinforcement learning," *IEEE Trans. on Smart Grid*, vol. PP, no. 99, pp. 1–11, 2016.

[31] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. of the 28th International Conference on machine learning (ICML-11)*, Bellevue, WA, US, 2011, pp. 465–472.

[32] T. Lampe and M. Riedmiller, "Approximate model-assisted neural fitted Q-iteration," in *Proc. 2014 International Joint Conference on Neural Networks (IJCNN)*, July 2014, pp. 2698–2704.

[33] G. Costanzo, S. Iacovella, F. Ruelens, T. Leurs, and B. Claessens, "Experimental analysis of data-driven control for a building heating system," *Sustainable Energy, Grids and Networks*, vol. 6, pp. 81 – 90, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352467716000138

[34] R. Bellman, *Dynamic Programming*. New York, NY: Dover Publications, 1957.

[35] W. Curran, T. Brys, M. Taylor, and W. Smart, "Using PCA to efficiently represent state spaces," in *The 12th European Workshop on Reinforcement Learning (EWRL 2015)*, Lille, France, 2015.

[36] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Nashua, NH: Athena Scientific, 1996.

[37] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[38] M. Scholz and R. Vigário, "Nonlinear PCA: a new hierarchical approach." in *ESANN*, 2002, pp. 439–444.

[39] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, pp. 237–285, 1996.

[40] U. Jordan and K. Vajen, "Realistic domestic hot-water profiles in different time scales: Report for the international energy agency, solar heating and cooling task (IEA-SHC)," Universität Marburg, Marburg, Germany, Tech. Rep., 2001.

[41] "Belpex - Belgian power exchange," http://www.belpex.be/, [Online: accessed March 21, 2015].

[42] "Elia - Belgian electricity transmission system operator," http://www.belpex.be/, [Online: accessed March 21, 2015].

[43] B. Dupont, P. Vingerhoets, P. Tant, K. Vanthournout, W. Cardinaels, T. De Rybel, E. Peeters, and R. Belmans, "LINEAR breakthrough project: Large-scale implementation of smart grid technologies in distribution grids," in *Proc. 3rd IEEE PES Innov. Smart Grid Technol. Conf. (ISGT Europe)*, Berlin, Germany, Oct. 2012, pp. 1–8.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[45] R. Fonteneau, S. A. Murphy, L. Wehenkel, and D. Ernst, "Batch mode reinforcement learning based on the synthesis of artificial trajectories," *Annals of Operations Research*, vol. 208, no. 1, pp. 383–416, 2013.

[46] M. Devaine, P. Gaillard, Y. Goude, and G. Stoltz, "Forecasting electricity consumption by aggregating specialized experts," *Machine Learning*, vol. 90, no. 2, pp. 231–260, 2013.

[47] E. Vrettos, K. Lai, F. Oldewurtel, and G. Andersson, "Predictive control of buildings for demand response with dynamic day-ahead and real-time prices," in *IEEE, European Control Conference (ECC)*, 2013, pp. 2527–2534.

[48] F. Sossan, A. M. Kosek, S. Martinenas, M. Marinelli, and H. Bindner, "Scheduling of domestic water heater power demand for maximizing pv self-consumption using model predictive control," in *IEEE PES ISGT Europe 2013*, Oct 2013, pp. 1–5.

[49] S. A. Klein, *TRNSYS, a transient system simulation program*. Solar Energy Laboratory, University of Wisconsin, Madison, 1979.

[50] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[51] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[52] Gurobi Optimization, "Gurobi optimizer reference manual," http://www.gurobi. com/, [Online: accessed March 21, 2015].