Towards an intelligent HVAC system automation using Reinforcement Learning

T Schreiber, A Schwartz and D Müller
RWTH Aachen University, E.ON Energy Research Center, Institute for Energy Efficient Buildings and Indoor Climate, Mathieustraße 10, 52072 Aachen, Germany
E-mail: thomas.schreiber@eonerc.rwth-aachen.de

Abstract. HVAC systems are among the biggest energy consumers in buildings and therefore in the focus of optimal control research. In practice, rule-based control and PID controllers are typically used and implemented at the beginning of the building operation. Since this approach neither guarantees optimal or even good control, optimal control algorithms (which can be predictive and adaptive) are in the focus of research. The problem with most of the approaches is that a model of the system is often needed which comes with high engineering efforts. Further, the required computing power can quickly exceed the capacities, even in modern buildings. Therefore, in this paper we investigate the application of a state-of-the-art Reinforcement Learning (RL) algorithm, as a self-calibrating valve controller for two water-air heat exchangers of a real-world air handling unit. We choose a generic problem formulation to pre-train the algorithm with a simulation of an admixing heater and use it to control an injection heater and a throttle cooler. Our results show that after only 70 hours, the control quality significantly increases. Therefore, it seems evident that with pre-trained RL algorithms, a self-improving HVAC automation can be realized with little hardware requirements and without extensive modelling of the system dynamics.

1. Introduction
Heating, ventilation, and air conditioning (HVAC) is one of the biggest energy consumers in buildings [1]. Thus, optimal control strategies for HVAC systems are becoming increasingly important for sustainable building operation [2]. Conventional HVAC control is based on On/Off and PID controllers, which are easy to implemented and have low initial costs [3]. However, in practice, the parameters of PID controllers are not set optimally, which significantly reduces their control performance and results in inefficient operation [4]. Model Predictive Control (MPC), a well researched optimal control technique, promises a nearly optimal operation but demands computing power and an accurate system model [5]. Especially the creation of the model is uneconomic in many cases and must be repeated after every change in the system [6]. Due to its model-free nature and recent developments in Deep Learning, optimal control with (model-free) Reinforcement Learning (RL), in which an agent learns a control policy from interactions with an environment, is considered a promising alternative [7]. RL benefits from low computational costs (after training) and inherent adaptiveness [8]. The interested reader is referred to the following review papers for a comprehensive literature review: opportunities and challenges of RL in building energy systems control [7] and RL for autonomous building energy management systems [9]. What most existing studies have in common is that they apply RL exclusively to
simulated environments. The application to real buildings is rare due to safety concerns. In addition, RL is rarely used for direct control of actuators, but rather for supervisory control. To address these open issues, we investigate simulation-based Deep Q-Learning pre-training for direct valve control of two water-to-air heat exchangers of a real air handling unit (AHU).

2. RL and Deep Q-Networks for AHU valve control
The task is to control the outgoing air temperature of one cooler and one reheater, water-air heat exchanger via their representative valves. Solving this problem via RL agents (one for each heat exchanger) requires the definition of a Markov Decision Process (MDP), which is a formalized framework for sequential decision problems [10]. The MDP should be as generic as possible to allow transferability from the simulation environment to different real-world heat exchangers. Transferability is important because the simulation model generally reflects the reality only to a limited extend and further because of different hydraulic characteristics of the real hydraulic systems. The hydraulic system determines the effect of a change of the valve position on the outgoing air temperature [11]. The underlying hydraulic system of the cooler is a throttle circuit [11] and the one of the reheater is designed as an injection circuit [11]. The RL agents training as well as the final control script are implemented on a typical personal computer, using Python and Tensorflow [12]. The communication between the computer and the AHU automation is realized via a REST API, a database, and a MQTT broker.

2.1. RL and Deep Q-Networks
As a sub-family of machine-learning the focus of RL is to solve optimal control problems based on delayed rewards from an environment [13, 14]. RL requires the formulation of a MDP [10]. A MDP is characterized by: a set of states $S$, a set of actions $A$, a state transition probability matrix $P = f(s, a)$, a reward function $R = f(s, a)$, and a discount factor $\gamma \in [0, 1)$. At a given time-step $t$, the agent takes action $a_t \in A$ based on the state $s_t \in S$ and the MDP transitions to the successor state $s_{t+1} \in S$ and the agent receives a reward $r_{t+1}$. The agent seeks to maximize the total return from the environment over time. A well researched family of algorithms, encoding the optimal policy in form of Q-values (state-action-values) is called Q-Learning. In the state-of-the-art algorithms, neural networks are used as function approximators, for the Q-function [10]. For our research we are using a Deep Q-Networks implementation, including two training stability improvements, namely a replay buffer and a target-network [15].

2.2. A generic problem formulation for AHU valve control
Figure 1 shows the available data points of the AHU. The incoming air with temperature $T_3$ streams trough the cooler before it enters the reheater. Both heat exchangers are connected to different supply water circuits. The task of the two agents is to provide a stable supply temperature $T_4$ set by the occupants of the supplied workshops. The temperatures with the highest influence are the temperatures of the incoming air ($T_3$ and $T_4$), the temperature of the cold supply water (cooler ($T_{w1}$)) and the temperature of the warm supply water (reheater $T_{w4}$)). We choose the reward function to be solely dependent on the deviation of the outgoing temperature from the setpoint: $R = f(|T^{set\text{ }}_{t+1} - T^{out\text{ }}_{t+1}|)$. Figure 2 shows the reward function. Maintaining the temperature in a range of 1 °C around the setpoint leads to the highest rewards. All sections of the reward function have a slope higher than 0 to steer the agent in the direction of the smallest deviations. The decision based on the Q-values also takes into account the possible future rewards, so the agent should also learn to avoid overshooting. We allow the agent to make an open, close, or stay decision: $a_t = [+X, \pm 0, -X]$. This has two advantages over the specification of fixed valve positions: Firstly, the online interaction with the real system is saver since the agent is not able to select extreme valve positions. Secondly, the MDP is of a much a lower dimension and therefore the training is less data demanding. On the other hand requires
the selection of X a trade-off between strong intervention in extreme situations and a granular control.

Figure 1. The hydraulic scheme of the cooler and the reheater.

Figure 2. The reward function.

The definition of the state-space is a key challenge, since it has to include all information necessary to solve the control problem, should be as compact as possible (to reduce training times), and should be generic enough to transfer the agent from the simulation to the different heat exchangers. Including all available datapoints together with historical values (to learn the dynamic of the thermal inertia), would result in a very large and over specified state-space. Taking these requirements into account, we choose the following state-space: \( s_t = [T_{set}^t - T_{out}^t, T_{set}^{t-1} - T_{out}^{t-1}, T_{set}^{t-2} - T_{out}^{t-2}, T_{set}^{t-3} - T_{out}^{t-3}, T_{set}^{t-4} - T_{out}^{t-4}, a_{t-1}, a_{t-2}, a_{t-3}, a_{t-4}, a_{t-5}] \). This state-vector incorporates the deviation from the setpoint in the current and the four steps before as well as the last actions. That makes the state-vector generic because the deviation is not dependent on the temperature levels of a heat exchanger. The past selected actions provide the agent with information about the effect of its actions on the outgoing air temperature under unknown boundary conditions in a generic way. Figure 3 visualizes the the state-vector. The dotted line plots a setpoint of 21 °C and the black line plots an exemplary course of a outgoing air temperature. The red and blue lines mark the deviations from the setpoint.

2.3. Deep Q-Network training

We use a simple Modelica simulation model of a water-air heat exchanger with an admixing water circuit, for pre-training. The used model is freely available in the AixLib Github repository [16] and provides the physical relationships sufficiently for our purpose. The simulation model is exported as a FMU [17] and is connected to the agent following the standardized Gym interfaces [18]. The incoming air temperature is designed as a sinus function, which is initialized with random parameters, each training episode. The incoming air temperature varies between 1 and 10 °C below the setpoint with different frequencies. The setpoint is constantly set to 20 °C. One episode consists of 50 simulated minutes, with one interaction every other minute. We set X in the action-space to 10. For this setup we use hyperopt [19] to identify the optimal agents hyperparameter via a Bayesian parameter search. The resulting hyperparameters are: batch size (64), replay memory size (10,000: approximately 167 hours of interaction), minimum replay memory size (480: training starts after 8 hours), target network update frequency (30), number of neurons (30), and discount factor \( \gamma \) (0.36). We handle the exploration versus exploitation trade-off via \( \epsilon \)-greedy and decrease the probability to select a random action after every interaction by 0.001 %. Figure 4 shows a positive trend in the rewards in the course of 4000 training episodes.
Considering the limiting influence of the randomly initialized input air temperature, the agent significantly improved in exploiting the rewards from the training environment (max reward is 150). We stop the training at this point to avoid over-adaption to the simulation and keep a sufficient degree of adaptability in the online interaction phase.

Figure 3. Visualisation of the state-space, consisting of the deviations from the setpoint.

3. Online valve control

Figure 5 shows the development of the average reward of the cooler agent and the associated exploration (one episode consists of 30 interactions). We observe a positive trend in the average rewards. Due to the minimum replay memory size, the agent controls the valve based on the initial policy in the first eight hours.

Figure 5. Rewards and exploration during online interaction.

During this period, the standard deviation (shaded area) is between 1 and 2.7. After ten hours of interaction, the rewards drop, increase again in the following 30 hours, and after 60 hours, the rewards rise to 5 with a significant lower standard deviation. A high standard deviation indicates a high fluctuation of the outgoing air temperature within one episode, a decreasing standard deviation can therefore be interpreted as a reduced oscillation around the setpoint. The real heat exchangers, react more sensitively to valve opening changes than the simulated
heat exchanger. Therefore, we set \( A_{\text{cooler}} = [+3, \pm 0, -3] \) and \( A_{\text{reheater}} = [+8, \pm 0, -8] \). Further, using the agent for the cooler requires to negate the deviations from the setpoint in the cooler’s agent state-vector. Both agents show the expected behaviour: If the temperature is too high, the agent increases the valve opening, whereas the valve opening is decreased when the temperature is too low (inverted for the heater). On the other hand, start both heat exchangers with strong oscillations around the setpoint.

![Control behavior on the last day of online training: cooler](image)

**Figure 7.** Undisturbed policy after 70 hours of online training.

Figure 6 shows the learned Q-values with respect to the current deviation from the setpoint. The current deviation from the setpoint is displayed on the x-axis while the y-axis shows the available actions. The surface illustrates the Q-value of each state-action pair. The red line shows the greedy action. In spite of the initial policy, where the agent does not change the valve position if the deviations are smaller than 1 °C, the control behavior improved. There is no deviation from the setpoint where the agents would not change the valve position. Additionally, shows figures 7 the plot of the temperatures and valve positions on the last day of online training. The outgoing air temperature does no longer oscillate around the setpoint. At the beginning (07:00 - 09:00), the temperature is slightly below the setpoint. However, the agent correctly counteracts firstly by gradually reducing the valve position (09:00 - 10:00) and then gradually increasing the valve position again, as the water temperature rises from 14 °C to 15.5 °C (10:00 - 11:00). The agent now selects to alternately increase and decrease the valve position. The results for the reheater cannot yet be conclusively assessed. Especially because our experiments were conducted in summer. Because of the high outside temperatures, the cooling agent had an easier task, as the boundary conditions are relatively static. The reheater, on the other hand, was trained to deal with the highly oscillating output temperatures of the cooler.

### 3.1. Discussion

The presented results show that with the introduced MDP formulation, a DQN agent can significantly improve its policy within 70 hours. In contrast to the beginning, there is no oscillation around the setpoint on the last day of online training. We expect that there is still a high potential for improvements. During the pre-training the agent is trained with 100,000 interactions, while 70 hours of online training are equivalent to 4,200 interactions. The fact that the cooler’s agent has improved visibly in such a short time indicates the adventurousness of the compact formulation of state- and action-space. Interestingly without the own past actions, the agent is not able to control the valve correctly. We conclude that this is important for the agent.
to distinguish between the own influence and the influence of changing boundary conditions. Because of the different thermal inertia of heat exchangers, in future work it may be necessary to either increase the discount factor $\gamma$ or include more historical values. In order to realise a more precise control, it could also make sense to extend the action-space: $A = [+Y, +X, \pm 0, -X, -Y]$. 

4. Conclusion and outlook
Our work contributes to the development of self-calibrating building automation systems. We demonstrate a generic MDP formulation where the accuracy of the simulation model does not play a crucial role in the applicability of pre-trained agents to different hydraulic systems. Our results show that the time required for online training can be significantly reduced by pre-training. In the case of the cooler, the simulated heat exchanger has both an opposite behavior and a different hydraulic system. After oscillating around the setpoint at the beginning of the online training, the outgoing air temperature was maintained at the setpoint at the end, without oscillating. Online training of the reheater’s agent was severely limited by unfavorable boundary conditions (very high outdoor temperatures) and therefore cannot be evaluated conclusively yet. In future work, we will further investigate the generic applicability of the MDP by applying the agents to more heat exchangers. The application of Long Short-Term Memory (LSTM) function approximators can also be investigated to move historical data from the state-space to the internal memory of the neural network. Additionally, the pre-training process could be further refined by running specific scenarios such as setpoint jumps.

5. Acknowledgments
We gratefully acknowledge the financial support provided by the BMWi (Federal Ministry for Economic Affairs and Energy), promotional reference 03SBE0006A.

References
[1] Yuan X, Pan Y, Yang J, Wang W and Huang Z 2019 Building Simulation
[2] Han M, Zhang X, Xu L, May R, Pan S and Wu J 2018 A review of reinforcement learning methodologies on control systems for building energy
[3] Mirinejad H, Sadati H, Maryam G and Hamid T 2008 Journal of Computer Science 4
[4] Abel D 2018 Umdruck zur Vorlesung Regelungstechnik und Ergänzungen (Höhere Regelungstechnik) (Verlaghaus Mainz) ISBN 3-8107-0067-3
[5] Afram A and Janabi-Sharifi F 2014 Building and Environment 72 343–355
[6] Schreiber T, Netsch C, Eschweiler S, Wang T, Storek T, Baranski M and Müller D 2021 Energy 230 120894
[7] Wang Z and Hong T 2020 Applied Energy 269 115036 ISSN 0306-2619 URL http://www.sciencedirect.com/science/article/pii/S0306261920305481
[8] Gorges D 2017 IFAC-PapersOnLine 50 4920–4928
[9] Mason K and Grijalva S 2019 Computers & Electrical Engineering 78 300–312
[10] Sutton R S and Barto A G 2018 Reinforcement learning: An introduction (MIT press)
[11] Recknagel 2017 Taschenbuch für Heizung + Klimatechnik 2017/2018 (Recknagel - Sprenger - Albers)
[12] Agarwal A, Barham P, Brevdo E, Chen Z and et al 2015 Tensorflow:large-scale machine learning on heterogeneous distributed systems
[13] Dayan P and Niv Y 2008 Current opinion in neurobiology 18 185–196
[14] Mohri M, Rostamizadeh A and Talwalkar A 2018 Foundations of machine learning (MIT press)
[15] Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare M, Graves A, Riedmiller M, Fidjeland A and et al 2015 Nature 518 529–33
[16] Müller D, Lausen M, Constantin A and Remmen P 2016 Aixlib - an open-source modelica library within the iea-ebc annex 60 framework (BauSim) pp 3–9
[17] Blockwitz T, Otter M and et al Functional mockup interface 2.0: The standard for tool independent exchange of simulation models Proceedings of the 9th International MODELICA Conference 2012 pp 173–184
[18] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J and Zaremba W Openai gym URL https://arxiv.org/pdf/1606.01540
[19] Bergstra, J and Yamins, D, Cox, D 2013 Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures