Autonomous Building Control Using Offline Reinforcement Learning

Jorren Schepers, Reinout Eyckerman, Furkan Elmaz, Wim Casteels, Steven Latré and Peter Hellinckx

Abstract Artificial Intelligence (AI) powered building control allows deriving policies that are more flexible and energy efficient than standard control. However, there are challenges: environment interaction is used to train Reinforcement Learning (RL) agents but for building control it is often not possible to use a physical environment, and creating high fidelity simulators is a difficult task. With offline RL an agent can be trained without environment interaction, it is a data-driven approach to RL. In this paper, Conservative Q-Learning (CQL), an offline RL algorithm, is used to control the temperature setpoint in a room of a university campus building. The agent is trained using only the historical data available for this room. The results show that there is potential for offline RL in the field of building control, but also that there is room for improvement and need for further research in this area.

1 Introduction

Building control is very complex due to the large number of components. The most basic control is rule based control, where a fixed schedule is always followed and no variability, such as open windows or weather conditions, is considered. Over the last years interest has risen in Reinforcement Learning (RL) powered building control. RL algorithms use environment interaction to learn, which is not straightforward for building control. Letting a learning agent control an occupied building can lead to discomfort and potentially unsafe situations for the occupants. During training the

Jorren Schepers · Reinout Eyckerman · Furkan Elmaz · Wim Casteels · Steven Latré · Peter Hellinckx
University of Antwerp - imec
IDLab - Faculty of Applied Engineering
Sint-Pietersvliet 7, 2000 Antwerp, Belgium,
e-mail: jorren.schepers@student.uantwerpen.be
reinout.eyckerman@uanterpen.be

agent will take undesirable actions, such as heating the room to a high temperature or letting the CO2 levels rise to explore the possibilities of the environment. Moreover, it can take a long time for an agent to converge in a real world environment, between 20 up to 50 days [1]. Due to this, simulators are most often used, which represent the environment in which the agent will be deployed with a model. This model has to be as accurate as possible, otherwise the agent might work well in the simulator but perform poorly in the real world due to some flaw it could exploit in the simulator.

Simulator design is a difficult task and requires a considerable amount of domain knowledge. Buildings are complex nonlinear systems, influenced by a wide range of parameters such as occupancy, weather conditions and deterioration. Additionally, a new model has to be created for every building due to building specific characteristics. Furthermore, some aspects can have other levels of importance in different cases, so even for every case the model might have to be recreated.

Recently, advancements have been made in the field of offline RL [2–4]. Here an agent is trained from a dataset, without need for environment interaction, eliminating the need for highly accurate and case specific simulators. This dataset can be collected using different policies from which the agent is often able to derive a better policy [2]. With the rise of the Internet of Things (IoT), buildings are often equipped with a wide range of sensors that can provide sufficient data for this approach.

In this paper we use offline RL to derive a control policy for a Heating Ventilation and Air Conditioning (HVAC) system in a campus building. This building has over 3000 sensors measuring temperature, window states, and occupation, among others. With just this information, we train an agent to control the room temperature with a better policy than the currently implemented rule based control policy.

First, an overview of related work is provided in section 2, continued with offline RL and its challenges in section 3. In section 4 our method is explained, followed by the results and discussion in section 5. After that we conclude in section 6 and suggest future work along with the limitations of this paper in section 7.

2 Related work

At the time of writing the authors have no knowledge of any other studies using offline RL for building control. There are, however, several studies that use online RL. Wang and Hong [5] provide a literature review on the use of RL for building control. The majority (76.7%) of used algorithms in this review use value based techniques. Furthermore, they state that, to have a fully observable environment, historical state values should be included in the observation space, although only one out of six reviewed studies considers this. In addition, some studies (16%) use predicted states such as weather forecasts, which can improve performance by 27% [6].

Most studies use discrete actions as value based algorithms are frequently used. Some examples: Brandi et al. [1] use discrete actions by defining 5 possible setpoints to control supply water temperature for heating terminals, and Zhang et al. [7] use a larger action space that ranges from 20°C up to 65°C in increments of 5°C and an

extra action to turn off the heating. Increasing the size of the action space linearly increases the execution complexity [8], so when working with discrete actions the value of adding more actions should be weighed to the increased complexity.

The reward function is based on which objectives should be optimized. Han et al. [9] focus on studies where occupant comfort is the primary objective, with a few studies only optimizing for comfort but the majority includes energy. Considering both objectives is important however, as energy and comfort are conflicting.

MATLAB and EnergyPlus white-box simulators are very popular for building simulation [5]. They are based on thermodynamic equations, opposed to black-box simulators, where a neural network is trained with recorded data, or gray-box simulators, where a mix of both is used. Afroz et al [10] provide a more detailed overview of simulator modeling techniques used in building HVAC control systems.

Very few studies evaluate the trained agent in a physical environment, less than 12% of the studies reviewed by Wang and Hong [5] and 18% of the studies reviewed by Han et al. [9]. Additionally, multiple papers suggest that there is need for a universal benchmark to test and compare RL algorithms for building control. Jia et al. [11] propose a Building Virtual Testbed that integrates EnergyPlus simulations with Python to enable easier evaluation of RL algorithms.

3 Offline reinforcement learning

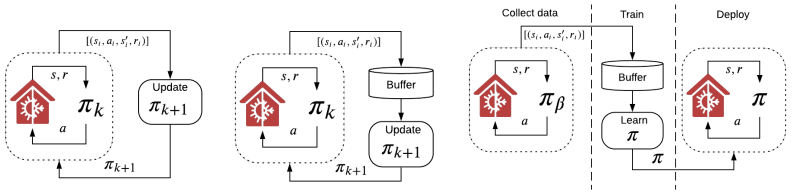
Traditional, or online, RL can generally be grouped in two categories, on-policy and off-policy. For on-policy RL the agent improves its policy by interacting with the environment using its current policy. The action, state, reward and new state are saved as transitions which are used to update to a new policy, as shown in Fig. 1a.

With off-policy RL every learned policy adds new transition tuples to a buffer, which is then used to learn a new policy as shown in Fig. 1b. This means that transitions are not only gathered by the current policy, but also by all previous policies.

Offline RL, a data-driven approach for RL, trains an agent without environment interaction [2]. A buffer is also used, but it is initialized with previously collected transitions and during training no new transitions are appended, as shown in Fig. 1c.

Off-policy algorithms can be used in offline settings by preinitializing the buffer and not using environment interaction, but suffers from distributional shift [2,12]. A new policy is learned from a dataset which has a certain state-action distribution, but when deployed it will induce a different state-action distribution which enters states that are not not in the dataset. An agent learns by approximating a high dimensional function with the training data, but this function extrapolates beyond the domain of the data. The policy might overestimate the value of out-of-distribution states, taking actions that yield a low reward. Distributional shift also occurs for off-policy RL, but with environment interaction, the agent can learn and correct those overestimations.

Levine et al. [2] provide an overview of the different solutions in previous literature. Three approaches are listed to deal with this problem:



(a) on-policy RL (b) off-policy RL (c) offline RL

Fig. 1 On-policy RL **(a)**: the new policy is updated with transitions from the current policy. Off-policy RL **(b)**: the new policy is updated using a buffer with transitions from all previous policies. Offline RL **(c)**: the new policy is learned from a preinitialized buffer

Explicit Policy Constraint Methods limit the difference between the learned policy π and the behavioral policy π_β. This approach is pessimistic but it does not include great risk. A disadvantage is that π_β must be known or estimated. Using an estimation for π_β can introduce an extra error.

Q-values Regularization changes the objective to learn a conservative value function. A penalty is added for state-action pairs that were not in the training distribution. This prevents overestimation of Q-values without constraining π to stay close to π_β, so there is no need to know π_β.

Implicit Policy Constraint Methods discourage selecting out-of-distribution actions by creating a weighted behavioral cloning objective that prefers good actions over bad actions. It uses a filter function to favor high-reward transitions that were sampled from the data. In this case π_β does not need to be known either.

Monier et al [12] show that the most promising algorithms are Conservative Q-Learning (CQL) [3] and Critic Regularized Regression (CRR) [4], based on the second and third approach, respectively. Both significantly outperform off-policy algorithms and are capable of deriving a better policy than present in the data.

When comparing CQL and CRR, CRR seems more robust to hyperparameter tuning and performs better with sparse rewards, whereas CQL is very sensitive to the value of α, but shows a better accuracy when properly tuned [12]. CRR is an actor-critic method whereas CQL supports both actor-critic and value-based settings. Due to a continuous action space we will focus on actor-critic CQL.

Regarding building control, CQL seems the best fit, since it is more conservative, which is safer and, when properly tuned, yields a better accuracy than CRR [12].

4 Method

In this section we will describe the problem setup and our goal by giving an overview on the data, how the reward is designed and how validation is handled.

4.1 Data

The dataset used in training contains data ranging from December 1st to December 24th 2020. This data is gathered using the sensors present in building Z at campus Groenenborger of the University of Antwerp. Roughly every minute all of the sensors log their value. From this data, features were selected to present to the agent.

These features include the room and outside temperature, the difference between the room and target temperature, occupation, the hour and the minutes. The target temperature is the setpoint provided to the Proportional-Integral-Derivative (PID) control unit, which is the action taken by the behavioral policy. This policy is based on a calendar system that defines a setpoint based on the current date and time.

As discussed in section 2, temperature history is important. We tested two approaches, the first one adds a certain amount of previous values as features. This creates a very large observation space when taking into account more history, which will make learning complex. The second one extracts the minimum, maximum and slope from all of the history values and uses those as features. This creates the possibility to include a longer history without creating a very large observation space, however a lot of information is lost when compared to the first approach

The state-action coverage of the dataset is essential. Since this building has a rule-based controller implemented during the collection of the data, the visitation in this dataset is considered expert, with little undesirable actions nor states. However, since this data was recorded during the Covid19 pandemic, some unusual states are present. Sometimes a temperature of 10°C is requested while there is somebody in the room, which in normal circumstances is not considered an expert action. We believe this is due to government restrictions where windows had to be opened for ventilation and it is probable that heating was turned off at this moment, since a 10°C setpoint does not appear in the calendar system. The fact that this data is recorded during a pandemic, in which schools and universities were partially closed, is also visible in the low occupation rate. For only 13.2% of the time the room is occupied.

4.2 Reward

The reward is based on four features: the current room temperature, the target temperature from the calendar system, the occupation and the action taken by the agent, which is the new setpoint for the PID control unit. In eq. 1 the reward is described in a mathematical form, O_c refers to the occupation, which is either 1 or 0, T_r refers to the room temperature, T_c is the temperature from the calendar and T_a is the action taken by the agent. All temperatures are in degrees Celcius.

$$-O_c * |T_r - T_c| - (1 - O_c) * |T_r - T_a| \tag{1}$$

This equation punishes the agent for the difference between the room temperature and the scheduled temperature if the room is occupied. If the occupation is 0, the

reward is the difference between the room temperature and the action the agent takes, to stimulate lower energy usage.

4.3 Algorithm

As discussed in section 3, CQL was used. The implementation was provided by d3rlpy [13], an offline RL library. A standard Soft Actor Critic (SAC) implementation is expanded with a conservative Q-function as suggested by Kumar et al. [3].

4.4 Validation

Validation is challenging due to lack of simulation and deployment possibilities. A black-box model from previous research [14] was used, which is trained to predict the temperature for the same room from which we have a dataset. This model is wrapped in an OpenAI Gym environment to test deploying the trained agent.

The agent is deployed in the environment for 3000 timesteps, which corresponds to roughly 60 hours. This allows observing the actions of the agent at both the start and the end of several days, and there is sufficient time for the agent to have a meaningful influence on the temperature. The behavior for this deployment is recorded and used to calculate different error metrics to compare the agent to the original PID control unit. The used error metrics are: Integral Absolute Error (IAE), Integral Square Error (ISE) and Integral Time-Absolute Error (ITAE) and they are calculated between the room temperature in the simulation and the target temperature (for the agent's performance), and the room temperature from the dataset and the target temperature (for the PID control unit's performance).

When there is no occupancy the controller is encouraged to keep its temperature as a naive approach to energy efficiency. Due to this, only values during daytime, between 7:00 and 17:00, are used to calculate the error metrics. This puts the focus on training an agent, only using data, to outperform the current policy by day.

During training several metrics give an indication if the agent is learning. This includes the losses for both actor and critic, the standard deviation and average of the values, and the reward from a short environment deployment (60 steps).

A large standard deviation can indicate that the agent is overfitting. A large average of the values can indicate overestimation due to out-of-distribution actions. High losses for actor or critic show that the learned policy or value function is not approximating with high accuracy. The reward can be interpreted as an indication of learning correct behavior, but it is not enough to conclude learning is succeeding. The reward can be high for episodes where the agent's actions do not fluctuate much and there is little occupation during that time, since then the agent will not be penalized often for not being close to the target temperature.

5 Results and discussion

In the first experiments the observation space presented temperature history as a feature for each value for the length of the history. This increases the observation space to almost four times its size and, as can be observed in the metrics in Fig. 2, creates a too complex problem for the agent to solve. The losses are very high, the values very low and the standard deviation is also very high compared to a training using just the extracted features as history. Because of this, the history was presented using the second approach discussed in section 4.1 for 30 steps.

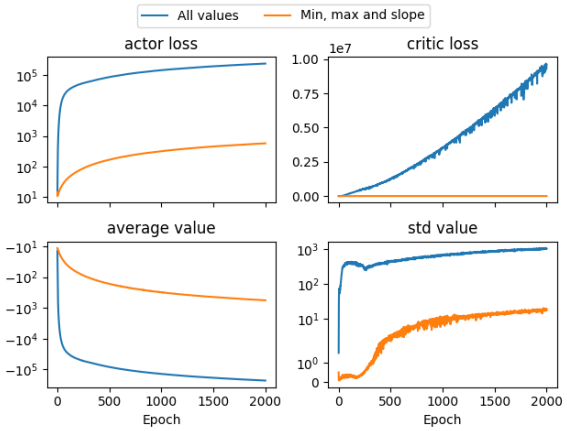


Fig. 2 Training statistics comparing history with every value to extracting features (min, max and slope)

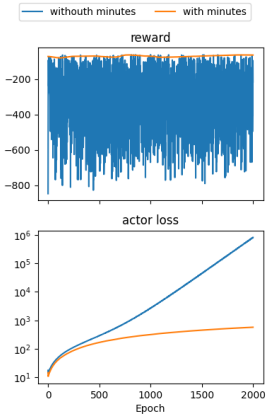


Fig. 3 Training statistics comparing minutes in the observations space to an observation space without minutes

Hyperparameter tuning uncovered a pattern in the actions in several experiments that seemed to strongly correlate with the minutes of the hour. As a test, some experiments were set up with the same hyperparameters and observation space, excluding the minutes feature. In the metrics shown in Fig. 3 we can see that this renders training unstable, the actor loss goes up to 10^6 and the reward strongly fluctuates every epoch and is lower compared to other experiments.

Using a parameter sweep we determined the actor and critic learning rate, gamma, and alpha. Other parameters were fixed, and can be observed in Table 1.

Since we still had relatively high losses, the learning rates for both actor and critic were lowered. A grid search was conducted with 1e-5, 3e-5 and 5e-5, training would stabilize most desirably for a learning rate of 1e-5 for both actor and critic.

For gamma the values 0.9, 0.99 and 0.999 were tested, 0.999 yields the best results when combined with the other parameters. This makes sense as temperature varies slowly and a high gamma assigns more importance to future rewards. This could be observed when comparing evaluations of agents that were trained with a

Table 1 Fixed hyperparameters

Parameter	Value
Initial SAC temperature	1
Temperature learning rate	1e-5
Tau	5e-3
Batch size	512

different gamma value. In Fig. 4 agent 1 and 2 are trained with gamma 0.999 and agent 3 is trained with gamma 0.99. All agents requested higher temperatures in the morning, but agent 1 and 2 chose higher setpoints than agent 3 and kept higher values until the late afternoon. This resulted in a faster rise of the temperature in the morning and a higher average temperature during the day.

The CQL specific alpha value determines how conservative the Q-function should be. In our experiments, several values are tested from 0.001 up to 10. For lower values of alpha, 0.001 up to 5, the agent often just picked one of the outer boundaries of the action space and always executed an action in a range of 5 degrees from it. With alpha equal to 10, the agent did learn to follow the schedule very roughly, as it did increase the temperature in the morning and lowered in the early evening, but the agent would almost never pick values higher than 22 degrees or lower than 19, which means the agent was too conservative. The best results were obtained with an alpha of 9.

In Fig. 4 the simulation temperature for the two best agents is shown, along with the original temperature in the bottom right plot and a simulation of an agent trained with gamma 0.99 in the bottom left plot. When observing the actions of the agent, the requested temperature, we can deduce that the agent learned from this dataset that heating should start early and that it can stop heating before the end of the day as the room will not cool down very fast due to the thermal mass of the building.

In Table 2 the error metrics are shown, all error metrics are calculated by integrating over time. The ISE uses the square of the error, tolerating smaller errors more than larger errors. When comparing the agents performance to the original data with this metric, we can conclude that the errors of the agents are generally smaller. The ITAE multiplies the absolute error with the time before integrating, this penalizes errors that persist. This can also be observed in Fig. 4, as time passes, the temperature is closer to the target temperature and therefore the error is smaller at later timesteps. The IAE does not add extra importance to any kind of error. The lowest error scores are marked in bold and, with the comparison above, we can conclude that the agents outperform the PID controller during the day.

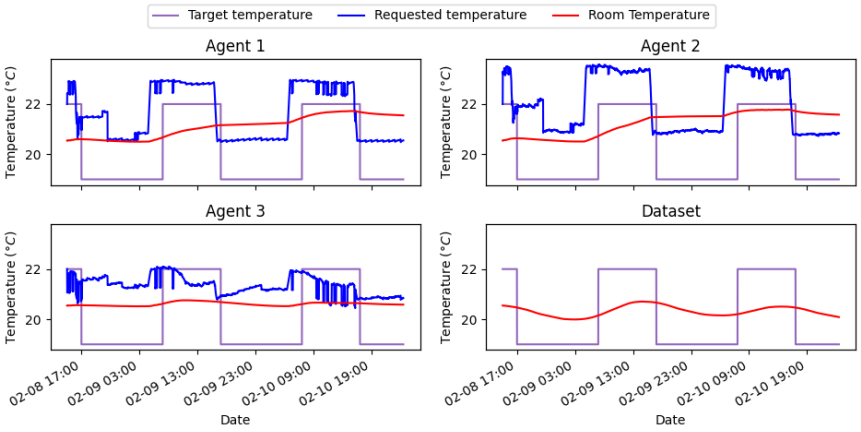


Fig. 4 Simulation results: agent 1 and 2 are trained with gamma 0.999, agent 3 with gamma 0.99

Table 2 Error metrics during training

Error metric	Agent 1	Agent 2	Agent 3	Data
ISE	880.97	**676.08**	2053.19	2726.52
ITAE	364638.38	**276110.46**	896962.30	1028230.78
IAE	894.03	**742.07**	1543.17	1771.49

6 Conclusion

In this paper we introduced offline RL to building control, first the challenges of offline RL are discussed, after that we elaborated on the available data and explained the set up for training. This was followed by the results of a case study in which a novel offline RL algorithm is used for continuous control of the setpoint temperature for an HVAC system. In these results we can see that, using just a historical dataset of a building, the agent could grasp the concept of thermal mass, as it starts and stops heating before the target temperature changes. Additionally, the agent was able to outperform a scheduled PID control unit during daytime by requesting higher temperatures than those that should be reached according to the schedule, so that the temperature would rise faster.

7 Limitations and future work

While in this work only one objective is optimized, most RL studies in the HVAC domain also try to optimize energy. However, energy data was not readily accessible due to the limitations of the building sensor measurements. This only allows performance optimization during the day. During the night, the temperature is kept con-

stant as shown in Fig. 4, which is likely to use more energy. In future work, energy consumption should be used as part of the reward. This does require an improved dataset. Other options include training a model to predict energy consumption based on the state or calculating a rough estimate when there is enough system knowledge.

Multi-room or even building wide control would improve the impact of the offline RL approach. We propose a multi agent approach since the large observation space will create a problem that might be too complex to solve in a single actor-critic setting. When all the rooms of the building have very similar characteristics, there is also an option to use a centralized critic with multiple cooperating actors.

References

1. S. Brandi, M. Piscitelli, M. Martellacci, and A. Capozzoli, "Deep reinforcement learning to optimise indoor temperature control and heating energy consumption in buildings," *Energy and Buildings*, vol. 224, 2020.
2. S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," *arXiv*, may 2020.
3. A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," *arXiv*, jun 2020.
4. Z. Wang, A. Novikov, K. Zołna, J. T. Springenberg, N. Siegel, S. Reed, J. Merel, B. Shahriari, C. Gulcehre, N. Heess, and N. de Freitas, "Critic regularized regression," *arXiv*, 2020.
5. Z. Wang and T. Hong, "Reinforcement learning for building controls: The opportunities and challenges," *Applied Energy*, vol. 269, 2020.
6. 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 Transactions on Smart Grid*, vol. 8, no. 5, pp. 2149–2159, sep 2017.
7. Z. Zhang, A. Chong, Y. Pan, C. Zhang, and K. P. Lam, "Whole building energy model for HVAC optimal control: A practical framework based on deep reinforcement learning," *Energy and Buildings*, vol. 199, pp. 472–490, sep 2019.
8. G. Dulac-Arnold, R. Evans, H. Van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, B. Coppin, and G. Deepmind, "Deep Reinforcement Learning in Large Discrete Action Spaces," Google DeepMind, Tech. Rep., 2015.
9. M. Han, R. May, X. Zhang, X. Wang, S. Pan, D. Yan, Y. Jin, and L. Xu, "A review of reinforcement learning methodologies for controlling occupant comfort in buildings," *Sustainable Cities and Society*, 2019.
10. Z. Afroz, G. M. Shafiullah, T. Urmee, and G. Higgins, "Modeling techniques used in building HVAC control systems: A review," pp. 64–84, mar 2018.
11. R. Jia, M. Jin, K. Sun, T. Hong, and C. Spanos, "Advanced building control via deep reinforcement learning," *Energy Procedia*, vol. 158, pp. 6158–6163, 2019.
12. L. Monier, J. Kmec, A. Laterre, T. Pierrot, V. Courgeau, O. Sigaud, and K. Beguir, "Offline Reinforcement Learning Hands-On," *arXiv*, nov 2020.
13. T. Seno, "d3rlpy: An offline deep reinforcement library," https://github.com/takuseno/d3rlpy, 2020.
14. F. Elmaz, R. Eyckerman, W. Casteels, S. Latré, and P. Hellinckx, "CNN-LSTM Architecture for Predictive Indoor Temperature Modeling," IDLab - Faculty of Applied Engineering, Antwerp, Tech. Rep., 2021.