Vrije Universiteit Amsterdam

Deep Reinforcement Learning for temperature setpoint control

Master Thesis Business Analytics

Michiel Kempkens (2595628)

First supervisor	Vincent François-Lavet
Company supervisor	Joshua van den Heuvel
Second reader	Alessandro Zocca

A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics

April 6, 2023

Abstract

In this research, Deep Reinforcement Learning is used to determine heating and cooling setpoints. In conventional control methods, setpoints are based only on business hours and business days. In this study, environmental factors are also considered in determining the setpoints. The algorithm checks these factors every 15 minutes and then determines what action to take. The deep reinforcement learning algorithms used are Deep Q-Network (DQN) and Proximal Policy Optimization (PPO). The algorithms are tested in two different environments, a simple building and a more complex office building. This research shows promising results for the simple environment. 10.7% energy can be saved while the comfort range is violated only 8.97% of the time. The results for the complex environment are less promising. It can be concluded that it is not possible to outperform the conventional control method with the available computational power. Further research is needed to use DRL in complex environments.

Acknowledgement

I would like to thank the team at EDGE next for allowing me to complete my internship with them. I learned a lot during my time at EDGE next. Everyone was very helpful, which helped me a lot with my research. I would especially like to thank Joshua van den Heuvel for his great support. His daily involvement and good insights helped me tremendously. I would also like to thank Vincent François-Lavet for the useful sessions at the VU. His expertise in the field of reinforcement learning has helped me a lot in this research. Finally, I would like to thank Alessandro Zocca for being my second reader.

Contents

List of Figures

List of Tables

Table of notation

Table 1: Table of notation

Sets:	
S	set of states (s $\in \{1, ..., S\}$)
A	set of actions (a $\in \{1, ..., A\}$)
T	set of transitions (t $\in \{1, ..., T\}$)
R	set of rewards (r $\in \{1, ..., R\}$)
H	set of hours (h $\in 0, ..., 23$)
Parameters:	
s_t	state at timestep t
a_t	action at timestep t
R_{energy}	total energy reward
λ_e	constant to scale energy usage
E_{usage}	energy usage in kWh for each timestep t
$\mathrm{R}_{comfort}$	total comfort reward
P_c	comfort penalty per thermal zone
λ_c	constant to scale comfort violation
R_{total}	total reward
W	weight for energy/comfort component
W_{hourly}	hourly weight for energy/comfort component

1 Introduction

Energy consumption accounts for a significant portion of global greenhouse gas emissions [19]. Energy conservation can help reduce greenhouse gas emissions, slow climate change, and mitigate its effects. It is important to protect the environment and ensure a sustainable future for our planet.

Another reason to reduce energy consumption is the high energy prices due to the current energy crisis. The energy crisis began in 2021, after the COVID crisis ended. The world economy started to recover slowly. This led to an increase in energy demand that could not be met quickly enough. In conjunction with Russia's invasion of Ukraine, energy prices have risen to record levels over the past two years. Energy bills have nearly doubled for most European households, and energy has become one of the largest expenses for building owners.

The real estate sector is one of the largest energy-consuming sectors, accounting for around 40% of the total energy consumption [11]. This number is even expected to grow in the coming years. Within commercial real estate the biggest energy consumer is the Heating, Ventilation and Air Conditioning (HVAC) system. It accounts for 50% of the energy used within the real estate sector.

An HVAC system is used to control temperature, humidity and air quality in a building. It is used to maintain a comfortable and healthy indoor environment. Since this system accounts for a large portion of energy consumption in a building, many studies have been conducted to reduce energy consumption. The biggest challenge in achieving this goal is maintaining an ideal climate for the building occupants.

The most commonly used control system is simple rule-based control (RBC). RBC-based control is usually static and determined by the experience of engineers and facility managers. It is not a continuous control system and can lead to energy waste[10]. In the RBC method, heating and cooling setpoints are determined in advance. In this traditional HVAC control approach, a fixed setpoint schedule is established for temperatures in a building based on expected occupancy patterns and environmental conditions. The HVAC system then operates to maintain these setpoints regardless of actual occupancy and ambient conditions. This approach has several limitations. As mentioned earlier, the traditional approach does not account for changes in occupancy patterns or ambient conditions, resulting in inefficient energy use. This type of control also provides limited control options. The traditional approach does not provide granular control options for HVAC systems, such as controlling the operation of individual HVAC components. This lack of control options can limit the effectiveness of the HVAC system in maintaining a comfortable and healthy indoor environment. In addition, the traditional approach does not provide feedback on HVAC system performance, making it difficult to identify and address inefficiencies [14].

One of the main advantages of RL is that it can adapt to changing conditions in real time, making it well suited for dynamic environments such as commercial real estate. This allows building managers to respond quickly to changes in occupancy, weather, and other factors that affect energy consumption. In addition, RL algorithms can be updated over time to incorporate new information and further improve performance.
This research investigates whether the HVAC system can be controlled to be more energy efficient using RL. The goal is to reduce the CO_2 footprint of commercial buildings. Since the HVAC system is responsible for the majority of energy consumption, the focus will be on investigating new methods of controlling the HVAC system to optimise energy consumption and maintain indoor conditions at the highest possible comfort level. An answer will be provided to the research question: *Can Deep Reinforcement Learning optimise temperature setpoint control?*

1.1 EDGE Next

The internship took place at EDGE next, a subsidiary of EDGE technologies. EDGE technologies is an international real estate developer that specialises in a new generation of high-tech buildings that put the health and well-being of tenants and the planet first and foremost. Each EDGE building guarantees an innovative, healthy, and sustainable modern workplace for its tenants.

EDGE Next helps achieve this goal. The company combines its real estate expertise with smart technology and Big Data. The platform, EDGE Next, provides a solution to optimise the performance of any office building, making it smarter, healthier and more sustainable. The platform uses multiple sensors and sources to not only collect data, but also provide valuable and actionable insights.

One of the company's biggest challenges is to improve building performance. This is done by focusing on four main pillars: Employee well-being, space utilisation, sustainable performance, and operational efficiency. In this research, the focus is on sustainable performance, taking into account employee well-being. The goal is to achieve a high level of comfort in the office while reducing the CO2 footprint by minimising energy use.

The internship took place at the EDGE Olympic building. This is the headquarters of EDGE technologies and EDGE Next, located at the Fred Roeskestraat 115, Amsterdam. This building is also used in this research for simulation and testing the RL models.

Figure 1: EDGE Olympic.

1.2 Problem statement

To better understand the problem, we must first understand how HVAC energy is used in the EDGE Olympic building. As mentioned earlier, most building management systems (BMS) control the temperature in their building in a conventional manner. In the EDGE Olympic building, the ideal temperature range during business hours is between 20 and 24 degrees Celcius. On the EDGE Next platform, this range is classified as category A. So this range is the ideal temperature in the building. Also by guidelines of the WVOI (werkgeversvereniging onderzoekinstellingen) the temperature ranges in offices in the Netherlands have to be within that range [21].

Maintaining a comfortable indoor temperature is important for the health and well-being of employees. A temperature that is too low or too high can cause discomfort, leading to decreased productivity and an increased risk of health problems. The comfort range is formulated to improve the indoor environment to increase health and productivity.

As discussed, the building used is the EDGE Olympic building. At the moment, the HVAC system is controlled by RBC methods. These methods are based on a set of predefined rules to control the system. They use upper and lower set points to control the temperature within certain boundaries. In the case of EDGE Olympic the following rules are set, on business days the system is 'on' from 08:00 until 19:00. The lower boundary is 20°C and the upper boundary is 24°C. On the weekends and non-working hours, the system is on 'standby' mode and the boundaries are set at 18°C and 25°C.

This static way of controlling the temperature does not take any of the environmental factors into account while these factors show a high correlation with the indoor temperature. This study will investigate if energy could be saved if environmental factors are taken into account while determining to heat and cooling setpoints.

1.3 Thesis outline

The next section of this paper discusses some relevant studies related to this research. The first studies on the optimization of the HVAC system and the first studies on the use of DRL for this problem are briefly discussed. This is followed by a general explanation of DRL and why this type of machine learning is useful for this particular problem. The Markov decision process is then explained. This includes all the special features for this DRL problem. For example, the reward function, the action space, and the observation space. In section 5, all algorithms are explained in detail, along with RBC, which is used as a benchmark in this research. Now that all the methods and all the specifics of the problem are known, the simulations can be performed. How the simulations are performed and which programs are used for them are explained in section 6. After that, the results of the simulations are analyzed. After the conclusion, future work is discussed and it is mentioned what needs to be done before this research can be implemented in the real world.

2 Literature research

The modern era of HVAC control began in the 1970s with the introduction of digital control systems [18]. These systems are among the most widely used control technologies in the building automation industry. Digital control systems are computer-based systems that use digital signals and processors to monitor and control various building systems such as HVAC. These were the first systems that could control HVAC systems based on data collected from sensors. The most commonly used control system is feedback control. Feedback control is a technique used in HVAC systems to maintain desired temperatures and air quality in a building. The control system uses sensors to measure the actual temperature in a building and compare it to set points established by the building operator. If there is a difference between the actual and desired values, the control system adjusts the HVAC system to bring the building conditions back to the desired temperature.

One of approaches to optimize the HVAC systems is with Model Predictive Control (MPC) control. MPC is a sophisticated control strategy that can be used in HVAC systems. MPC involves using a mathematical model of the system to predict its behavior over a certain time horizon, and then using this prediction to optimize the control inputs [1]. Privera et al. (2014) presented a predictive controller uses both weather forecast and thermal model of a building to inside temperature control [17]. The model was tested in a real university building and achieved savings around 20%. The main difficulty in applying MPC is that it is labor intensive and requires specialized knowledge. It remains a challenge to generalize a common building energy model for numerous buildings because each building and its energy systems are different. For this reason, MPC has not yet gained widespread acceptance in the building sector, despite its positive results [24].

One of the first attempts to apply RL to HVAC control dates back to 1977. In their research, Anderson et al. (1977) described a simulated heating coil system that used a combination of RL and neural networks [2]. This system was compared with a proportional-integral controller (PI). Both systems attempted to control the temperature of the coil. The purpose of both methods was to maintain a specific temperature while minimizing energy consumption. The RL algorithm used in this study was Q-learning, which will be discussed in more detail later. The results of the study showed that the RL system outperformed the PI controller in terms of energy efficiency. The authors noted that RL enabled the system to adapt to changes in the environment, such as variations in outdoor temperature while maintaining optimal performance.

The main objective of these studies is to explore the use of RL for HVAC control. Specifically, the studies aim to investigate how RL can be used to optimize the energy efficiency and performance of HVAC systems in different settings, such as commercial and residential homes. Each approach to achieve this objective is slightly different. Barret et al. (2015) proposes an RL algorithm that uses a model-based approach to learn a control policy for an HVAC system in a commercial building [4]. Specifically, the algorithm uses a Q-learning approach to learn an optimal control policy that maximizes energy efficiency while maintaining occupant comfort levels. The algorithm is tested using a simulation model of a commercial building, and the performance of the RL approach is compared to that of a traditional RBC strategy.

Later, DRL was also used. In their study, Wei et al. (2017) use a DRL algorithm that employs a neural network to learn a control policy for an HVAC system in a building [25]. The algorithm uses a combination of a deep-Q network (DQN) and a dueling network architecture to learn an optimal control strategy that maximizes energy efficiency and minimizes energy consumption while maintaining indoor comfort. The algorithm is tested using a simulation model of a building, and the performance of the DRL approach is compared to that of a conventional RBC strategy.

Azuatalam et al. (2020) used proximal policy optimization (PPO) to control temperature setpoints [3]. The study uses a virtual testbed to demonstrate the effectiveness of RL use in HVAC control. This method is most commonly used in previous research and allows setpoints to be adjusted in real time. The work shows that PPO is a promising algorithm for the goal of optimizing HVAC systems. The authors were able to achieve significant energy reduction while taking comfort into account.

Overall, the results of previous research indicate that DRL-based HVAC control approaches can achieve better energy efficiency and lower energy consumption compared to conventional control-based control strategies, while maintaining indoor comfort for building occupants. In addition, multi-agent DRL approaches can be particularly effective for HVAC control in larger, more complex buildings. The aforementioned algorithms performed particularly well and are therefore tested in this research.

3 (Deep) Reinforcement learning

With reinforcement learning (RL) systems can learn to predict the consequences of decisions and optimize their behaviour in environments [7]. In these environments actions leads them from one state to the next one while gathering rewards or punishment for their actions. In Figure 2 an overview of reinforcement learning is given.

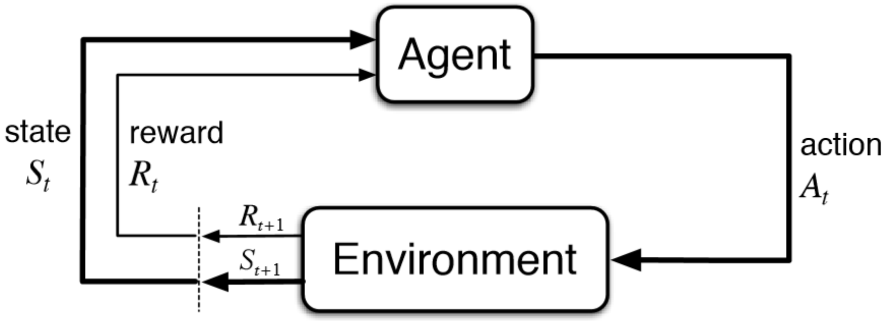

Figure 2: Reinforcement learning outline.

In RL, any action the agent takes leads to two consequences, receiving a immediate reward, and arriving at a new state [24]. The agent could not just simply select the action with the highest reward, it needs to consider the delayed future rewards corresponding to the new state. For instance, the action of pre-heating will lead to higher immediate energy consumption. However, in the long term, the new state might save energy cost. RL can optimize this trade-off between these short and long term benefits. This makes RL a really good fit for this particular research. In most buildings it is ready to use. The behaviour of the environment is unknown to the controller which RL can handle perfectly. The agent can find out the optimal policy without modeling the environment.

For this reason, only Model-free RL algorithms are used. Model-free RL algorithms learn the optimal policy without the need of explicitly modeling the environment. A model-free approach learns to directly map states to actions based on trail-and-error experience [16]. This approach is generally faster and less computationally expensive than model-based RL.

Within model-free algorithms there are two different types of algorithms, value-based and policy-based algorithms. The differences are in how they update the behavior of the agent. Value-based algorithms focus on estimating the optimal value function. The value function is used to determine the optimal policy, which is the action that maximizes the expected reward for a given state. Policy-based algorithms focus on learning a policy that directly maps states to actions without explicitly estimating the value function. The policy is represented by a neural network that takes the current state and outputs an action. The agent learns to adjust the parameters of this function to maximize its expected reward. An overview on how all the different types relate to each other is shown in Figure 3.

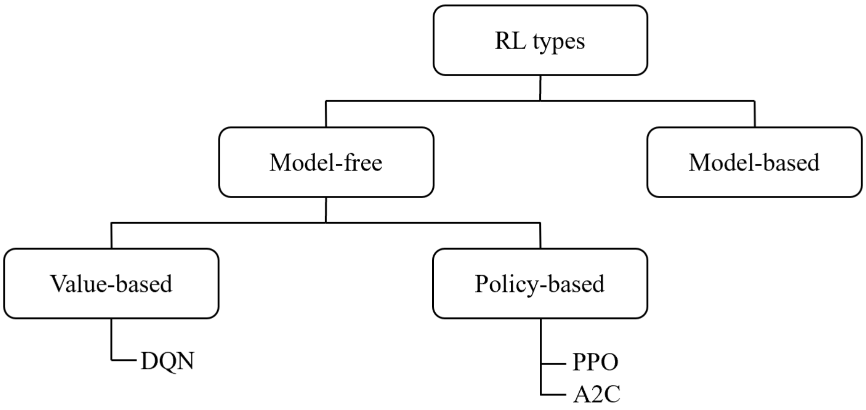

Figure 3: Reinforcement learning types.

Next to these types of RL algorithms, Each type can be on- or off-policy [9]. The on-policy approach updates the policy based on the actions that the current policy generates. That is, they learn from the experience of following the current policy, which is also used to generate new experience. On the other hand, the off-policy approach updates the policy based on the actions generated by a different policy. This allows the agent to learn from a larger variety of experiences, including those generated by other policies. The action space of the used models and wheter they are on- or off-policy can be seen in Table 2.

Algorithm	**Action space**	**Type**
PPO	Discrete / Continuous	On-Policy
DQN	Discrete	Off-Policy
A2C	Continuous	On-Policy

Table 2: Deep reinforcement algorithms.

Although RL seems well suited for this research, there are still some complications. The environment is quite complicated. Many variables need to be considered to make good predictions based on real-life circumstances of the environment. Therefore, deep reinforcement learning (DRL) is used. DRL uses deep neural networks as function approximators to handle high-dimensional state spaces and complex decision-making tasks [12]. All the algorithms used in this research will be explained in detail in section 5.

4 Markov Decision Process

A Markov Decision Process (MDP) is a mathematical framework for modeling decision-making in situations where an agent interacts with an environment over a sequence of time steps.

A MDP is defined as a tuple (S, A, T, R) with the following components:

- States: The states $S_t \in S$ represent each state possible in the environment at each time step t $t \in \{0, 1, 2, ...\}$.

- Actions: The actions $A_t \in A$ represent the different possible choices that the agent can make at each time step t

- Transitions: The transitions define the probability of moving from one state to another as a result of taking an action [23]. The transition function T is defined as $T : S \times A \times S \to [0, 1]$. This defines the conditional probability distribution of the next state s_{t+1}, given the current state s_t, and action a_t. This can be expressed as:

$$T(s_t, a_t, s_{t+1}) = \mathbb{P}(s_{t+1}|s_t, a_t) \tag{4.1}$$

- Reward: The reward function takes the current state s_t, the action taken a_t, and the next state s_{t+1} as input. It returns the immediate reward obtained by the agent for this transition. The reward can be denoted as:

$$R(s_t, a, s_{t+1}) \tag{4.2}$$

- Policy: The policy π is a mapping from states to actions that specifies what action the agent should take in each state. It can be expressed as:

$$\pi(a|s) = \mathbb{P}(a|s) \tag{4.3}$$

The goal of an MDP is to find an optimal policy that maximizes the expected cumulative reward over time. This is typically achieved using RL algorithms that iteratively update the policy based on the agent's experience with the environment. In this section, we specify the reward function, action space, and observation space for this specific problem.

4.1 Reward function

The reward function is a crucial component that defines the objective of the agent. The reward function serves as a feedback signal to the agent, guiding it towards good decisions and shaping its behaviour.

In this research there are different rewards functions created and tested. The reward function consist of two components; the energy usage and the thermal comfort of the building occupants. The objective of this research is to minimize the energy use while while giving the users of the building an high thermal comfort.
The power component is calculated as follows:

$$R_{energy} = \lambda_e * E_{usage} \tag{4.4}$$

Where λ_e is a constant to scale the power variable. In the simple environment, this variable is set to 1.0 as default. In the complex environment, this variable is set to 0.5 as default. The reason is that in the complex office more energy is used per timestep. So to make the energy penalty in the same range as the comfort penalty λ_e is adjusted. Otherwise the energy penalty becomes too large and the algorithms will focus on energy savings completely.

The comfort component can be calculated in multiple ways. The first way is linear. This is calculated as follows:

$$P_c = \begin{cases} T_{indoor} - T_{min}, & \text{if } T_{indoor} < T_{min} \\ T_{max} - T_{indoor}, & \text{if } T_{indoor} > T_{max} \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

In words, the linear comfort penalty how much the indoor temperature is outside the comfort range. The second way of calculating the comfort penalty is the exponential way. This is calculated in the same way but the exponential is taken as shown in Equation 4.6.

$$P_c = \begin{cases} e^{T_{indoor} - T_{min}}, & \text{if } T_{indoor} < T_{min} \\ e^{T_{max} - T_{indoor}}, & \text{if } T_{indoor} > T_{max} \\ 0, & \text{otherwise} \end{cases} \tag{4.6}$$

Where T_{min} is the lower bound of the comfort range and T_{max} is the upper bound of the comfort range. The comfort penalty is calculated for each floor separately. The total comfort penalty is the sum of all the comfort penalties of each floor:

$$R_{comfort} = \lambda_c * \sum_{i=1}^{5} P_{c_i} \tag{4.7}$$

Now that the two components of the reward function are know, the total reward function is:

$$R_{total} = -W * R_{energy} - (1 - W) * R_{comfort} \tag{4.8}$$

Where $W \in (0, 1)$ is a weight that can be adjusted. This weight determines how much you want to focus either on energy saving or thermal comfort. The default setting of W is 0.5. This means that there is a 50% focus on saving energy, and 50% focus on thermal comfort. λ_c is an constant to scale the comfort penalty. The default setting is 1.

Next to these options an hourly weighted reward is created. This means that weight can be set by the hour. With this setting we can create an option to focus more on energy saving in non-working hours and more on thermal comfort in working hours. This hourly weight is determined in the following way:

$$W_{hourly} = \begin{cases} W_{business}, & \text{if } h \in H \\ 1, & \text{if } h \notin H \end{cases} \tag{4.9}$$

Where $h \in 0, 1, 2, ..., 23$ is the hour of the day, and H are working hours. As will be explained in Section 5.1, working hours are from 08:00 until 19:00. $W_{working}$ is a variable that can be changed to preference. This is the weight of R_{energy} during working hours. By making the weights time dependent we can learn the algorithm to focus on energy savings completely during the non-working hours and focus more on comfort during working hours.

4.2 Action space

The action space refers to the set of all possible actions that an agent can take in a given environment. It is the range of all available choices an agent can make at any given state. The action space can be discrete, where a finite number of actions are available, or continuous, where an infinite number of actions can be taken. As can be seen in Table 2, both discrete and continuous action space are used in this research. The goal of the agent is to learn the optimal policy, which is a mapping from states to actions that maximizes the expected cumulative reward over time.

Action	Heating setpoint	Cooling setpoint
0	15	30
1	16	29
2	17	28
3	18	27
4	19	26
5	19	24
6	20	23
7	20	24
8	21	24
9	21	25
10	22	24
11	22	25

Table 3: Discrete action space.

As can be seen in Table 3, there are 12 different discrete actions that can be taken (Discrete(12)). These setpoints are set for different reasons. The first two setpoint ranges are large, so the system can set the setpoints low for heating and high for cooling to save energy at night or on weekends. For the last actions, the ranges are smaller to make it easier to stay in the comfort zone. In the simple environment there are only two setpoints to set. In the complex environment there are 5 thermal zones. Thus 5 heating and cooling setpoints need to be determined. For the discrete action space, the selected actions are the same for each thermal zone. So if the action 0 is executed, the heating setpoint for each thermal zone is 15.

Action	**Variable name**	**Min**	**Max**
0	Heating setpoint	15.0	22.5
1	Cooling setpoint	22.5	30.0

Table 4: Continuous action space.

Table 4 shows the continuous action space. With a continuous action space the algorithm is free to choose a heating setpoint in between 15.0°C and 22.5°C and a cooling setpoint in between 22.5°C and 30.0°C. In the continuous action space the setpoints can differ for each thermal zone. In the complex environment, the setpoints are therefore determined separately for each floor.

4.3 Observation space

The observation space is the set of all possible states that an agent can perceive in a given environment. It defines the range of all observable information available to the agent at any given point in time. The observation space can also be continuous or discrete and is defined by the characteristics of the environment and in this case, the sensor data available in the building. Because sensor data is used the observations are continuous in this research.

The agent uses the observations to determine its current state and to make decisions on what actions to take. The observation space, together with the action space and the reward function, are the key components of a RL problem.

The more observations you add to the environment, the more information must be processed by the agent. This may lead to the fact that it takes longer to learn the relevant features and relationships between the observations and actions. As the observation space becomes more complex, it may become more difficult to find an appropriate representation for the observations, which can affect the efficiency and accuracy of the learning process. Moreover, a larger observation space can also increase the difficulty of the exploration problem, which refers to the challenge of exploring different parts of the environment to learn the optimal policy. With more observations, the agent must explore a larger space of possible states to discover the most rewarding actions, which can require more exploration and, in some cases, more time to converge to the optimal policy.

For this reason, observation variables must be chosen carefully. For this research, a simulation is made with an empty action space. So no setpoints are set, all the possible observation variables are included and the one that shows the most correlation with inside temperature and energy use is taken into account. For this purpose, a simulation is run with an empty actions space. So no actions are taken during the simulation. The observation variables that show the most correlation with the inside temperature and energy usage are taken into account in this study.

Variables	**Zone Air Temperature**	**kWh**
Outdoor temperature	0.543	-0.423
Outdoor Humidity	-0.468	0.063
Wind speed	0.077	-0.030
Wind direction	0.079	-0.100
Solar radiation rate	0.655	-0.154

Table 5: Correlation matrix of the observation variables.

In Table 5, the factors are shown that have the most correlation with the indoor temperatures. The outdoor temperature and the solar radiation rate show the most positive correlation and are taken into account for this reason. Table 6 shows all the observation variables that are taken into account during simulation. Next to the variables that show a high correlation with indoor temperatures the current date and time are also taken into account. The total energy used by the HVAC system is also included to be able to determine the energy usage during the simulation.

Observation variables
Current month
Current day
Current hour
Outdoor Air Temperature
Solar Radiation Rate per Area
Facility Total HVAC Energy use (kWh)

Table 6: Observation variables.

After the observation variables are chosen, they are scaled to a common range or zero-mean and unit variance. This can be important because RL can be sensitive to the scale of the input variables. Scaling can help improve the stability and convergence of the learning process, the learned policy can be more robust to changes in the environment and better adapt to new situations. If the observation variables have significantly different ranges, the algorithm may focus more on variables with larger values, which can cause sub-optimal performance.

In this research, the min max method is used for scaling the observation variables. It is a technique to scale numerical variables to a fixed range, which is in this case between 0 and 1. The simple formula used for scaling the observations is:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.10}$$

5 Methodology

This section explains which algorithms are used in detail. First, the RBC model is explained. This method is used as a benchmark. Then, the Deep Q-Network and Proximal Policy Optimization is explained.

5.1 Rule-based control

The most common building control system is the RBC method. As explained in section 2, this method relies on some predetermined rules for selecting temperature setpoints. These rules are usually in the form of "if-then" statements that dictate what action to take depending on certain conditions. These rules are programmed into a controller that continuously monitors the temperature and takes action based on the predefined rules. The goal of rule-based temperature control is to maintain a specific temperature range in a specific environment.

The biggest advantage of this method is that it can keep the comfort of the occupants at a high level. The indoor temperature is very likely to be between the limits. The exact schedule is shown in Table 7.

Day	Time	Heating setpoint	Cooling setpoint
2*Business day	08:00 - 19:00	20	24
	Else	18	26
Weekend	-	18	26

Table 7: Rule-based schedule

5.2 Deep Q-learning Network

Deep Q-Learning Network (DQN) uses a deep neural network to approximate the Q-value function. It is an extension of the regular Q-learning algorithm. Q-learning is a model-free RL algorithm that uses traditional tabular methods to store the Q-value function. The Q-value for a state-action pair (s,a) is the expected sum of discounted future rewards. This is done using the Bellman equation:

$$Q(s,a) = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{5.1}$$

Where r is the reward obtained by performing action a in state s, s' is the next state after performing action a, a_{t+1} is the action that maximizes the Q-value in state s_{t+1}, γ is the discount factor that determines the importance of future rewards, and $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$ is the maximum Q-value for the next state s_{t+1}.

DQN is preferred to Q-learning in this study. The reason is that it can handle larger and more complex state spaces. Q-learning requires a Q-table to store the Q-values for each possible state-action pair. In this case, there is a relatively large observation space, as explained in Section 4.3. Moreover, the observations are mostly continuous, since they are observed by sensors, e.g., temperature. DQN can handle large and complex state spaces. The Q-value function is replaced with a neural network to approximate the Q-values. Such that $Q(s_t, a_t) \approx Q(s_t, a_t, \theta)$ [13]. The goal of the is to minimize the loss in Q-values:

$$L(\theta) = (Q_{target} - Q_{predicted})^2 \tag{5.2}$$

where Q_{target} is the target Q-value obtained from the Bellman equation 5.1. $Q_{predicted}$ is the predicted Q-value from the neural network. This will give the formula:

$$L(\theta) = (r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t, \theta)^2 \tag{5.3}$$

5.3 Policy Gradient Methods

Policy gradient methods are a class of RL algorithms that directly optimize a policy function, which maps states to actions, in order to maximize the expected cumulative reward. Unlike value-based methods that estimate the value of each state-action pair, such as DQN, policy gradient methods learn a parametric representation of the policy function and update the parameters using gradient descent [22].

The general goal of policy optimization in RL is to optimize the policy parameters θin so that the expected return $J(\theta)$ is:

$$J(\theta) = \mathbb{E}\Big\{\sum_{k=0}^{H} \gamma^k r_k\Big\} \tag{5.4}$$

where γ^k is the discount factor and r_k is the reward received at each time-step t [15]. The methods update the policy parametererization according to the gradient update rule:

$$\theta_{h+1} = \theta_h + \alpha_h \nabla_\theta J(\theta_t)\big|_{\theta=\theta_t} \tag{5.5}$$

Policy gradient loss:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\Big[log\pi_\theta(\alpha_t|s_t)\hat{A}_t\Big] \tag{5.6}$$

π_θ is the policy. It is a neural networks that takes the observed states from the environment as an input and suggests actions to take as an output. $\hat{A}_t$ is an estimator of the advantage function at step t [20]. $\hat{A}_t$ determines how much better the action taken was than expected. This function makes sure that only the actions that are better than average receive a positive nudge. If the advantage estimate is positive, meaning that the actions that the agent took resulted in better than average return will increase the probability that the actions will be selected again in the future when in the same state. The opposite holds for negative values of the advantage estimate.

5.3.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is one of the policy gradient methods. It is designed to learn policies for decision-making tasks that involve sequential decision-making. PPO is a model-free, on-policy algorithm, meaning that it learns directly from experience and only uses data from the current policy.

The main idea behind PPO is to limit the amount that the policy can change between updates. This is achieved by introducing a constraint on the policy update step, which is based on a clipped surrogate objective function. The clipped surrogate objective function takes the minimum of two terms, one term that measures the probability ratio between the new policy and the old policy, and another term that is a clipped version of the probability ratio. This constraint helps to prevent the new policy from deviating too far from the old policy, which can lead to instability and poor performance.

PPO also incorporates a value function into the policy learning process, which helps to reduce variance and speed up convergence. The value function is used to estimate the expected return for a given state, and is learned alongside the policy. The value function is used to calculate the advantage, which is the difference between the expected return for a given state and the value of that state under the current policy.

PPO typically uses a neural network to represent the policy and value function, and employs a stochastic gradient descent (SGD) optimization algorithm to update the parameters of the network based on the clipped surrogate objective function. PPO is known for its stability and ability to handle a wide range of environments, and has been successfully used to learn policies for a variety of challenging tasks, such as playing games and controlling robots.

The objective function in PPO is:

$$L^{CLIP}(\theta) = \bar{E}_t[min(r_t(\theta)\bar{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\bar{A}_t] \tag{5.7}$$

θ is the policy parameter. $\bar{E}$ denotes the empirical expectation over time steps r_t denotes the ratio of the probability under the new and old policies, respectively $\bar{A}_t$ is the estimated advantage at time t. ϵ is a hyperparameter, usually 0.2.

6 Simulations

To test the various methods, simulations must be performed while the algorithms determine the heating and cooling setpoints. To do this, a connection must be made with a controller within the simulation and the algorithms must communicate with each other. Sinergym is used for this connection. In this section, all aspects of the simulations are discussed. The time step in the simulations is 4, which means that the setpoints are determined every 15 minutes.

6.1 Sinergym

Sinergym is used to create a link between the simulation program and various algorithms. Sinergym is an open source framework for simulation and control of energy buildings to perform HVAC control with DRL. It contains all the imports needed to perform building simulations when training an RL agent. The framework of Sinergym is shown in figure 4.

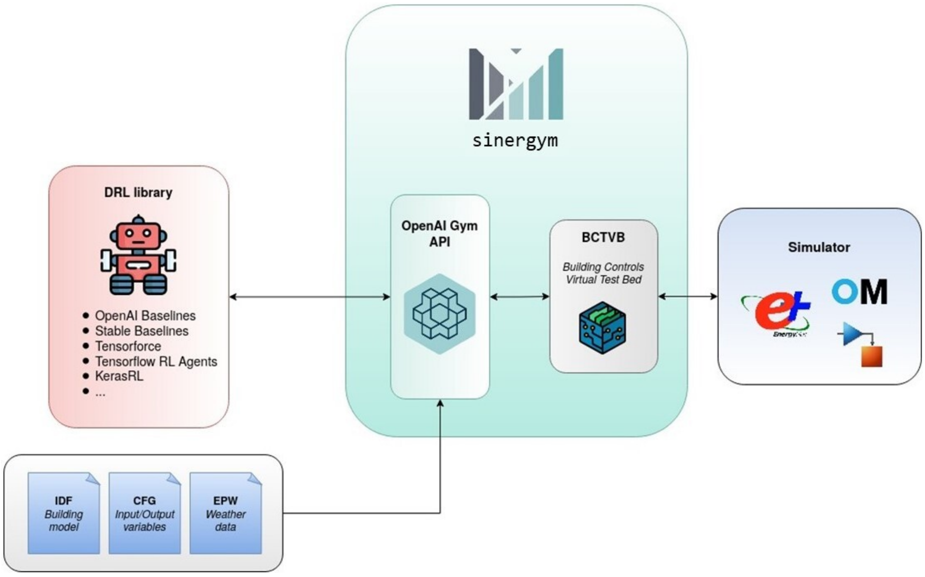

Figure 4: Sinergym.

As shown in Figure 4, sinergym creates the environment for running the simulations. The most important part is the Building Controls Virtual Test Bed (BCVTB). This is software that allows the user to connect different simulators to a control system. BCVTB enables the exchange of real-time data during simulations. This ensures that the agents have all the information in each state and time step of the simulation. The control system is an energy management system (EMS), which is a computer that can be programmed to control building-related energy systems [8].

The BCVTB must be connected to a simulator. EnergyPlus is used in this study, and this simulation program is discussed in more detail in the next section. OpenAI gym is used to create the algorithms described in section 5. OpenAI gym is a toolkit for RL research [5].

6.2 EnergyPlus

All simulations in this study were performed using EnergyPlus. EnergyPlus is a whole-building energy simulation program that uses advanced physical algorithms to model the performance of buildings and their associated HVAC, lighting, and other systems. It is a powerful and adaptable building simulation software that can be used for a variety of purposes, including energy modeling, analysis, and optimization. Therefore, it is perfectly suited for the objective of this research work.

EnergyPlus was developed by the United States Department of Energy (DOE) in 1996 [6]. EnergyPlus is primarily a simulation program without an interface. To visualize the building being simulated, we use Openstudio. This is an open software platform to support energy modeling with EnergyPlus.

To run the simulations, several files need to be uploaded. The building and a weather file to get the climate conditions at each time step. The buildings are stored in an intermediate file (IDF). This file contains all the important information about the building, the geometry, the materials, the location, and of course the HVAC system.

6.3 The environments

In order to perform simulations, a IDF file is needed. This will be the environment of the DLR algorithms. In this study two environments are tested.

6.3.1 Simple building

The first building that is tested is a simple digital building that is publicly available by EnergyPlus. The simple building is shown in Figure 5. In this building only one thermal zone is actively managed.

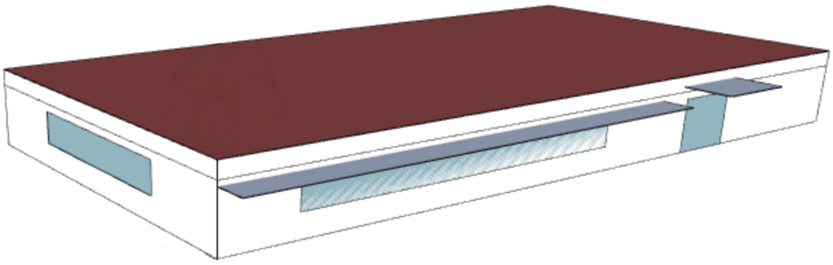

Figure 5: Simple building.

6.3.2 EDGE Olympic

The other building included in this research is the office of EDGE Next. This building is called EDGE Olympic and it is located at the Fred. Roeskestraat 115 in Amsterdam. A digital version of this building is needed before simulations can be performed. All the features of the EDGE Olympic building need to be translated into a IDF file. The energy model of the building is displayed in figure 6. This is a simplified version of the building for energy simulation purposes only.

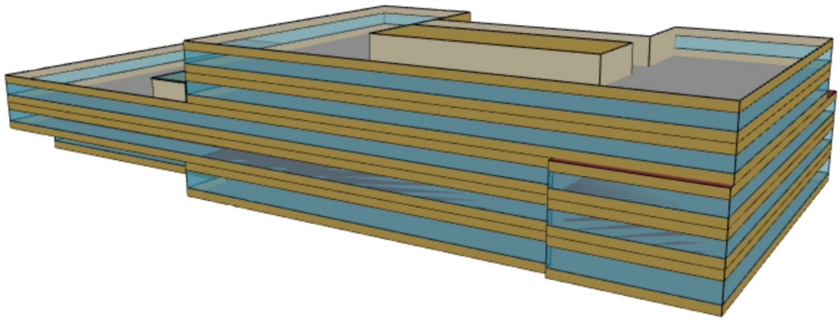

Figure 6: EDGE Olympic model.

The model is simplified to keep the RL problem simple. Which is perfectly fine for the goal of this research. EDGE Olympic is a 5-floor building and each floor is a thermal zone. This means that each floor has its space condition requirements like heating and cooling setpoints, which can be controlled separately.

6.4 Weather file

To perform a good simulation at the right location a weather file has to be included. As mentioned, the office is located in Amsterdam. Therefore an EnergyPlus weather file (EWP) of Amsterdam is added. This includes all the data variables of the weather conditions in Amsterdam, such as temperature, solar radiation rate, humidity, wind speed, wind direction, etc. EWP files can be downloaded from the EnergyPlus website.

6.4.1 Weather variability

When training the model on the same dataset over and over again the probability of over fitting arises. To prevent overfitting from happening, weather variability will be introduced. This will be implemented with the Ornstein-Uhlenbeck process. The Ornstein-Uhlenbeck process can be used to model stochastic processes, such as the weather. This process will introduce noise to the weather data which makes the data slightly different over time.

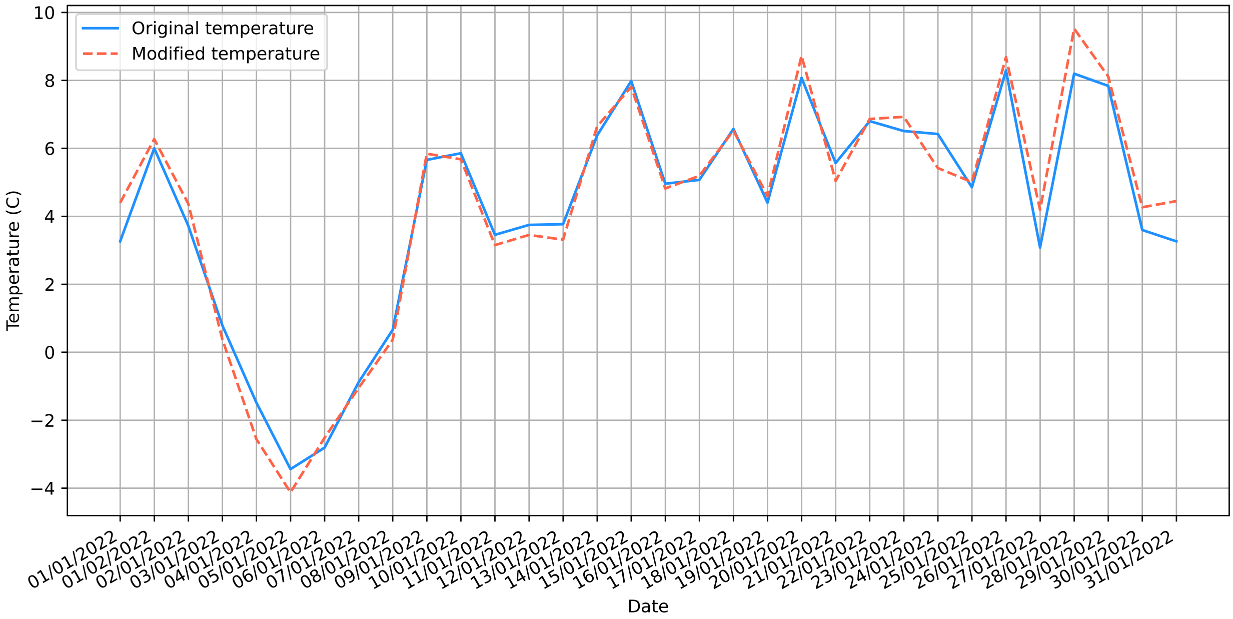

Figure 7: Weather variability for January.

The weather variability in Figure 7 is only an example of what the variability looks like. This example is only for the month of January and only for the temperature factor. The process is done for each weather component that is taken into account in the simulations. As mentioned, due to this weather variability the weather conditions slightly differ in each episode while training the model.

7 Results

In this section, the results will be discussed. The models are trained for 200 episodes. After training, the best model is selected. The best model is based on the average reward generated by the model. This model is then validated 10 times. The results shown are the averages of the 10 validation episodes. The reward function used for the final simulations is explained first. As explained in Section 6.3, two buildings are included in this research. Therefore, these results are presented separately.

7.1 Reward

The reward used in the final simulations is the hourly linear reward. This reward function is described in detail in section 4.1. This means that the weight W changes from business hours to non-business hours. The following choices are made for the variables in the final simulations.

Variable	Value
Business hours	08:00-19:00
$W_{business}$	0.3
$W_{non-business}$	1
Comfort range (Business hours)	(20,24)

Table 8: Variables for simulation.

For the algorithms there is no comfort range in non-working hours.

7.2 The base case

As explained, the method that the algorithms will be bench-marked against is the RBC method. The simulations are done for the year 2022 with the weather conditions of Amsterdam. In Section 5.1, it is explained how RBC is performed. For both the simple environment and the EDGE Olympic building, the RBC method is performed with the same parameters.

7.3 The simple environment

First, the results of the simple environment are discussed. To be able to determine if the algorithms are learning the mean reward of both algorithms is plotted for every episode.

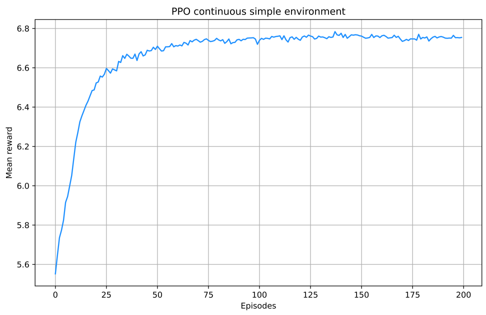

Figure 8: Learning curve PPO.

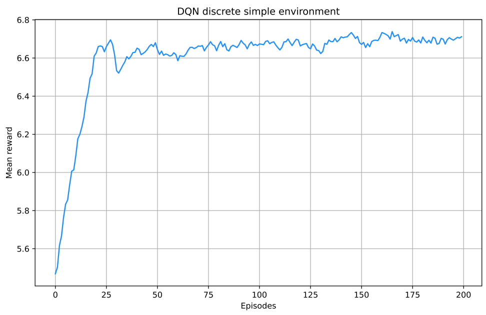

Figure 9: Learning curve DQN.

In Figure 8 it can be seen that PPO seems to converge steadily to a value around 6.7. This is reward is reached after almost 75 episodes. The DQN algorithm in Figure 9 seems to learn less steadily then

PPO. There is a small drop in performance after 25 episodes. After it drops it seems to converge around 6.7 as well.

That both algorithms converge is a good sign but what are the results in terms of energy usage and comfort? To visualize the energy consumption the amount of kilowatt-hour (kWh) is plotted for each month.

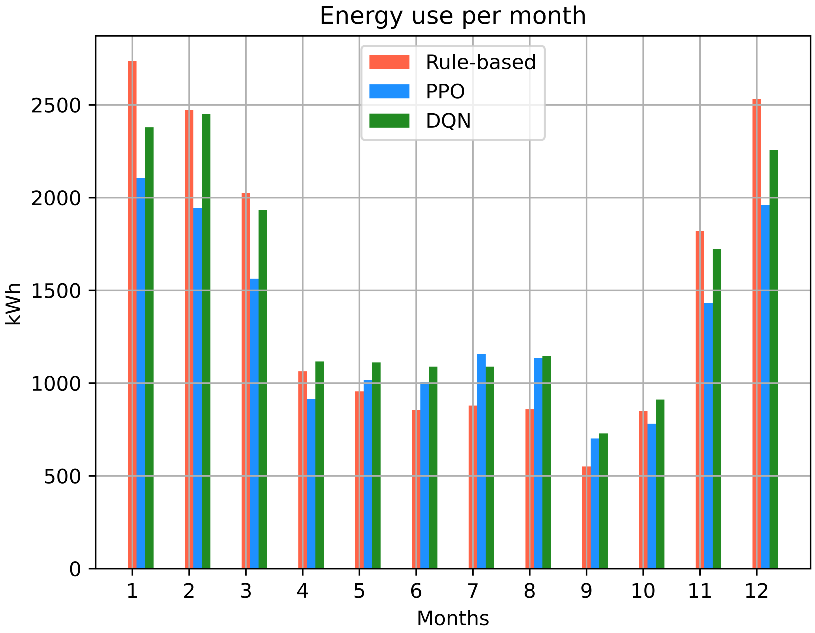

Figure 10: Energy use per month in simple environment.

It can be seen that the algorithms outperform the RBC approach in most of the months. Energy is especially saved during winter months. In the summer the RBC is a little more energy efficient.

As explained, the indoor temperature needs to be between 20°C and 24°C during business hours. During weekends or non-business hours, the indoor temperature does not matter. Therefore, the following violations only hold for business hours.

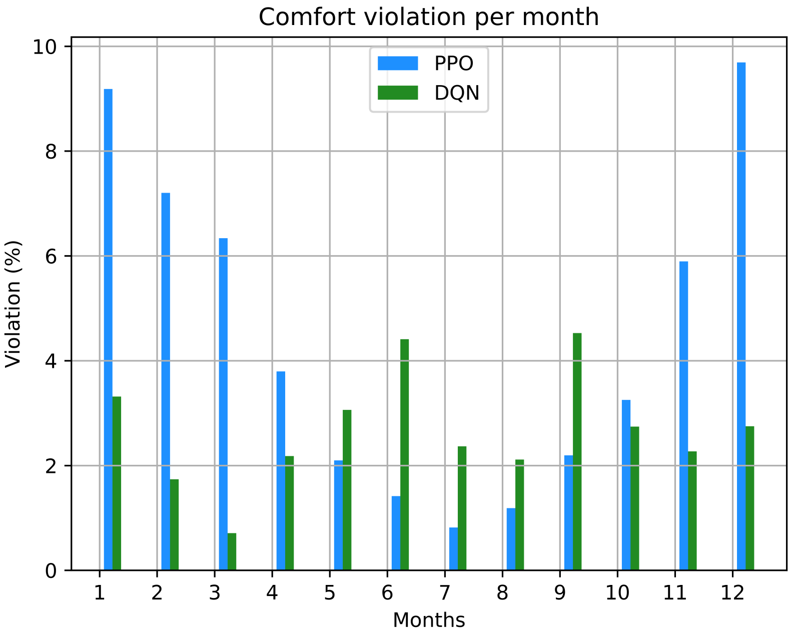

Figure 11: Comfort violation in simple environment.

The comfort violation in the simple environment is minor. In the winter the comfort violation of the PPO is somewhat larger than in the summer. However, it is still within an acceptable range. When investigating the reason why this occurs, it can be concluded that most of the violations are in the first business hour. So the most violations are in between 08:00 and 09:00. Violations occur more in winter than in summer. This can easily be explained due to the fact that the outdoor temperatures are lower and hence it takes more time for the building to reach the desired temperature. Violations in summer are negligible since it is only around 1%. The DQN algorithm seems to show the opposite results in terms of comfort violation. In the summer the comfort range is violated more often, but in the winter this number is lower relative to PPO. Note that the comfort violations are not displayed in Figure 11. The reason is that the indoor temperatures are almost always in between the comfort range due to the static setpoints. Therefore, the comfort violations are negligible.

PPO	Variable
Energy saved (%)	10.70
std dev	0.01
Comfort violation (%)	8.97
std dev	0.01
Mean comfort violation	0.72

Table 9: Result summary PPO.

Table 9 shows a summary of the results of the PPO algorithm. As explained, are these results the averages for 10 validation episodes. The performance is valuable since the algorithm shows that it was able to save 10.7% annually while only violating the comfort range 8.97% of the time on average. If the comfort range is violated, it is by 0.72°C on average. The standard deviation of both the energy saved and comfort range is low. This means that the model can produce steady results over time.

DQN	Variable
Energy saved (%)	-1.89
std dev	0.86
Comfort violation (%)	2.68
std dev	0.06
Mean comfort violation	0.58

Table 10: Result summary DQN.

The DQN algorithm shows less positive results. On average, the model is less energy efficient than RBC. However, the standard deviation is higher than for the PPO algorithm. This means that the performance of the model is less constant and the results of the model fluctuate more. In some episodes, the model performs better than the RBC method and in others, it performs worse. If the average of 10 validation episodes is taken, the model performs 1.89% worse than RBC. The comfort violation is better than PPO, as it violated the comfort range only 2.68% of the time. The standard deviation of comfort violation is low, which means that the model performs stable in terms of comfort violation. The average violation was 0.58°C.

It is interesting to look at the behaviour of the agent in non-business hours. In the RBC method, the choice is made to set the setpoints to 18°C and 26°C for the heating and cooling setpoints respectively. These choices are made so that the building could be on temperature fast as soon as people enter the building in the morning. It would also be more energy efficient to keep the temperature at a certain level during the night.

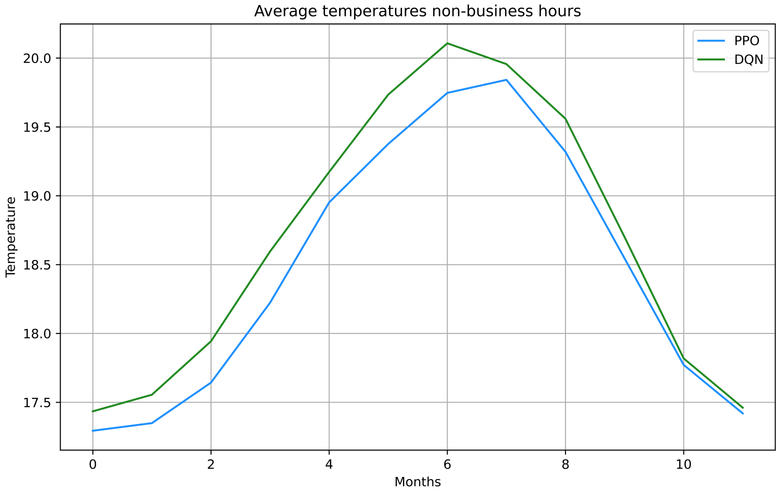

Figure 12: Indoor temperatures in non-business hours.

In Figure 12 can be seen that both algorithms made a choice to keep the indoor temperature at a certain level during the night. Even during winter, the indoor temperatures do not go below 17.5°C. This indicates that it is indeed more energy efficient to not let the indoor temperature drop completely during nights and weekends.

7.4 Results of EDGE Olympic

Unfortunately, the results of the EDGE Olympic building are not as favorable as the results in the simple environment. The environment in this case is much more complex as there are 5 thermal zones. This makes the observation space much larger. There are 5 indoor temperatures that must be within a certain range. This also makes the action space larger. For the discrete action space, all setpoints are the same for each thermal zone. For the continuous action space, the setpoints can be different. This leads to the fact that 10 different setpoints have to be determined in each time step. Together with the large observation space, this results in a very complex problem for the DRL algorithms. In the time available for this study and with the available computational power, it was not possible to create a model in the complex environment that would save energy while taking into account tenant comfort. An additional algorithm was included in the complex environment, the Actor-Critic (A2C) algorithm. The results obtained in the EDGE Olympic building are presented in this section.

The the mean rewards of every algorithm used is shown in the plots below. This indicates how well the agent can learn in this environment with the specific algorithm.

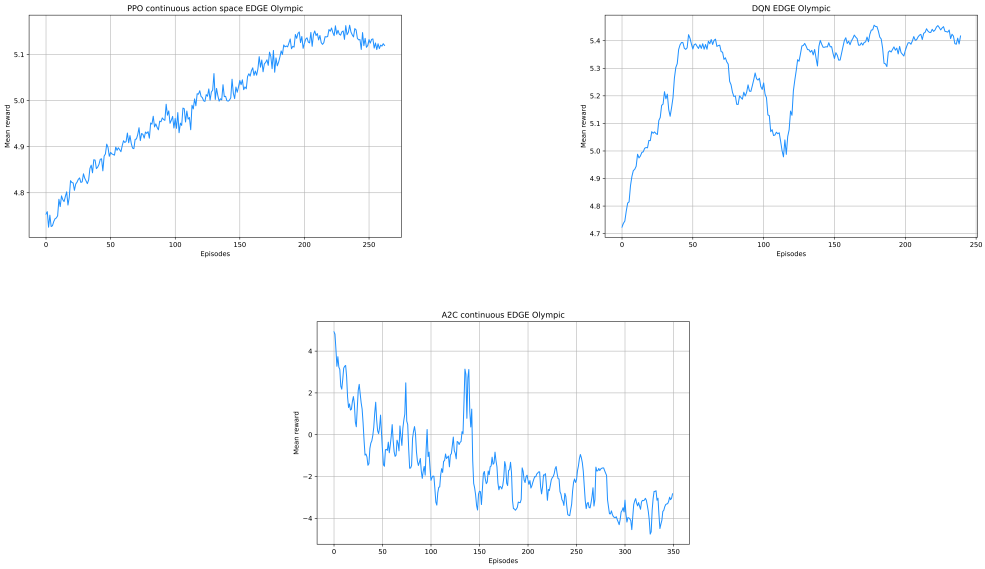

Figure 13: Mean rewards per episodes in the complex environment.

The PPO algorithm seems to be learning smoothly and converges a little above 5.1. The algorithm seems to be improving steadily and is at its peak around 225 episodes. The mean reward of the DQN algorithm seems to converge slowly to a value of around 5.4. There is a lot of divergence in the learning curve of the algorithm. In a good functioning DRL you would like to see a more constant improvement of the mean reward. The A2C algorithm is performing the worst. The algorithm does not converge. What is even worse, is that the mean reward gets lower over time. This indicates that the algorithm is not improving and even get worse over time.

The progress of the mean rewards of DQN and especially PPO seems promising. To check if they perform good results, the energy consumption and comfort violation are checked.

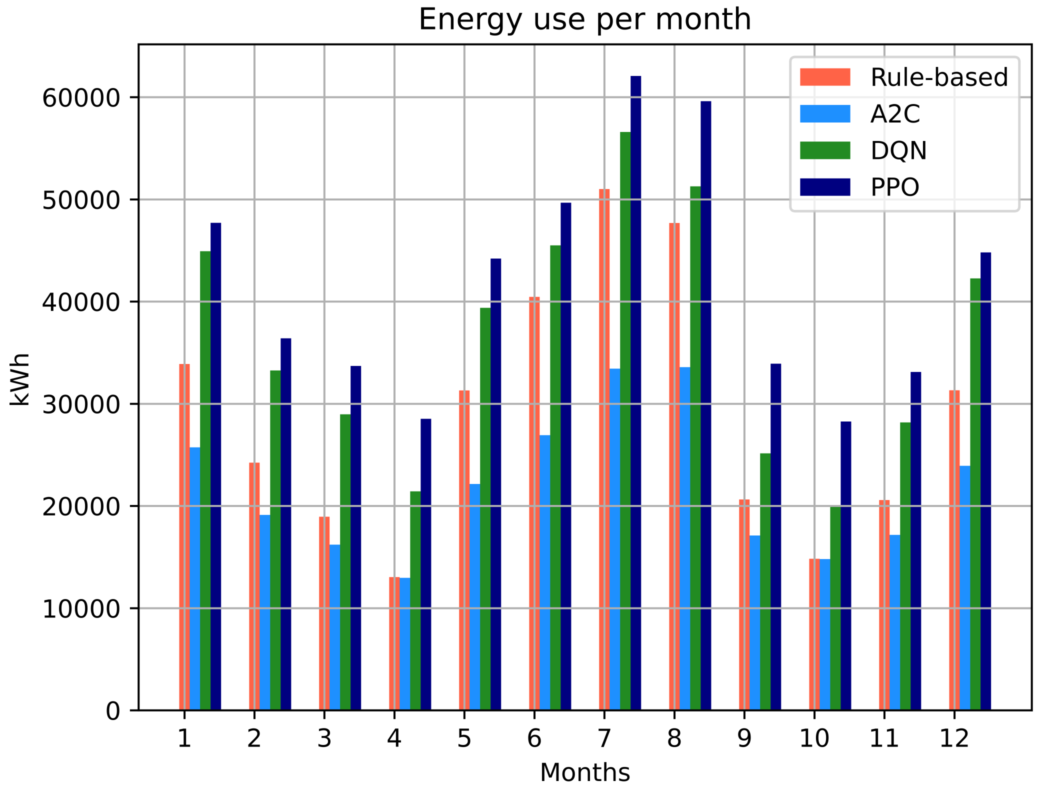

Figure 14: Energy usage per month in EDGE Olympic.

In Figure 14 the energy use per month of all the algorithms is shown. It becomes clear that DQN and PPO consume more energy every month of the year. The only algorithm that performs better than RBC in terms of energy usage is A2C. However, it was already concluded that this model did not converge and became worse over time. To investigate if the agent still performs better than RBC it will be checked if the comfort violation is also better.

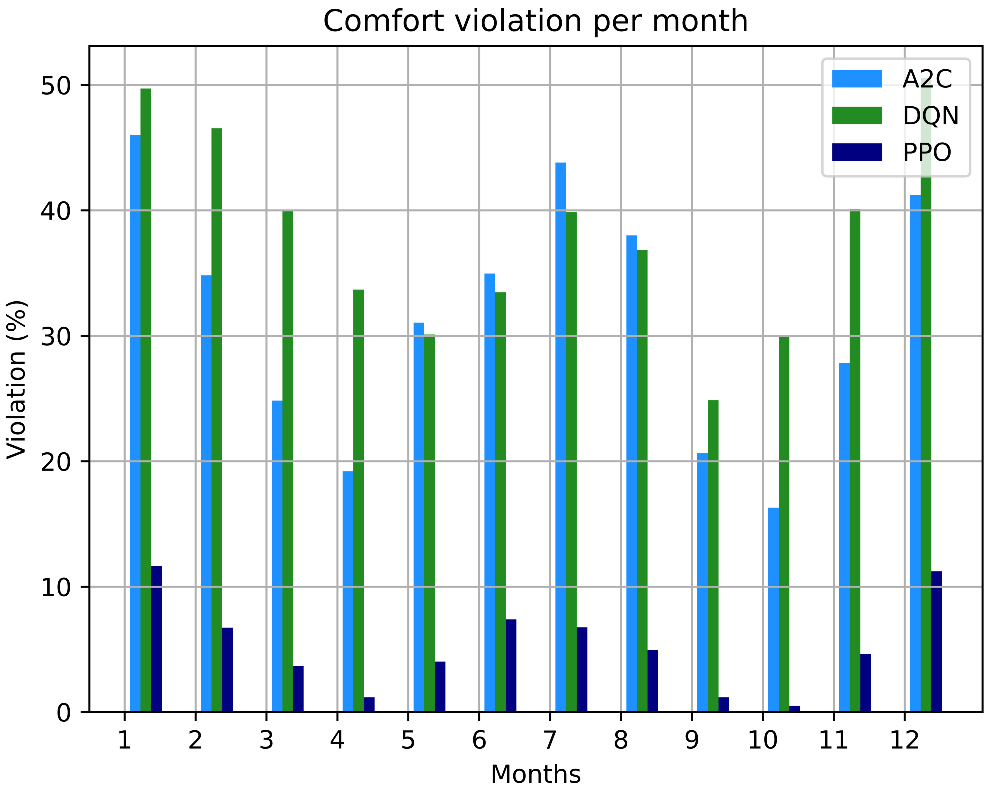

Figure 15: Comfort range violation per month in EDGE Olympic.

From Figure 15, it is clear why A2C achieved such good energy results. The comfort violation is very high almost every month. This means that the indoor temperature is not maintained at 31.5% on average, with outliers of 45% per month. This is not a desirable indoor climate and therefore not a good result. For the DQN algorithm, the comfort violation is even worse. The PPO algorithm performs very well in terms of comfort. However, as explained in Figure 14, the energy consumption with this algorithm is enormous, so this is not a desirable result either.

This problem seems to be too complex to solve for now. Due to a limitation in computational power, the training of the agent takes way too long in such a complex environment. Runs take up to 40 hours to complete which makes parameter tuning a time-consuming task.

8 Conclusion

It can be concluded that Deep Reinforcement Learning is a very promising method for controlling HVAC systems. This research shows that DRL can save a significant amount of energy in a simple environment while considering tenant comfort. The PPO outperformed the traditional RBC by 10.7% in energy savings. The algorithm was also able to maintain the comfort range 91.03% of the time.

Thus, DRL can optimize temperature setpoint control. This answers the research question: *can deep reinforcement learning optimize temperature setpoint control?* In this research, it is shown that DRL can learn to apply the settings of a comfort range within a certain time window. To take advantage of the specifics of the environment along with the current weather conditions, DRL was able to create an agent that adjusts the heating and cooling setpoints to achieve significant energy savings.

In more complicated environments, however, there is still much to improve. Due to a lack of computing power, the benefits of using DRL in complex environments have not been fully explored. More research is needed to achieve the same results as in simple environments. At the moment, the algorithms used are not capable of achieving the same results in a complex environment.

9 Discussion

There is still much to improve in the field of this research. As mentioned earlier, the results in the complex environment are far from ideal. Therefore, the use of DRL in complex environments needs to be further explored. In addition to the environment, other observational variables should also be included. Heating systems are not the only sources of heat in buildings. Lighting, electrical equipment, and people themselves also generate heat. For simplicity, these variables are not included in this research. However, it would be very interesting to see how this would affect temperatures in the building. Occupancy in particular would be a big improvement. Firstly, because they generate heat, and secondly, because a room does not need to be heated/cooled if it is known that this room will not be used that day.

It would also be interesting to focus on more variables than just thermostat setpoints. Ventilation systems are also a large part of HVAC systems and could also be controlled by DRL. Due to lack of time and computing power, the decision is made to focus only on heating and cooling setpoints.

Other DRL algorithms could also be tested. In this research the focus was mainly on DQN and PPO. The reason was that these algorithms performed best in previous studies. However, other algorithms could also be well suited for this problem and need to be tested as well.
Also, more parameter tuning could be done. The DRL algorithm consists of a large number of parameters, each of which individually affects the result. Also, due to lack of computational power, it was difficult to do extensive parameter tuning. Now, only a simple grid search is performed, but more parameters could be tested, which might lead to a better result.

It can be concluded that even better results can be obtained if more computational power is available. Due to the computational complexity of the problem, not all of the above could be considered. It would be very interesting to see how much savings could be achieved if these variables can be included.

9.1 Real world implementation

As explained in section 6.3.2, the building is divided into five thermal zones. Each floor has its thermal zone and is therefore treated as one large room. In real life, of course, this is not the case, and there are several separate rooms on each floor with their own thermostats. Even in the EDGE Olympic building each room can regulate its own indoor climate. In this research, the simplified office building was already too complex to handle. To use DRL in real life, many improvements are still needed, as real buildings are much more complex.

If you want to implement a trained agent to control the temperature setpoint in real buildings. The environment must be closer to the real building. An actual digital twin must be made of the building with all the aspects of the HVAC system and the building materials. Otherwise, the trained agent will perform very poorly when implementing. The agent needs to be trained for multiple episodes before good results can be achieved.

References

[1] Abdul Afram and Farrokh Janabi-Sharifi. Theory and applications of hvac control systems–a review of model predictive control (mpc). *Building and Environment*, 72:343–355, 2014.

[2] Charles W Anderson, Douglas C Hittle, Alon D Katz, and R Matt Kretchmar. Synthesis of reinforcement learning, neural networks and pi control applied to a simulated heating coil. *Artificial Intelligence in Engineering*, 11(4):421–429, 1997.

[3] Donald Azuatalam, Wee-Lih Lee, Frits de Nijs, and Ariel Liebman. Reinforcement learning for whole-building hvac control and demand response. *Energy and AI*, 2:100020, 2020.

[4] Enda Barrett and Stephen Linder. Autonomous hvac control, a reinforcement learning approach. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part III 15*, pages 3–19. Springer, 2015.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[7] Peter Dayan and Yael Niv. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology*, 18(2):185–196, 2008.

[8] Peter G Ellis, Paul A Torcellini, and D Crawley. Simulation of energy management systems in energyplus. 2008.

[9] Rasool Fakoor, Pratik Chaudhari, and Alexander J Smola. P3o: Policy-on policy-off policy optimization. In *Uncertainty in Artificial Intelligence*, pages 1017–1027. PMLR, 2020.

[10] Xi Fang, Guangcai Gong, Guannan Li, Liang Chun, Pei Peng, Wenqiang Li, Xing Shi, and Xiang Chen. Deep reinforcement learning optimal control strategy for temperature setpoint real-time reset in multi-zone building hvac system. *Applied Thermal Engineering*, 212:118552, 2022.

[11] Jingke Hong, Geoffrey Qiping Shen, Shan Guo, Fan Xue, and Wei Zheng. Energy use embodied in china s construction industry: a multi-regional input–output analysis. *Renewable and Sustainable Energy Reviews*, 53:1303–1312, 2016.

[12] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[14] Ivars Namatēvs. Deep reinforcement learning on hvac control. *Information Technology and Management Science*, 21:29–36, 2018.

[15] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2219–2225. IEEE, 2006.

[16] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

[17] Samuel Privara, Jan Široký, Lukáš Ferkl, and Jiří Cigler. Model predictive control of a building heating system: The first experience. *Energy and Buildings*, 43(2-3):564–572, 2011.

[18] Timothy I Salsbury. A survey of control technologies in the building automation industry. *IFAC Proceedings Volumes*, 38(1):90–100, 2005.

[19] David Satterthwaite. Cities' contribution to global warming: notes on the allocation of greenhouse gas emissions. *Environment and urbanization*, 20(2):539–549, 2008.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[21] Sdu-uitgeverij. Arbo-informatieblad nr. 24, binnenmilieu, 2013.

[22] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[23] Martijn Van Otterlo and Marco Wiering. Reinforcement learning and markov decision processes. *Reinforcement learning: State-of-the-art*, pages 3–42, 2012.

[24] Zhe Wang and Tianzhen Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

[25] Tianshu Wei, Yanzhi Wang, and Qi Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6, 2017.

Appendix

In this section all the used parameter are shown. Parameter tuning is done with simple grid search. However, it was very difficult to perform this search due to the long running time. As mentioned, simulations in the simple environment took up to 10 hours and the complex even over 40 hours.

Parameter	**Value**
Hourly weight	0.3
Number of episodes	200
Learning rate	0.0001
Gamma	0.98
Batch size	8192

Table 11: Parameters tested

These parameters are tested in multiple combinations. The values mentioned in Table 11 are the best performing parameters and therefore used in this research.