Master's Thesis

ENERGY EFFICIENT CONTROL OF DATA CENTER HVAC SYSTEMS USING REINFORCEMENT LEARNING

Engineering Technology - Chair of Applied Mechanics and Data Analysis

coolgradient

UNIVERSITY OF TWENTE.

AUGUST 29, 2024

Author

PEPIJN SIX DIJKSTRA

External Supervisor

LENNART VAN DE GUCHTE

Internal Supervisor

PROF. DR. IR. BOJANA ROSIC

This page intentionally left blank.

Summary

Dear reader,

This thesis investigates the use of Reinforcement Learning (RL) to create a controller that can reduce the energy usage of the Heating, Ventilation and Air-Conditioning (HVAC) system in a Data Center (DC), while still meeting critical temperature constraints. The introduction highlights the importance of reducing energy in DC HVAC systems while exploring the current literature on the topic.

Following the introduction comes the problem statement, where a simple HVAC system using a chiller plant is defined, of which a simulation model is created in EnergyPlus. This thesis will design its controller on this plant. Also, the goal of the controller of minimizing HVAC power while handling critical temperature constraints in a server room is formalized in an optimization problem.

Then, the methodology chapter first describes the design of a baseline controller to compare with the RL-based controller. Finally, it proposes an RL framework with a tuneable reward function and consequently, algorithm and reward hyperparameter tuning experiments are set up. These experiments are used to tune the RL-based controller. All frameworks and experiments are implemented in Python.

The numerical results chapter presents the results of the baseline and tuning experiments outlined in the method. From this, 3 tuned RL-based controllers are then selected and compared to the baseline. These 3 controllers are either good at handling constraints, reducing HVAC power and one with a combination of both. From this comparison, it follows that the RL-based controllers can outperform the baseline, but how these agents learn is still unpredictable, as they are sensitive to their initialization.

In the conclusion, the key findings of this thesis are summarized, and the limitations of the simulation model and proposed controller are discussed. Next to that, directions for further research are proposed.

Enjoy reading the thesis!

Kind regards,

Pepijn Six Dijkstra

Contents

List of Abbreviations

AHU	Air Handling Unit 5
AL	Air Loop 12, 14–16, 18, 25, 30, 36, 57, 78–81
API	Application Programming Interface 50, 51
BA	Balanced Agent iv, viii, ix, 71, 73, 74, 78, 79, 81, 82, 105, 106
BES	Building Energy System 2, 6
BMS	Building Management System 2
CC	Cooling Coil viii, 10, 11, 14, 15, 18, 25, 81, 95, 96
CFD	Computational Fluid Dynamics ii, 5–7, 17
ChL	Chilled Water Loop 12, 14–16, 18, 25, 26, 36, 55, 57, 81
COA	Constraint-Optimized Agent iv, viii, ix, 71, 73, 74, 76, 78–81, 107, 108
CoL	Condenser Water Loop 13, 15–18, 25, 27
COP	Coefficient of Performance 15, 19, 96
CRAH	Computer Room Air Handler 6, 10, 12, 14, 18, 25, 31, 32, 67, 86, 94–96
CT	Cooling Tower viii, 12, 16, 18, 27, 86, 97
DC	Data Center i, ii, 1–5, 7–12, 14, 17–19, 21, 23, 24, 28, 36–38, 41, 52, 83, 85–87, 94, 95, 97
DRL	Deep Reinforcement Learning 5
EIR	Energy Input Ratio 18
EOA	Energy-Optimized Agent iv, viii, ix, 71, 73, 74, 76, 78, 81, 83, 103, 104
GAE	Generalized Advantage Estimator 38, 40, 63
HVAC	Heating, Ventilation and Air-Conditioning i–iii, vii, viii, 1–25, 27–32, 34, 36, 37, 40, 41, 44, 46–49, 51–58, 64, 67–69, 71, 74–76, 78–80, 83, 85–87, 94–97

IDF Input Data File 51
IEA International Energy Agency 1
IP Intellectual Property 41
IQR Inter Quartile Range 49, 50
ITE IT Equipment iii, vii, 1–3, 9, 10, 14, 17, 20, 21, 25, 41, 48, 54, 56, 80

MDP Markov Decision Process iii, x, 28, 29, 35, 36, 46
ML Machine Learning 4, 42, 59, 73
MLP Multi-Layer Perceptron 39
MPC Model Predictive Control ii, vii, 3, 4, 7

NN Neural Network 7, 37, 39

PG Policy Gradient 37, 38
POD Proper Orthogonal Decomposition 3
PPO Proximal Policy Optimization x, 36–40, 43, 46, 58, 59, 63, 64, 68, 70, 72, 73

ReLU Rectified Linear Unit 33, 34
RL Reinforcement Learning i–iii, vii, 3–6, 8, 9, 12, 17, 18, 22–25, 28–47, 49–53, 57, 58, 66–68, 71–74, 78, 79, 82–87, 103
RNN Recurrent Neural Network 5

SOO Sequence of Operations 2
SRL Safe Reinforcement Learning 5, 87

TARP Thermal Analysis Research Program 19, 20
TPE Tree-structured Parzen Estimator 43, 45, 46, 59

List of Figures

List of Tables

Chapter 1

Introduction and literature research

Data Centers (DCs) are vital in our current digital world. From scrolling on social media to hosting large amounts of business data, almost everything that happens online is stored and processed in DCs [1]. With an ever-increasing number of global internet users, the demand for DC services has grown with 340% between 2015 and 2022 [2] and it is expected to continue growing in the coming years. The International Energy Agency (IEA) expects the energy demand of DCs to grow from 460 TWh in 2022 to 650 up to 1,050 TWh in 2026 [3].

While these DCs are thus essential for an increasingly digital world, they are also a type of building that is well known for being exceptionally energy-consuming. DCs are responsible for about 2% of worldwide energy consumption [3] and 1% of the global CO_2 emissions related to energy use [2]. To put this into perspective: all of the CO_2 emissions related to energy in the United Kingdom accounted for just 0.84% of the global emissions in 2022 [4]. Since DCs use so much energy, making improvements in their energy efficiency can have a large impact on global energy emissions.

In DCs, most of the power is consumed by the servers or IT Equipment (ITE), and the cooling of this equipment by the Heating, Ventilation and Air-Conditioning (HVAC) system. This means that large energy improvements in DCs can be achieved by improving the energy efficiency of either the ITE or the HVAC system. This thesis focuses on the optimization of the HVAC system. The power used by the HVAC system is largely dependent on its settings. These include desired temperatures, mass flows, and pressures. In a study by Ni and Bai, which compared over 100 DCs, it is seen that the HVAC system is running at sub-optimal settings in most DCs [5]. A lot of research shows that the HVAC systems of both general buildings [6, 7, 8, 9, 10, 11] and of DCs specifically [12, 13, 14, 15] can be optimized by using traditional optimization techniques. Since, on average, $\sim$ 38% of the total energy usage in DCs is related to the HVAC system [5], reducing the energy use of this system has a large impact on the total energy use of a DC.

An additional challenge for the reduction of this energy usage is that DCs are mission-critical, meaning that it is essential that they always keep operating. If they fail when, for example, a server overheats, this has severe consequences for the companies that use these servers. Therefore, this mission-criticality should always be considered when reducing the energy usage of a DC's HVAC system.

This thesis proposes a controller to minimize the energy usage of the HVAC system of a DC

while ensuring safe operation of the DC. This introduction introduces a gap in the research by first reviewing the current situation of the control of DC HVAC systems in section 1.1. Then, existing research into different techniques used to create optimized controllers for HVAC systems are discussed in section 1.2. This is followed by a review of modelling techniques of HVAC systems in DCs in section 1.3. Finally, the goal of this research and an outline of the thesis is presented in section 1.4.

1.1 Control of DC HVAC systems

A DC is essentially a large hall that contains servers. These servers produce heat, which is cooled by the HVAC system. In a DC, the HVAC system is managed by the Building Management System (BMS). The BMS controls the entire Building Energy System (BES), which consists of all energy-using components in a building. The cooling system, the main focus of this thesis, is an important subsystem of the BES. Each part of the BES influences the others, and the BMS ensures their proper control and integration [16, 17]. The BMS often works on multiple levels, which can be divided into component levels and system levels. At the component level, the components, such as pumps or fans, are controlled to ensure that for example desired mass flows, temperatures or pressures are obtained. In the system level controllers, it is determined which desired parameter values are suitable for optimal or safe operation of the system and whether equipment should be switched on or off [17, 18]. An example of this higher-level control system is the control of the chillers, which cool down water that cools server rooms. The higher level control system determines how many chillers should be turned on and it provides a setpoint for the leaving chilled water temperature.

Currently, the control at the component levels is typically executed by PID controllers (or controllers based on PID) [19]. As the lower level components have few variables to control and typically respond reasonably fast, these simple controllers are sufficient to control the components in close to optimal conditions [19].

As previously mentioned, the system level controller has to provide desired values for temperature, pressure, and other variables (which are called *setpoints*) and has to decide whether to switch equipment on or off. A PID controller is not suited for this application and thus, rule-based controllers are typically used at this level [20, 21, 22]. Rule-based controllers (or in other words a Sequence of Operations (SOO)) are heuristic controllers based on previous knowledge and first principle physics [20]. However, a HVAC system is a very complex system. There are a lot of different setpoints in such a system and thus many different combinations of setpoints are possible. Moreover, the optimal operating condition of the HVAC in a DC can shift due to changing outdoor temperatures, radiation or a varying ITE load. Finally, the system itself is subject to change due to deteriorating equipment conditions or varying ITE occupation [18]. Combining all these makes it very complex to create a SOO which operates at (close to) optimal efficiency in all possible conditions [20, 23, 24, 25].

1.2 HVAC optimization techniques

As the current control of HVAC systems is often not optimal, a lot of research is performed to improve the energy efficiency of its controllers. In this section, the state-of-the-art optimization techniques used to create optimal controllers for DCs are discussed. The main approaches are:

a two-stage optimization approach, where first a model is created of which subsequently a set of optimized control rules is derived; Model Predictive Control (MPC) which uses a model of the system to predict the optimal control inputs of the system; or RL which learns to control an unknown system by interacting with it repeatedly.

1.2.1 Two-stage optimization

In most current applications for the optimization of the higher-level controllers, a model of a HVAC system is developed first and then this model is optimized. In [6, 7, 8, 9, 10, 11] different versions of this approach are used to optimize the chiller plant of a building. All of them use thermodynamics-based models to model the chiller system. The optimization techniques, however, vary between papers. For example, in [7], a hierarchical optimal controller is applied, consisting of component-level controllers and a supervisory higher-level controller. While in [6], a genetic algorithm has been used to find the optimal operating point of a chilled water system. In [15], this approach is used for the optimization of a DC. In this paper, the airflow is modeled using a Proper Orthogonal Decomposition (POD) which is then in turn optimized.

However, as stated in [26], this two-stage approach of first creating a model and then optimizing the model has some important drawbacks. One drawback is that the models created are specific to the system. Therefore, optimizing a different system necessitates creating a new model. Given the time-consuming nature of model creation, this limitation presents a significant challenge. Another drawback is the complexity of the HVAC system in DCs, which makes it very difficult to create an accurate model [27]. Consequently, a controller that has been optimized on a model does not necessarily operate optimally on a real system. This is especially the case when the model parameters shift due to for example deteriorating equipment, varying ITE load, or even global warming when looking at larger timescales.

1.2.2 Model Predictive Control

In recent years, MPC [28, 29, 30] has emerged as a popular optimization technique for the control of DC HVAC [12, 13, 31]. The MPC framework, shown in Figure 1.1, involves a prediction model and an optimizer working together to determine and implement actions that minimize a specific loss function, such as reducing HVAC power usage. Feedback from the environment is used to continuously update and refine the actions taken. In [31], MPC combined with RL is implemented on the HVAC of a DC and shows improvements over the PID controllers. In [12], an adaptive MPC is proposed for a cooling unit in a DC and in [13], an MPC is proposed that both manages the cooling system and the ITE workload of a data center. All three papers show promise of 10% of power savings when compared to traditional methods.

The application of MPC for DCs is reviewed in [32]. As discussed in the paper, there is a range of modelling techniques that can be applied in the MPC framework. The modelling methods can be divided into 2 categories: physics-based and data-based models. As discussed before, it is very hard to model complex processes using physics-based models. However, they do have good interpretability. On the other hand, the data-based models can handle complex processes. Their disadvantages are the problem of overfitting on existing data and that they can not guarantee accurate predictions for situations that do not occur in the data the models are based on.

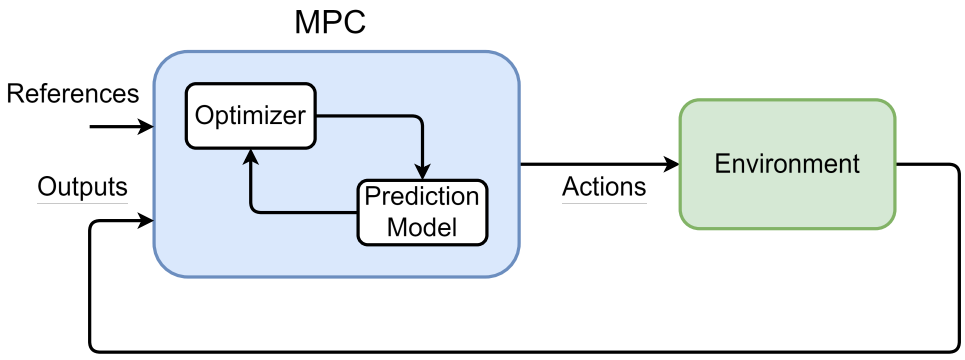


Figure 1.1: The Model Predictive Control framework

1.2.3 Reinforcement Learning

RL is a Machine Learning (ML) technique that learns to control a system by interacting with it repeatedly. Because of this interaction, RL is capable of learning optimal control policies for complex systems without requiring a predefined model [33, 34, 35]. The way RL works is by sending control actions to a system, or environment, and it receives a so-called reward and information on how the system is affected back. Through many interactions, the RL algorithm can learn how to maximize this reward. Figure 1.2 displays this framework. RL seems a good

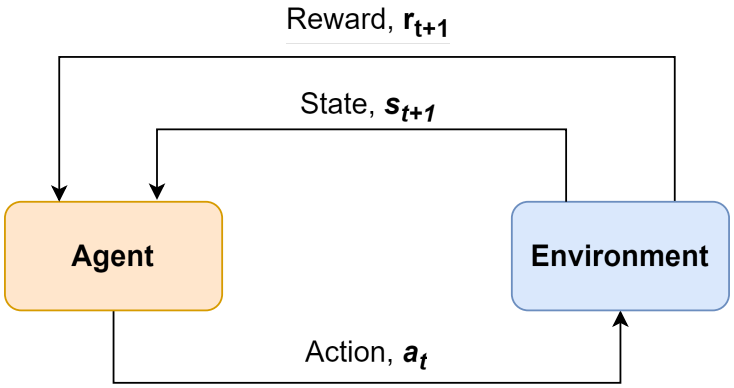


Figure 1.2: The RL framework

fit to control HVAC systems because of the nature of RL, which does not require a model, and can learn to control complex systems. This is a good fit with the complex, hard-to-model nature of HVAC systems. Moreover, the power consumption of a DC serves as a natural reward function within the RL framework.

The use of RL in DC HVAC systems has been reviewed in [18, 23, 32]. These reviews conclude that RL is a promising technique to optimize the control in DCs HVAC systems. However, they do address several challenges in RL. Firstly, [18] concludes that nearly all RL-based optimization is still performed on models, instead of actual DCs. Secondly, RL has low sample efficiency, which means that the algorithms are computationally inefficient [23] and may take a long time to converge on slow, real-life DCs [32]. [32] also concludes that RL algorithms often struggle with maintaining temperature constraints, which is particularly critical in DCs due to their mission-critical nature, making any violation of these constraints highly catastrophic.

In recent literature, the findings of these review papers are supported. Initially, research demonstrated the feasibility of using RL to optimize HVAC systems through simplified models. Studies such as [27] and [24] showcased energy efficiency improvements using Deep Reinforcement Learning (DRL) agents to control air and chilled water flow in thermodynamics-based building models. Similarly, [26] achieved up to 15% energy savings by optimizing the controllers of HVAC systems using RL based on data traces from actual data centers.

After this initial research proved that the concept of using RL for the optimization of HVAC controllers in both normal buildings and DCs, a lot of research has been performed to improve the performance of the RL agents. These improvements are on the level of energy efficiency, reliability, increasing the sample efficiency, and closing the gap between simulation and reality.

The following literature presents advancements in using RL to improve the energy efficiency of DC HVAC controllers. In [36], an offline, model-based DRL agent for the HVAC system in buildings is proposed, which can train on interactions a model it learned from the environment, while fine-tuning itself using interactions with the environment. This greatly increases the sample efficiency. In [37], principles of Safe Reinforcement Learning (SRL) are combined with batch learning, to both decrease the number of constraint violations and increase the sample efficiency of a DRL agent for the HVAC system in a commercial building. [38] uses a Recurrent Neural Network (RNN) to learn patterns in the behaviour of an Air Handling Unit (AHU) and as such improve the efficiency of the original controller by up to 30%. Although all of this research proposes novel methods, they are all evaluated on either thermodynamic models or data traces. In [20], Google DeepMind applies an RL algorithm to a real building, achieving 9-13% energy savings in real-world tests. The agent does this by learning to use cooler condenser water for more efficient chiller plant operation and learning to compensate for temperature sensor miscalibrations.

In the specific case of DCs, [25] pre-trains its RL agents on thermodynamics-based models, and then tests them on more realistic Computational Fluid Dynamics (CFD) based models, demonstrating that pre-trained agents can adapt to a more realistic DC environment relatively fast. This is a step in bridging the simulation-to-reality gap. [39], applies a physics-guided RL agent, where exploration is bound by a physical model. Finally, in [40], a RL agent is used to manage both the server allocation and HVAC system.

It should be noted that all research on DCs has been applied to models, where either only the mass flow or a single setpoint temperature is adjusted. Next to that, these models all control just a small number of variables, such as only the room temperature setpoint in [25] or only mass-flows in [40].

1.3 Data center modelling

As DCs have a mission-critical infrastructure, models of the HVAC system are required for research of new types of controllers. The previous section already briefly mentioned a number of them, and this section further discusses three state-of-the-art DC HVAC modeling techniques: thermodynamics-based models, CFD models, and data-based models.

1.3.1 Thermodynamics based models

The first simulation method to be discussed are the thermodynamics-based models. As the name suggests, thermodynamics-based models use the principles of thermodynamics as the mathematical base of a simulation. These models can approximate the energy usage of a system over longer time periods. They are also able to model all the components in a HVAC system. Another advantage is their low computational complexity since they allow for relatively large timesteps (1-60 minutes) and are based on first-order physics [41, 42]. Since thermodynamics-based models are based on physical relations, they can accurately model a wide range of settings of the HVAC system. The largest drawback of these models is that they are based on first-order physics, and are thus not able to accurately simulate details, such as the airflow inside a data center room.

A widely used approach for simulating BES with this type of model is using the EnergyPlus [42] software. EnergyPlus offers a modular approach for the simulation of buildings. Both building geometry (like the size of rooms or the insulation of walls and windows) and a complete HVAC system can be modeled in EnergyPlus. This leads to great flexibility in the type of building that can be modeled using this software. However, EnergyPlus suffers from the drawbacks associated with thermodynamics models. In the case of a data center, one of the largest drawbacks is that EnergyPlus models a room as a thermal zone and does not allow for temperature differences inside a room [42, 43]. When recalling Figure 2.2, it is clear that this causes a problem in a data center room, where large temperature differences occur. In [44], a solution for this problem is proposed. A data center room is simulated using both EnergyPlus and CFD, which offers a lot more accurate modelling of the airflow in a room. Then the EnergyPlus model was calibrated such that the temperatures entering and leaving the Computer Room Air Handler (CRAH) match the CFD simulation. Another drawback of EnergyPlus is that it was originally meant for the simulation of buildings over the course of a year. To solve this problem, custom solutions have been proposed to make it suitable for RL, such as a testbed to connect the EnergyPlus simulation to python, making it possible to adjust actions every time-step [45] and a python implementation of the EnergyPlus code [46].

1.3.2 Computational Fluid Dynamics

The second simulation method that is used for HVAC modelling is CFD. It is the branch of fluid mechanics that solves the behaviour of fluids by numerical analysis. CFD predicts this behaviour by solving a set of underlying equations. In most tools, these underlying equations are the Navier-Stokes equations [47]. CFD methods allow for very accurate simulation of the air flows in a data center. Comparable to the thermodynamics-based models, these methods can accurately model a wide range of situations in a room, since they are based on physics. However, CFD methods do require a fine grid and small timesteps [48], which gives these methods a high computational complexity. Due to this complexity, it is seen that in studies, these methods have not yet been used for the training of agents, although there are studies that test their agent for a few simulation days on a CFD model [25]. Another drawback is that it is very hard to model a HVAC system and room simultaneously using CFD, due to the complex heat transfer mechanisms between these 2 systems. Therefore, simultaneous simulation of these 2 systems is not found in the literature.

1.3.3 Data-based models

The third model type used for modelling DC HVAC systems are the data-based models. These data-based or data-driven methods uses data which is collected from use with a typical higher-level HVAC controller. Using this data, relationships are found using techniques like regression or Neural Network (NN) [41]. Among the data-driven models are: fuzzy logic models, statistical models, state-space models, stochastic models, and many more. These models have the large advantage that they are based on the real data of an HVAC system and thus have little to no assumptions on the behaviour of the model. This means that they can predict the states of a HVAC system more accurately than most physics-based methods under the circumstances the models are trained on [41]. However, the large disadvantage is that they are not based on any underlying physics, and thus are not able to approximate the behaviour of a HVAC system under circumstances that differ from the circumstances the model was trained on. For example, in the case of a DC, the setpoints are rarely adjusted in reality. When a model is trained on real data, it can accurately predict the behaviour of the HVAC system in this small operating range. If the setpoints are then changed during the training of a new type of optimal controller, the model has to extrapolate from the generated data, but there is no way of quantifying whether this is what would happen to the system in reality.

1.3.4 Combinations of methods

These are the three types of models typically used for developing HVACs controllers, often in combination. For instance, CFD models were used to calibrate thermodynamic models in [44]. Grey box methods, which blend data-driven and physics-based approaches, are another example. In these methods, a physics-based model forms the underlying structure, with unknown parameters estimated from real-world data. This combination offers high accuracy and can predict behavior in scenarios different from the measured data.

1.4 Research goal

In the previous sections, it was seen that the HVAC system in DCs uses lots of energy. These systems are currently often controlled by rule-based controllers and the research covered suggests that improvements in energy efficiency can be made by optimizing these controllers for energy efficiency [5].

Section 1.2 covers research on different techniques that are used for the optimization of these controllers. Two-stage optimization techniques have the drawback that they require a model of the system on which the energy-optimized controller then is designed. The HVAC system of DCs proves to be very difficult to model, and these models do not adapt to changes in the system over time. Therefore, the two-stage solution often does not lead to optimal controllers. Next to that, the creation of a model is specific for individual DCs, and thus this approach is not scalable to a large number of DCs.

Another often-researched approach is MPC. Although MPC still requires a model of the DC, its model is adaptable to changes in the system. Thus it does show improvements in the energy efficiency of the controller when compared to the two-stage approach. However, the creation of a model for MPC is still labour intensive and has a chance of having a mismatch with the actual system [49].

The one-stage approach of RL mitigates these problems, as it does not require the creation of a model. Therefore, this technology shows a lot of potential, as the lack of a model reduces the risk of bias and increases the possibility to scale to more DCs without the labour associated with model creation. Thus, this thesis focuses on using RL to develop an energy-efficient HVAC control system for DCs.

From the research on RL follows that there are still several challenges that should be addressed before RL agents could be implemented as the controller of DC HVAC systems. These challenges include improving the sample efficiency of the agent or improving the handling of constraints in the system. Next to that, most research focuses on only a single type of control variable, such as only mass flows or temperature set points. Finally, the models used to train RL agents are often heavily simplified when compared to the real HVAC systems [25, 36, 45]. As not all problems related to RL in DCs can be addressed in a single thesis, the focus will be on a subset of these issues. Given that DC environments are mission-critical, and the reliable operation of these systems is extremely important, this thesis primarily concentrates on handling the constraints within a DC's HVAC system. Additionally, the control variables will include both mass flow and temperature setpoints, as it is believed that the true strength of RL lies in optimizing the interaction between these variables to achieve the best performance.

In the existing research, often only the final reward function is presented, without further investigation into the effects of modifying this reward function. This thesis creates a controller that minimizes the HVAC power of a DC using RL and then identifies the effects of changes in the reward function on both the power usage and the ability of the controller to handle temperature constraints.

This thesis addresses this issue by first defining a HVAC system on which the controller is designed. This is done in chapter 2, which first gives the required background on HVAC systems in DC, then defines the exact HVAC system which is used in the remainder of the thesis. Finally, an optimization problem is formulated, including all constraints of the system.

The RL agents are trained and tested on a simulation model of the HVAC system. The performance of the controller is compared to a conventional controller which serves as a baseline for a fair analysis of the performance of the RL controller. How this is done is described in chapter 3. First, the creation of the simulation model is described in section 2.3. Then, the working of the baseline controller is discussed in section 3.1. Following that, section 3.2 gives a short background on RL, followed by the implementation of RL on the HVAC system. This includes the formulation of the HVAC system in a suitable framework for RL, the selection of an algorithm, and the creation of a reward function. Finally, section 3.4 describes how the RL agent's hyperparameters are tuned and how different reward settings are investigated.

Following all this, chapter 4 gives the results of the experiments outlined in chapter 3, and subsequently compares the performance and control strategies of the tuned and trained RL-based controllers to that of the baseline controller.

Finally, a conclusion and discussion on the results of this thesis are presented in chapter 5.

Chapter 2

Problem statement

This thesis will design and implement an RL-based controller for a DC's HVAC system, to optimize the energy efficiency of this system while maintaining the safe operation of the system. The reasoning behind this is that the HVAC systems in DCs use large amounts of energy and RL shows promise to improve upon the energy efficiency of conventional controllers. This chapter defines the system on which this controller will be implemented and defines a formal goal of the controller.

First, section 2.1 gives required background knowledge of HVAC systems in a DC. Then, section 2.2 provides the exact design of the system. This includes the layout of the system; a description of the energy balance in the system and the power used by the individual HVAC components and finally a description of the variables that can control the system and variables that disturb the system. Consequently, in section 2.3, a simulation model of this system is set up. This will be used to train and assess the performance of the controllers used in this study. Finally, a minimization problem is formalized for this system, which serves as the performance objective of the RL based controller.

2.1 HVAC systems in Data Centers

A DC consists of large rooms filled with servers. The HVAC system of a DC ensures the right operating conditions for this ITE, such as desired temperatures, air mass flow rates, and air humidity.

In this section, the HVAC system in a DC is described. To narrow the scope of the research, the focus is on one of the most common cooling layouts: a chiller plant. Such a layout is shown schematically in Figure 2.1. The cooling loads of this picture are caused by the servers of the DC in a server room. Section 2.1.1 discusses how such a room itself is cooled. Then, in subsection 2.1.2, it is discussed how the heat is extracted from the room using the rest of the chiller plant.

Mission-critical system A very important aspect of DCs is that they are mission-critical. In other words, when the servers in a DC malfunction, this has severe consequences for the companies that host their data on these servers [51]. Since the operation of these servers is mission-critical, also the HVAC system is mission-critical. If it fails, the servers might overheat and be damaged [52]. Even violating certain temperature constraints can already damage the servers.

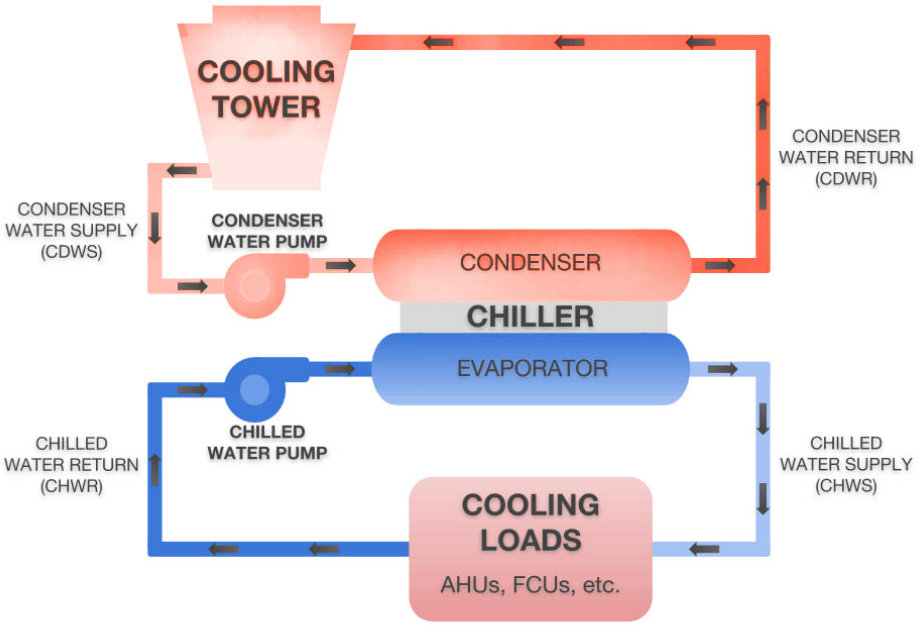


Figure 2.1: A schematic overview of a chiller plant [50].

Therefore, the HVAC system should always meet these constraints.

This mission-criticality is one of the main goals in the design of the HVAC systems of a DC and is thus also a fundamental part of the problem statement. Next to the constraints formulated in this chapter, it is important to realise that this section provides a simplified version of the HVAC systems in DCs and that in reality a lot of additional measures are implemented to ensure this mission-criticality of the system.

2.1.1 Cooling layout in a server room

To better understand the specific cooling layout employed in a server room, it is important to first have a better understanding of its central component: the server rack. Server racks contain ITE, such as servers or networking equipment [53]. The racks are specifically designed to be easily accessible on all sides, allowing for easy cable management and good airflow. The goal of the HVAC system is to ensure that none of the equipment inside these racks overheats.

Figure 2.2 displays the air flow in the room. In summary, the server racks in a DC room are cooled using one or multiple CRAHs. These systems cool down air using a Cooling Coil (CC) which contains chilled water. Using a fan, cold air is provided below a raised floor, as displayed in Figure 2.2. Through perforated tiles, the cold air enters the server room, where it is passed through the server racks to cool the servers. The hot air is then directed back to the CRAH so it can be cooled again.

Hot and cold aisles The layout of server racks significantly impacts HVAC system efficiency due to its effect on airflow optimization and IT equipment cooling. The hot and cold aisle lay-

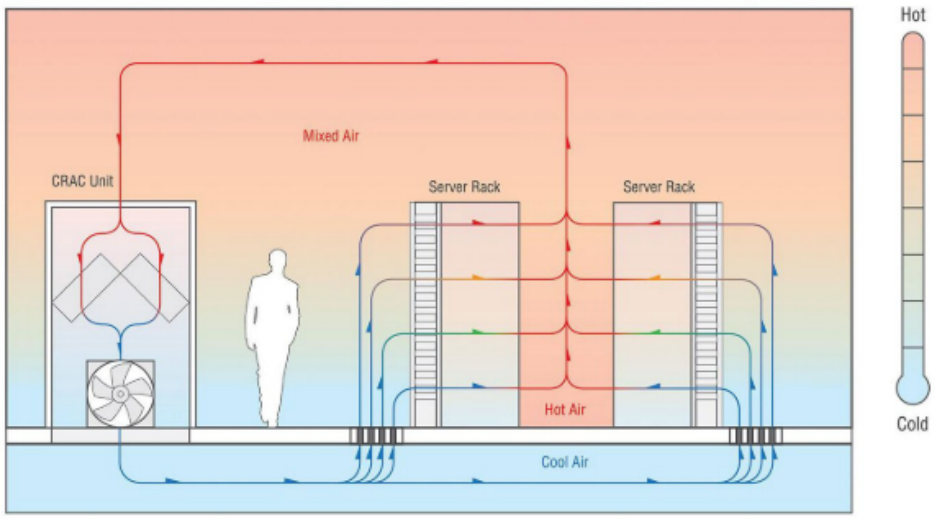


Figure 2.2: A schematic representation of the airflow in a server room [44]

out, introduced by IBM in 1992 [54], is widely adopted. It alternates rows between cold aisles supplied with cold air and hot aisles where hot air is expelled. This separation minimizes air mixing, enhancing cooling efficiency. Adopting this layout can reduce HVAC energy consumption by 20-25% [55]. Figure 2.3 illustrates the clear separation of cold and hot air streams.

Since mixing of air is still possible using this layout, both hot and cold aisle containment strategies have been employed recently. An in-depth comparison of these strategies is given in Appendix A.1. For this thesis, containment of the hot aisles, as displayed in Figure 2.4 is employed, as it is the most efficient option of the two.

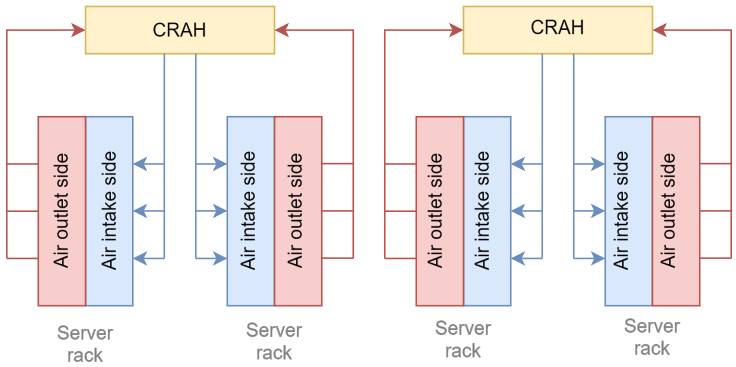


Figure 2.3: A server room with a hot and cold aisle layout.

2.1.2 Chiller plant

Now that it is known how the servers are cooled inside the room, a closer look can be taken at the chiller plant, which actually cools the air. As has been shown in Figure 2.1, a chiller plant consists of 2 water loops. The first one is the chilled water loop, which cools down the air in the room using a CC and a water chiller. The chiller uses a refrigeration cycle to cool down the water to a desired temperature. The other loop is the condenser water loop. This loop extracts

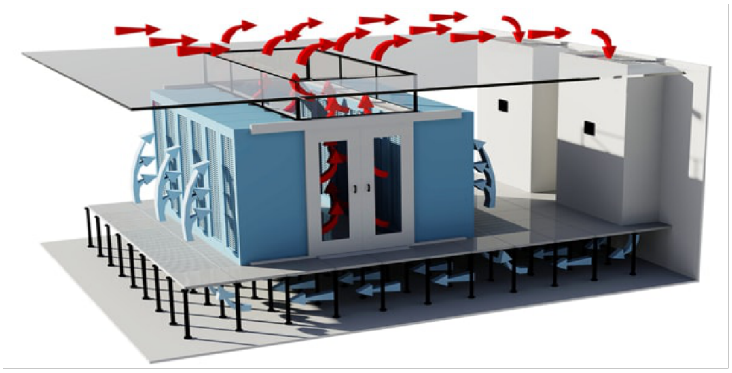


Figure 2.4: Schematic representation of hot aisle containment [56].

energy from the chiller through a heat exchanger and rejects this energy to the outdoor air by a Cooling Tower (CT), which is a water-to-air heat exchanger. More details about these components can be found in Appendix A.2.

It should be noted that, although Figure 2.1 only displays a single chiller, CRAH and cooling tower, there can actually be multiple of each of these components connected to the loops in a real DC. However, for comprehensibility, this explanation assumes that there is only one of each component, as in Figure 2.1. To get a more in-depth understanding of what happens in these loops, each component and its functions are analyzed separately.

2.2 HVAC system design

Using the background knowledge from section 2.1, a simple HVAC system has been defined which is optimized in the remainder of this thesis. The designed system is a simple chiller plant containing 1 CRAH, 1 chiller, and 1 cooling tower. This plant is used to cool a single DC room. While real DCs typically employ more complex HVAC systems, this simplified layout has been chosen for this thesis to facilitate the isolation of the effects of specific components. This simplicity allows for a clearer analysis of these individual factors. Next to that, there is also a practical reason for a simpler system. During initial investigations, it was found that the implementation of more complex HVAC systems in most simulation software requires a lot of non-standard HVAC components and customization of the software. Because of the expectation of clearer analysis and to keep the focus of this thesis on RL instead of the modeling of a DC, the simple layout is chosen.

The layout of the HVAC system is shown in Figure 2.5. The system consists of the following 3 loops which all have a specific purpose:

- **Air Loop (AL):** This is the loop that cools the servers. It consists of a CRAH and the server room. The CRAH cools the hot air that is coming into the CRAH using a cooling coil. Then, the humidifier ensures that the correct humidity is maintained in the room. Finally, the CRAH fan blows the cooled and humidified air under the raised floor. In the server room, the air cools the servers using a cold air containment layout.
- **Chilled Water Loop (ChL):** This loop has the goal of extracting heat from the air loop and transporting it to the chiller. The chiller cools the water in the loop to a desired chilled

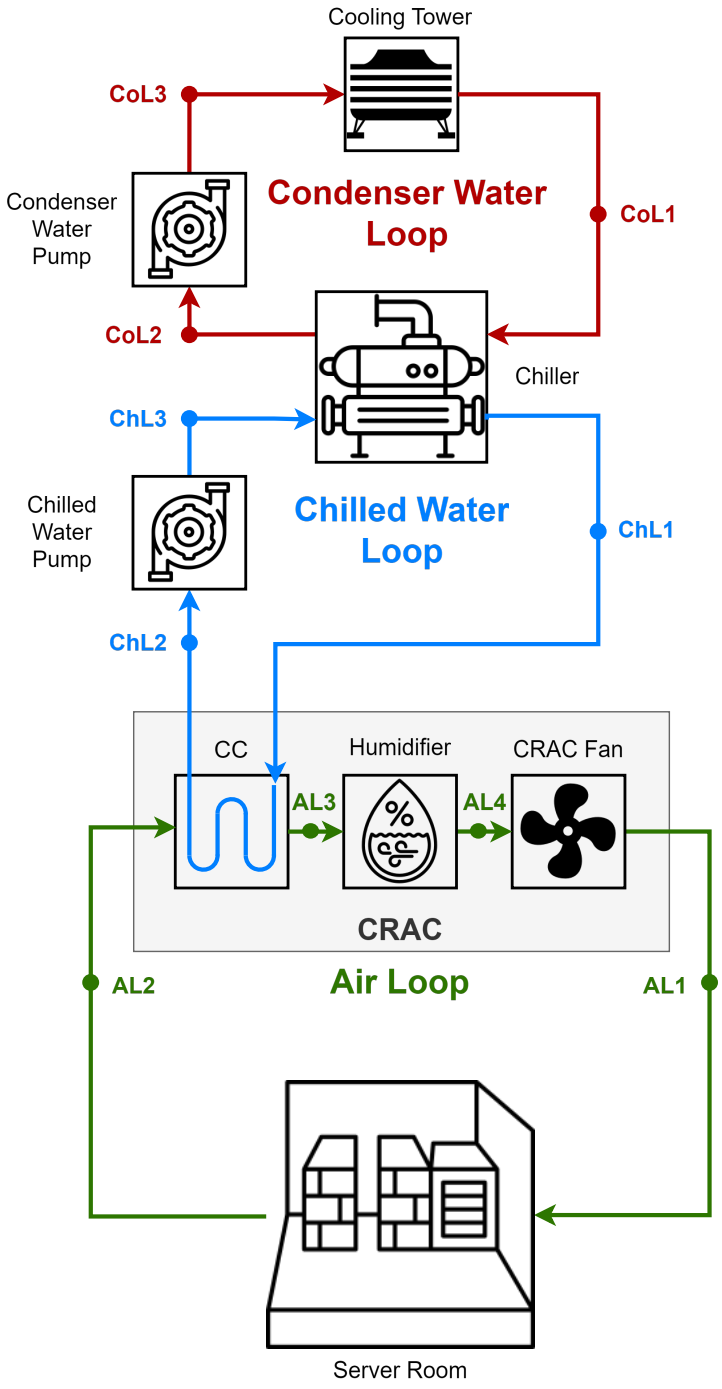


Figure 2.5: A schematic representation of the HVAC system used in this thesis

water temperature. Then, the chilled water extracts the heat from the air loop by using the cooling coil. This water, which has heated up by now, is then pumped back to the chiller by the chilled water pump.

- **Condenser Water Loop (CoL):** This loop cools the chilled water by extracting its heat and then rejects this heat to the environment using a cooling tower. The chiller uses a refrigeration cycle to extract the heat from the chilled water and transfer it to the condenser

water (which is possibly warmer than the chilled water). This water is pumped to the cooling tower where the heat is rejected to the environment.

The following naming convention of variables within the loop has been used: T^{x} is the temperature at location x. For example, the temperature leaving the CRAH is defined as T^{AL1}. Likewise, p^{x} represents pressure, $\dot{m}^{x}$ represents mass flow and ϕ^{x} represents relative humidity, all at location x. Additionally, P^{x} is the power of component x, for example, P^{CT} is the power of the cooling tower.

2.2.1 Energy balance and power

The variables are all related to each other through the energy balance equations in the system, of which the most important ones are discussed here. The best way to understand these relations is by starting from the server room and following the flow of the energy throughout the system to the outside air.

Energy in the ChL The energy balance in the server room is given by:

$$\dot{Q}^{ITE} = \dot{m}^{AL} c^{a} \left(T^{AL2} - T^{AL1} \right) \tag{2.1}$$

where $\dot{Q}^{ITE}$ is the heat transfer of the ITE, which is assumed to be equal to P^{ITE}, meaning that all energy is converted to heat. c^{a} is the heat transfer coefficient of air. From this equation follows that the air is warmed up by the ITE. This added heat is then removed from the room by the CC. Here, the heat transfer is defined as:

$$\dot{Q}^{AL,\,CC} = -\eta^{CC} \dot{Q}^{ChL,\,CC} \tag{2.2}$$

in this equation:

- $\dot{Q}^{AL,\,CC}$ is the heat transfer rate of the air in the CC, defined as:

$$\dot{Q}^{AL,\,CC} = \dot{m}^{AL} c^{a} \left(T^{AL3} - T^{AL2} \right) \tag{2.3}$$

- $\dot{Q}^{ChL,\,CC}$ is the heat transfer rate of the chilled water in the CC, which is defined as:

$$\dot{Q}^{ChL,\,CC} = \dot{m}^{ChL} c^{w} \left(T^{ChL2} - T^{ChL1} \right) \tag{2.4}$$

- η^{CC} is the efficiency of the CC.

After the CC, the air is moved by the fan, which is the only power-consuming HVAC component in the AL. The most important relation between the fan and the mass flow is that it is cubic proportional to the mass flow:

$$P^{AL,\,Fan} \propto \left(\frac{\dot{m}^{AL}}{\rho} \right)^{3} \tag{2.5}$$

Under the assumption that the density does not change much in a DC setting, the fan power is thus proportional to its mass flow rate. As this concludes the behaviour of the AL, we can now continue to the energy balance inside the ChL.

Energy balance in the ChL Firstly, $\dot{Q}^{\text{ChL, CC}}$ is related to the AL by Equation (2.1) and its relation is already given in Equation (2.4). It should be noted that a condition for the working of the CC is: $T^{\text{ChL2}} > T^{\text{AL2}}$ and $T^{\text{ChL1}} > T^{\text{AL3}}$. After the CC, the water is pumped towards the chiller by the ChL pump. The power in this pump is related to the mass flow rate by:

$$P^{\text{ChL, Pump}} = \frac{\Delta p \dot{m}^{\text{ChL}}}{\rho^{\text{w}} \eta^{\text{Pump}}} \tag{2.6}$$

where Δp is the pressure difference before and after the pump, ρ^{w} is the density of water and η^{Pump} is the efficiency of the pump. After this pump, the water is cooled down by the chiller. This is again given by a heat balance:

$$\dot{Q}^{\text{ChL, Chiller}} = -\eta^{\text{Chiller}} \dot{Q}^{\text{CoL, Chiller}} \tag{2.7}$$

where:

- $\eta^{\text{Chiller}}\left(T^{\text{ChL1}}, T^{\text{CoL1]}}\right)$ is the efficiency of the chiller, which is highly dependent on the temperature of the water entering and leaving it.
- $\dot{Q}^{\text{ChL, Chiller}}$ is the heat transfer rate from the chilled water to the chiller. This is defined as:

$$\dot{Q}^{\text{ChL, Chiller}} = \dot{m}^{\text{ChL}} c^{\text{w}} \left(T^{\text{Chl3}} - T^{\text{ChL1}}\right) \tag{2.8}$$

- $\dot{Q}^{\text{CoL, Chiller}}$ is the heat transfer rate between the condenser water and the chiller. It is defined as:

$$\dot{Q}^{\text{CoL, Chiller}} = \dot{m}^{\text{CoL}} c^{\text{w}} \left(T^{\text{CoL1}} - T^{\text{CoL2]}}\right) \tag{2.9}$$

Important to note here is that, for the chiller, there is no condition regarding the temperatures, as it can actively cool the system down by its refrigeration cycle.

The power of a chiller depends on the Coefficient of Performance (COP) and cooling rate:

$$P^{\text{Chiller}} = \frac{\dot{Q}^{\text{ChL, Chiller}}}{COP} \tag{2.10}$$

As the COP is a function of its inlet and outlet temperatures, and this function is different for each specific chiller, this power cannot be determined more specifically.

Energy balance in the CoL As the chiller concludes the ChL, the behaviour of the CoL can now be discussed. How the chiller heats the water in the CoL has already been discussed. After the chiller, the pump controls the mass flow in this loop. It works analogously to the ChL pump. Finally, the cooling tower has an energy balance with the outdoor air.

The cooling tower has a single-speed fan, meaning the fan can only run at one speed or be turned off. This makes the energy balance slightly different, as only 2 mass flows are possible, and a cycle ratio is introduced: r^{CT} which indicates the percentage of time the fan is turned on over a timespan. This leads to the following energy balance:

$$\dot{Q}^{\text{CoL, CT}} = -\eta^{\text{CT}} \dot{Q}^{\text{out, CT}} \tag{2.11}$$

in this equation:

- $\dot{Q}^{\text{CoL, CT}}$, is the heat transfer from the water in the CoL to the outdoor air, defined as:

$$\dot{Q}^{\text{CoL, CT}} = \dot{m}^{\text{CoL}} c^{w} \left(T^{\text{CoL1}} - T^{\text{CoL3}} \right) \tag{2.12}$$

- η^{CT} is the efficiency of the cooling tower, which is again highly dependent on the specific cooling tower and requires modeling to accurately predict.
- $\dot{Q}^{\text{out, CT}}$, is the heat transfer rate from CT to the outside air. Here, the cycle ratio is implemented for the outdoor air mass flow. It is defined as:

$$\dot{Q}^{\text{out, CT}} = \dot{r}^{\text{CT}} m_{max}^{\text{CT Fan}} \left(T_{\text{out}}^{\text{outdoor}} - T_{\text{in}}^{\text{outdoor}} \right) \tag{2.13}$$

Regarding the power, the CT is based on the fan power, and thus is defined as:

$$P^{\text{CT}} = r^{\text{CT}} P_{max}^{\text{CT, Fan}} \tag{2.14}$$

where $P_{max}^{\text{CT, Fan}}$ is the power used by the fan inside of the CT when it is running at full speed.

These equations summarize the energy balance and power used by the system. They show how the different variables interact with each other, and already give an idea of how the components use their power.

2.2.2 Actuation variables

This thesis focuses on the higher-level controllers in the HVAC system, as the component level controllers often already work close to optimal. The higher-level controller controls the settings of all the components in the system and is generally rule-based. The higher-level behaviour of this system is controlled using a number of setpoints, or desired values, which are defined by the higher-level controller. The lower-level controllers then control the components to meet these setpoints. The setpoints in the system are:

- $T_{\text{SP}}^{\text{CT}}$, the leaving CT temperature setpoint. This setpoint is defined as:

$$T_{\text{SP}}^{\text{CT}} = T^{\text{outdoor}} - T^{\text{CoL1}} \tag{2.15}$$

  where T^{outdoor} is the outdoor temperature.

- $\dot{m}_{\text{SP}}^{\text{CoL}}$, the mass flow setpoint of the CoL.
- $T_{\text{SP}}^{\text{ChL1}}$, the temperature setpoint of the chilled water which is leaving the chiller.
- $\dot{m}_{\text{SP}}^{\text{ChL}}$, the mass flow setpoint of the ChL.
- $\dot{m}_{\text{SP}}^{\text{AL}}$, the mass flow setpoint of the AL.
- $\phi_{\text{SP}}^{\text{AL4}}$, the setpoint of the humidity of the air leaving the humidifier.

2 disturbance variables affect the system and can not be controlled. These variables cause the system to be of a stochastic nature, as they are influenced by external factors beyond the system's control. This leads to random variations in the system. This requires the use of probabilistic methods in system modelling and control. The disturbance variables are:

- T^{outdoor}, the outdoor temperature. This affects how easily the water in the CoL can be cooled.
- P^{ITE}, the ITE load. This load determines how much energy the system has to cool.

All other variables are dependent on these 8 variables. This includes *actual* temperatures, mass flows, and humidities, as well as the pressure at every point in the system and the power of the HVAC components.

2.3 HVAC system simulation model

In this section, a simulation model of the HVAC system designed in section 2.2 is created. This model is used for both the baseline and RL experiments. First, a simulation software is selected. In this software, the model is implemented. This implementation starts with the definition of a server room model and expands it with a complete HVAC system. After this, suitable algorithms are selected to calculate the energy and heat balances within the system. Finally, the setup of a timestep study is proposed, to test the robustness of the simulation model to its timestep.

There are 3 different types of HVAC simulation that are commonly used: thermodynamics-based models, CFD based models, or data-based models. For the training and testing of RL agents, a model must have a good computational complexity. Next to that, the model should be able to accurately simulate situations that do not happen often when using conventional controllers. Therefore, thermodynamics-based models are chosen, as they have low computational complexity, are based on physical relations, and DC HVAC models are readily available. On the other hand, CFD simulations have high computational complexity, and data-based models are not physics-based, and thus are bad at extrapolating. The software used in this thesis is EnergyPlus, as it is open-source and has off-the-shelf models for DC rooms.

Drawbacks of EnergyPlus A large drawback of thermodynamics-based simulation is that most models do not take any system delays into account. This includes delays due to the length of pipes, or the start-up time of the equipment. This leads to a large limitation in the simulation of the system when compared to a real system, and has an effect on the controller for sure since handling system delays is a major challenge in control engineering. However, this simplification is accepted as solving it would cost a lot of time, and would shift the focus of this thesis from creating a new controller to creating an accurate HVAC model of the system.

2.3.1 EnergyPlus HVAC layout

In the simulation model, the server room model in EnergyPlus is based on the design from [44]. This paper features both large and small DC rooms with hot and cold aisle containment. The study uses CFD to determine temperature offsets for server inlet and outlet temperatures (T^{AL1} and T^{AL2}) at several values of P^{ITE} and $\dot{m}^{\text{AL}}$, which are then incorporated into EnergyPlus through a lookup table, with linear interpolation used for other operating conditions. This effectively handles the large drawback of EnergyPlus, which is that it is not able to model the movement of the air inside the room. The small DC room is selected, as also a simple HVAC system is used in this thesis. Hot aisle containment is implemented, as this is the default layout for modern DCs.

The original model uses a very simplified HVAC system which calculates the dynamics based on the outdoor and chiller inlet temperatures. This is not suitable for the RL controller, as it lacks influence over variables like $\dot{m}^{\mathrm{ChL}}$ and setpoint temperatures inside the system.

To overcome these drawbacks, the room model from [44] is expanded with a HVAC system which matches the designed HVAC system. The different loops in Figure 2.5 are rebuilt in EnergyPlus using the EnergyPlus HVAC components listed in Table 2.1. The CT is of the `SingleSpeed` type, which means that its fan can be switched on or off, but its speed can not be controlled. This component type has been chosen as this is what is often seen in real DCs. For the pumps and fan, `VariableSpeed` type components were chosen, to be able to control the mass flow rates in the loops. The chiller type is `ElectricEIR`. This is a model of an electric chiller whose power calculations are based on the Energy Input Ratio (EIR), which is the ratio between the energy input and cooling output of the chiller. This is a versatile model that provides accurate simulation results in differing settings [57]. Finally, the CC, humidifier, and CRAH diffuser models are all the standard choices for these types of components in EnergyPlus.

For all of these models, except for the chiller model, the standard settings have been used. These settings determine the behaviour of a component such as its power usage and temperature increase as a non-dimensional output. Then, using reference factors that are autosized by EnergyPlus, these non-dimensional values are converted to actual values using reference variables. The autosizing is done to ensure the HVAC components will meet the requirements of the system.

Table 2.1: Overview of the EnergyPlus model types in the simulation model

HVAC component	EnergyPlus component model
Cooling Tower	`CoolingTower:SingleSpeed`
Condenser Water Loop pump	`Pump:VariableSpeed`
Chiller	`Chiller:ElectricEIR`
Chilled Water Loop pump	`Pump:VariableSpeed`
Cooling Coil	`Coil:Cooling:Water`
Humidifier	`Humidifier:Steam:Electric`
Air Loop fan	`Fan:VariableVolume`
CRAH diffuser	`AirTerminal:SingleDuct:VAV:NoReheat`

Chiller model While standard EnergyPlus models are suitable for most components, the chiller models available in the EnergyPlus database do not meet the temperature requirements necessary for DC applications. Typical chilled water temperatures in DCs range from 18 to 24°C, which is significantly higher than the maximum specified chilled water temperature of 12°C in the EnergyPlus database. Therefore, a chiller model has been modified to meet the situation in a DC. In EnergyPlus, the chiller model is based on 3 (bi)quadratic functions [57]:

- The cooling capacity $\left(\dot{Q}\right)$ as a function of T^{CoL1} and T^{ChL1}:

$$\begin{aligned}\dot{Q}\left(T^{\mathrm{CoL1}}, T^{\mathrm{ChL1}}\right) =& a_1 + a_2 T^{\mathrm{CoL1}} + a_3 \left(T^{\mathrm{CoL1}}\right)^2 + \dots \\ & a_4 T^{\mathrm{ChL1}} + a_5 \left(T^{\mathrm{ChL1}}\right)^2 + a_6 T^{\mathrm{CoL1}} T^{\mathrm{ChL1}}\end{aligned} \tag{2.16}$$

where $\dot{Q}$ is the amount of cooling power that the chiller can extract at a specific temperature and a_1 to a_6 are the coefficients of the function.

- The energy input to cooling output ratio ($r_{E,Q}$) as a function of T^{CoL1} and T^{ChL1}:

$$r_{E,Q}\left(T^{CoL1}, T^{ChL1}\right) = b_1 + b_2 T^{CoL1} + b_3 \left(T^{CoL1}\right)^2 + \ldots \\ b_4 T^{ChL1} + b_5 \left(T^{ChL1}\right)^2 + b_6 T^{CoL1} T^{ChL1} \tag{2.17}$$

where $r_{E,Q}$ is a measure for the energy efficiency of the chiller (this is also the inverse of the COP) and b_1 to b_6 are the coefficients of the function.

- The energy input to cooling output ratio ($r_{E,Q}$) as a function of the part load ratio (r_{PL}):

$$r_{E,Q}\left(r_{PL}\right) = c_1 + c_2 r_{PL} + c_3 r_{PL}^2 \tag{2.18}$$

where r_{PL} is the ratio between the actual cooling and the maximum available cooling of the chiller and c_1, c_2 and c_3 are the coefficients of the function.

These functions have been modified to show similar behavior in new ranges, by changing the coefficients of the curve to fit the desired temperature ranges, while keeping the curve's shape the same between the upper and lower temperature limits. Although the chiller model created from this approach is not based on a real chiller, the approach does lead to a stable model and is easy to apply. Therefore, this approach has been used to modify the chiller curves.

An overview of the original and the transformed temperature limits for these curves is given in Table 2.2. The temperature limits of T^{CoL1} are based on temperatures of this value which were seen in initial simulations. The temperature limits of T^{ChL1} are based on typical chilled water temperatures in DCs.

Table 2.2: Original and transformed temperature limits.

Temperature limit	**Original temperature** [°C]	**Shifted temperature** [°C]
Lower limit, T^{CoL1}	24	5
Upper limit, T^{CoL1}	35	35
Lower limit, T^{ChL1}	5	15
Upper limit, T^{ChL1}	10	25

2.3.2 EnergyPlus simulation algorithms

EnergyPlus offers a wide range of simulation settings and heat transfer algorithms. These algorithms are used to calculate the change of variables related to heat transfer in the HVAC system. They offer a trade-off between simulation speed and accuracy for these variables. Table 2.3 contains an overview of the heat transfer algorithms that have been used in the simulations.

Surface convection algorithm For both the inside and outside surface convection, the TARP [58] algorithm has been used, which is named after the Thermal Analysis Research Program (TARP). This is the standard option in EnergyPlus, and also the one which was utilized in the

data room model [44], therefore it is applied here as well. The TARP algorithm is a very simple, empirically based calculation of the surface convection:

$$h = \begin{cases} 1.31|\Delta T|^{\frac{1}{3}} & \text{if } \Delta T = 0.0 \text{ or surf is vertical} \\ 1.302|\Delta T|^{\frac{1}{3}} & \text{if } (\Delta T > 0.0 \text{ and surf is up}) \text{ or } (\Delta T < 0.0 \text{ and surf is down}) \\ 1.309|\Delta T|^{\frac{1}{3}} & \text{if } (\Delta T > 0.0 \text{ and surf is down}) \text{ or } (\Delta T < 0.0 \text{ and surf is up}) \end{cases} \tag{2.19}$$

where ΔT is the temperature difference between the surface and air, and $surf$ is the direction in which the surface faces. It is thus a very simple algorithm, but it is accurate enough for studies like this one where great accuracy is not required and computationally inexpensive.

Conduction heat balance algorithm The conduction heat balance algorithm is used to calculate the heat balance of the conduction processes in the system, mainly for the conduction between surfaces. The algorithm used for this is the conduction finite difference method [58], which uses a 1D finite difference solution in each construction element (such as a wall). This can accurately calculate the conduction but is computationally expensive. EnergyPlus does provide computationally cheaper options, such as transfer functions, but these led to errors in the model. As the source of these errors could not be identified, the more stable and accurate finite difference option has been chosen.

Zone air heat balance algorithm EnergyPlus offers 3 types of zone air heat balance algorithms: first order backwards difference; third order backwards difference and an analytical solution [58]. This thesis employs the third order backwards difference algorithm, as it offers the best accuracy. This is preferred over the added computational complexity, as the heat balance in the zone is the place where the ITE is actually cooled.

The zone air heat balance in EnergyPlus is defined as [43]:

$$\dot{m}^{\mathrm{AL}} c^{\mathrm{a}} \frac{dT^{\mathrm{AL2}}}{dt} = \dot{m}^{\mathrm{AL}} c^{\mathrm{a}} \frac{T^{\mathrm{AL1}}}{dt} + P^{\mathrm{ITE}} \tag{2.20}$$

Rewriting this for a third order backwards difference gives:

$$11T^{\mathrm{AL2}}(t) - 18T^{\mathrm{AL2}}(t-\Delta t) + 9T^{\mathrm{AL2}}(t-2\Delta t) - 2T^{\mathrm{AL2}}(t-3\Delta t) = 6\Delta t \frac{dT^{\mathrm{AL1}}}{dt} + \frac{6\Delta t}{\dot{m}^{\mathrm{AL}} c^{\mathrm{a}}} P^{\mathrm{ITE}} \tag{2.21}$$

From this follows that the algorithm thus offers a third order accuracy, which is desirable in the system that has relatively large timesteps.

Table 2.3: The algorithms used for the EnergyPlus model.

Calculation	Algorithm
Inside surface convection	TARP
Outside surface convection	TARP
Heat balance	Conduction Finite Difference
Zone air heat balance	Third order backward difference

2.3.3 Timestep sensitivity analysis

The accuracy of the EnergyPlus algorithms depends on the chosen timestep size. When the timestep becomes smaller, the accuracy of the model increases. However, this comes at the cost of an increased computation time of the simulation. Therefore, choosing the timestep size of the simulation depends on a trade-off between accuracy and simulation time. To assist with the selection of a good timestep, a timestep sensitivity analysis is performed. In this analysis, the dependence of the simulation accuracy and simulation time on the timestep size is determined.

To perform this timestep analysis, the baseline study (which is discussed in section 3.1) is performed with all possible timesteps in EnergyPlus $t^{\text{step}} \in \boldsymbol{t}^{\text{step}}$, where:

$$\boldsymbol{t}^{\text{step}} = [1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60]^{\text{T}} \tag{2.22}$$

These studies are compared to each other for the following parameters: room air temperature, the air loop mass flow, the total HVAC power, and the total simulation time. These parameters have been selected to compare the accuracy of the system on all important aspects.

To fairly compare the sensitivity of the model to the timestep size, the original, stepwise ITE load profile from the room model (shown in Figure 3.10) [44] is used. This ensures that the error is not a control error but is actually caused by the timestep size. For more information on the ITE load profiles used in this study, the reader is referred to section 3.3.

In an ideal timestep sensitivity analysis, the simulation results are compared to the analytical solution. Since no analytical solution is available, it is assumed that the smallest timestep provides the most accurate simulation. Therefore, the other simulation results are linearly interpolated and compared to the simulation with the 1 minute timestep.

Using the results of this analysis, a timestep that has a good trade-off between simulation time and accuracy can be selected. Another factor to be taken into account in this selection is that a typical data collection interval in DCs is around 10 to 15 minutes.

2.4 The HVAC controller optimization goal

The goal of this thesis is to design a controller that minimizes the total energy the HVAC system provided in section 2.2 uses over the course of any year. While doing so, it should be ensured that temperature constraints are met at all times, due to the mission-critical nature in DCs. This section will formalize this objective of the controller into a constraint minimization problem.

In the previous section, it was seen that this power, P^{HVAC}, depends on the state of the system, $\boldsymbol{s}_t$, which is the set of all thermodynamic properties of the system at time t. In turn, these states are a function of the two sets of variables discussed in section 2.2. Firstly, the high-level controller which determines setpoint variables, $\boldsymbol{x}$. Secondly, the disturbance variables, $\boldsymbol{d}$. It was also seen that the system is stochastic, therefore, only an expectation of the power can be

minimized. Mathematically, this problem is defined as:

$$\begin{aligned} \underset{\boldsymbol{x_t}}{\text{minimize}} \quad & \sum_{t=0}^{T} \mathbb{E}\left[P_t^{HVAC}\left(\boldsymbol{s_t}\left(\boldsymbol{x_t}, \boldsymbol{d_t}\right)\right)\right] \\ \text{subject to} \quad & c_{i,t} \leq 0, \quad \forall i, t \\ & \boldsymbol{x_t} \in \mathcal{X}, \quad \forall t \end{aligned} \tag{2.23}$$

where:

- P_t^{HVAC} is the total power of the HVAC equipment. It can be expressed as the sum of the power of the individual HVAC components at time t:

$$P_t^{\text{HVAC}} = P_t^{\text{CT}} + P_t^{\text{CoL Pump}} + P_t^{\text{Chiller}} + P_t^{\text{ChL Pump}} + P_t^{\text{Hum}} + P_t^{\text{AL Fan}} \tag{2.24}$$

- $c_{i,t}$ are the constraint equations for the performance in the room. This system has 3 constraint equations, 2 of which are to ensure that the cold air temperature stays in a certain bandwidth around its setpoint:

$$c_{1,t} = -T^{\text{AL1}} + T_{\text{SP}}^{\text{AL1}} - bw^{T^{\text{AL1}}} \tag{2.25}$$

$$c_{2,t} = T^{\text{AL1}} - T_{\text{SP}}^{\text{AL1}} - bw^{T^{\text{AL1}}} \tag{2.26}$$

  where $bw^{T^{\text{AL1}}}$ is the allowable bandwidth of this temperature. Next to that, there is a constraint on the violation of a temperature limit of the temperature after the servers T^{AL2}. This is the most important constraint, as it means the servers overheat if it is violated. It is defined as:

$$c_{3,t} = T^{\text{AL2}} - T_{\text{constr}}^{\text{AL2}} \tag{2.27}$$

  where $T_{\text{constr}}^{\text{AL2}}$ is the maximum allowable temperature.

- The actuation variable vector at time t, $\boldsymbol{x_t}$, is defined as:

$$\boldsymbol{x_t} = \left[T_{\text{SP},t}^{\text{CT}}, \dot{m}_{\text{SP},t}^{\text{CoL}}, T_{\text{SP},t}^{\text{ChL1}}, \dot{m}_{\text{SP},t}^{\text{ChL}}, \dot{m}_{\text{SP},t}^{\text{AL}}, \phi_{\text{SP},t}^{\text{AL4}}\right]^{\text{T}} \tag{2.28}$$

- The control space $\boldsymbol{\mathcal{X}}$ is the vector space containing all actuation variables. It is bounded by the upper and lower limits of the HVAC equipment.

- The disturbance vector $\boldsymbol{d_t}$ is given by:

$$\boldsymbol{d_t} = \left[T^{\text{outdoor}}, P^{\text{ITE}}\right]^{\text{T}} \tag{2.29}$$

The constraint equations contain a number of parameters of which the value still has to be determined. From discussions within Coolgradient, it follows that $T_{\text{SP}}^{\text{AL1}} = 24°\text{C}$, $bw^{T^{\text{AL1}}} = 3°\text{C}$ and $T_{\text{max}}^{\text{AL2}} = 30°\text{C}$ are typical values for these parameters. Therefore, this will be used in the remainder of this thesis.

To conclude, the minimization problem of Equation (2.23) is the goal of the controller that will be proposed to optimize the HVAC system. The goal of this thesis is to implement a RL based controller which performs better at this minimization problem than a conventional controller.

2.5 Summary

In summary, this section has introduced a simple chiller plant in a DC. For this system, it has provided the energy balance in the system and relations between the thermodynamic quantities and power of the HVAC components. Following that, it provided variables that can be used to control this system and variables that disturb it. Based on this system, a simulation model of the HVAC system has been constructed. For this, EnergyPlus is used as simulation software. The model has been constructed using an existing DC room model, expanded with a HVAC system with a custom chiller model. A timestep sensitivity study has been outlined to test its sensitivity to changes in the timestep.

Finally, the objective of this thesis has been formalized, which is to design a HVAC controller that minimizes the power used by the HVAC equipment of a DC, while still ensuring the operation of the system is safe and constraints are met. The upcoming chapters detail the methodology, experiment setup, and results of the design of this controller, providing a thorough analysis of the RL framework that will be proposed. At the end of the thesis, the performance of the RL agent will be examined based on the minimization problem stated in Equation (2.23).

Chapter 3

Methodology

The use of RL as a controller to improve the energy efficiency of a DC is a promising method. The key advantages are that RL does not require a detailed model, thus avoiding modelling bias and reducing the labour involved in modelling each individual DC. Additionally, RL can adapt to changes in the model and new situations. Therefore, the goal of this thesis is to enhance the energy efficiency of current controllers for DC HVAC systems by developing an RL-based controller.

Section 2 has formalized this controller optimization problem in Equation (2.23). In addition to this main goal, the thesis aims to provide insights into the effect of the reward function on the behaviour of the controller. This section provides the methodology of how such an agent is designed, tuned, tested, and compared to conventional controllers.

To fairly show the performance improvements of the RL-based controller, a conventional controller is designed and tested. This controller acts as the baseline for the RL controller. Its design is discussed in section 3.1.

Next, in section 3.2, a framework for the RL framework is proposed. Here the system is described as a mathematical decision-making framework, which includes the reward of the environment. For the reward, a framework is suggested in which weights can be adjusted. A suitable algorithm for this framework is then selected.

Then, in section 3.3, the data partitioning of the disturbance variables (T^{outdoor} & P^{ITE}) into training, validation and testing data is discussed. The partitioning of this data ensures that the RL based controller is robust to disturbances.

Finally, section 3.4, describes several experiments to fine-tune the RL framework. First, a setup for hyperparameter tuning experiments is provided. These experiments aim to tune the hyperparameters of the RL algorithm, ensuring that the trained agents always learn stably. Finally, reward-tuning experiments are performed. Their goal is to gain insights into how different reward settings affect the RL controller. Using these insights, an optimally tuned controller is selected for comparison to the baseline.

3.1 Baseline study

The baseline experiments are performed by simulation of the EnergyPlus model using a conventional controller for the HVAC system. The goal of these experiments is to have results of a conventional controller to which the trained RL algorithms can be compared. This section first discusses this conventional controller and its settings in subsection 3.1.1. Then, it describes how this controller is used in the baseline experiments.

3.1.1 Baseline controller

The conventional controller operates based on fixed temperature setpoints maintained throughout the year. EnergyPlus calculates the required mass flow rates for each loop to achieve these temperature setpoints. The conventional control strategy consists of the combination of these predetermined setpoints and the corresponding mass flow rates.

For the calculation of the mass flow rates, EnergyPlus uses a hierarchical structure that works from the inside of the HVAC system outwards. First, the mass flow in the AL is calculated based on the mass flow rate required to cool the server room sufficiently. Next, the mass flow of the ChL is calculated based on the requirements of the CC to cool the air adequately. Finally, the mass flow required by the CoL is calculated based on the water in the ChL. To align with this hierarchy, the conventional controller is discussed following the same structure.

AL control Figure 3.1 shows the schematic representation of the Air Loop. At 3 points in the Air Loop, setpoints are set to achieve desired behaviour of the system:

- **AL1** At this location, a setpoint is used to control the temperature which leaves the CRAH into the raised floor. This is used to meet the constraints from Equations (2.25) and (2.26)
- **AL2** At this location, a setpoint is used to control the maximum air temperature after the servers. This is used to ensure the constraint from Equation (2.27). The maximum air temperature setpoint only interferes with the system if the air temperature at that point is higher than the limit set in the setpoint.
- **AL4** Here, right after the humidifier inside the CRAH, a humidity setpoint is applied to ensure the desired humidity in the system.

These setpoints determine the settings in the AL. The corresponding air mass flow rate, $\dot{m}^{\mathrm{AL}}$, is calculated based on the method which is used in the EnergyPlus server room model from [44]. The calculation of $\dot{m}^{\mathrm{AL}}_{\mathrm{SP}}$ is based on the heat transfer rate of the servers. It is defined as follows:

$$\dot{m}^{\mathrm{AL}}_{\mathrm{SP}} = \frac{\dot{Q}^{\mathrm{ITE}}}{c^{\mathrm{air}}\left(T^{\mathrm{AL2}}_{\mathrm{SP}} - T^{\mathrm{AL1}}\right)} \tag{3.1}$$

where $\dot{Q}^{\mathrm{ITE}}$ is the cooling load of the ITE (assumed to be equal to P^{ITE}), and c^{air} is the specific heat of the air. Since there are setpoints for both T^{AL1} and T^{AL2}, the desired mass flow can be determined using this formula.

ChL control The ChL is displayed in Figure 3.2. This loop employs just one temperature setpoint, which is the following:

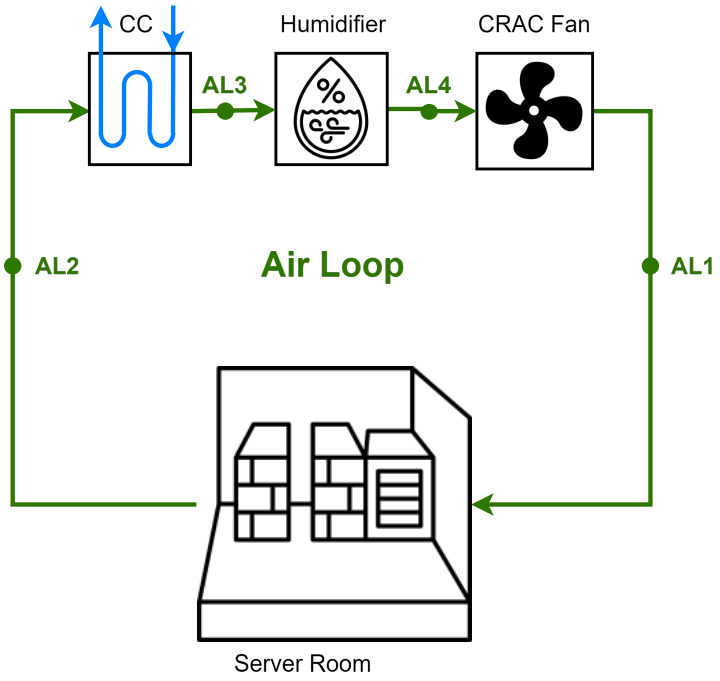


Figure 3.1: Schematic representation of the air loop (Zoomed in from Figure 2.5).

- **ChL1** A setpoint is used for the water temperature that leaves the chiller. This directly influences the chiller power, as more energy is required to cool the water to a lower temperature. This temperature also affects the temperature of the water entering the cooling coil, and thus how much the air in the air loop is cooled.

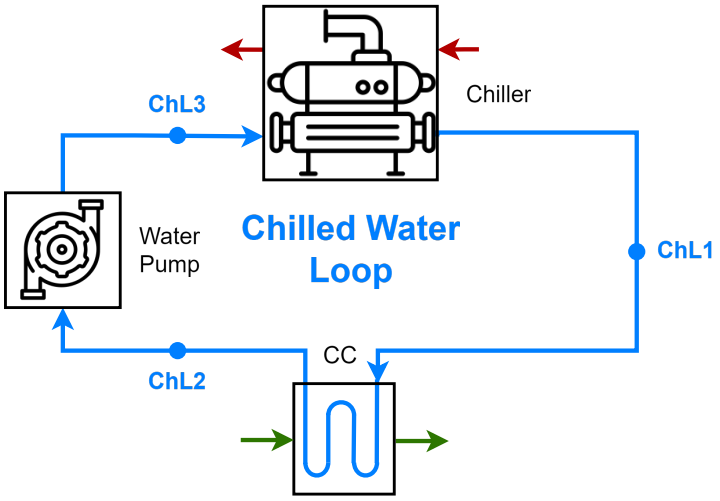


Figure 3.2: Schematic representation of the chilled water loop (Zoomed in from Figure 2.5).

The corresponding mass flow rate in the ChL, $\dot{m}_{\mathrm{SP}}^{\mathrm{ChL}}$, is determined by the energy balance in the cooling coil from Equation (2.1). Rewriting this energy balance gives:

$$\dot{m}_{\mathrm{SP}}^{\mathrm{ChL}} = \frac{\dot{Q}_{\mathrm{CC}}^{\mathrm{AL}}}{\eta_{\mathrm{CC}} c_{\mathrm{water}} \left(T_{\mathrm{SP}}^{\mathrm{ChL1}} - T^{\mathrm{ChL2}}\right)} \tag{3.2}$$

where $\dot{Q}_{CC}^{AL}$ follows from the air loop, and T^{ChL2} is calculated by EnergyPlus [59].

In this loop, also T_{SP}^{ChL1} is controlled. This is done by a thermostat in EnergyPlus internally, which switches the chiller on or off when the temperature deviates from the setpoint. This is a large simplification of reality, as it is not possible to switch a chiller on and off this often.

CoL In Figure 3.3, the CoL is shown schematically. Again, this loop employs just a single setpoint.

- **CoL1** A so-called offset temperature setpoint is used for the water that leaves the CT. This type of setpoint keeps the temperature to a certain predefined offset from another temperature ($T^{outdoor}$ in this case). In this way, it differs from the other setpoints that have been used, as they keep the temperature at a fixed value.

To achieve this setpoint, $\dot{m}_{SP}^{CoL}$ always runs at maximum power, and r^{CT} is determined according to the energy balance from Equation (2.11):

$$r^{CT} = \frac{\dot{Q}^{CoL,\,CT}}{\dot{m}_{max}^{CT,\,Fan}\left(T_{out}^{outdoor} - T_{in}^{outdoor}\right)} \tag{3.3}$$

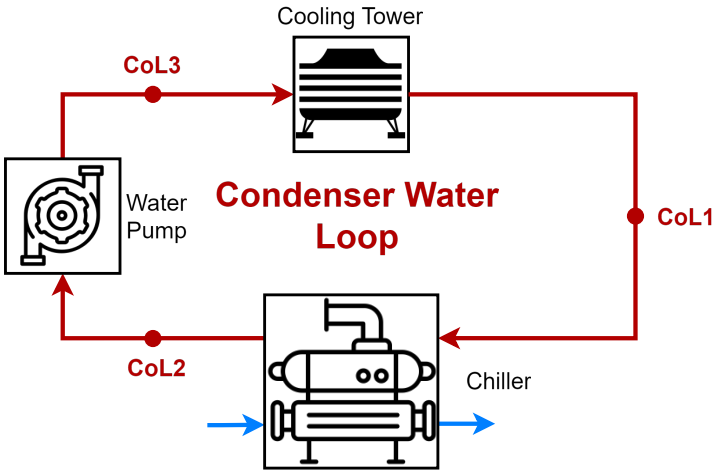


Figure 3.3: Schematic representation of the condenser water loop (Zoomed in from Figure 2.5).

To summarize, the conventional controller consists of setpoints, for both important temperatures and humidity, to determine the desired state of the HVAC system. The mass flows of the cooling loops are then calculated to meet both these setpoints and the cooling demand of the HVAC system. This controller has two simultaneous objectives: on one hand, no constraints should be violated by the system, on the other hand, the system should be operating in an energy-efficient manner.

3.1.2 Baseline experiments

As has been mentioned before, the baseline experiments are performed using this conventional controller and fixed setpoints throughout the year. These settings are based on the constraints in section 2.4. The weather and IT load data in the experiments are of the test year, 2023. To tune the baseline controller, simulations with multiple values of its most important parameter, T_{SP}^{ChL1}, will be performed.

3.2 Reinforcement Learning framework

In this section, the design of a RL framework for the RL-based controller of the HVAC system is discussed. First, a general background on RL is given in subsection 3.2.1. Then, using this understanding of the basics of RL, subsection 3.2.2 describes the HVAC system in a Markov Decision Process (MDP), which is a mathematical decision-making framework that is used in RL. The MDP is a very important part of the controller design, as it defines the variables the controller observes (states), the variables it can control (actions), and the function the controller tries to maximize (the reward). After the framework is defined, a suitable RL algorithm which will be trained is selected in subsection 3.2.3.

3.2.1 Reinforcement Learning working principle

This section will cover the basic working principle and terminology of RL, and how it applies to the DC's HVAC system. RL learns to optimally control complex systems without needing a predefined model [33, 34, 35]. RL algorithms ("agents") interact with a system ("environment") by mapping situations ("states") to control actions ("actions") to maximize some reward function of the environment [33].

Figure 3.4 shows the general framework in which an agent interacts with an environment. This framework is time-discrete. The goal of this agent is to learn to interact optimally with an unknown environment. The agent sends a control action to the environment, which then returns the next state and the reward of the next timestep. Through a lot of interactions with the environment, the agent can learn to predict the next state and reward for a given state-action pair [33]. For a DC HVAC system, the environment is the HVAC system, the agent is the controller, and the reward is a function with the goal of solving the control minimization problem of Equation (2.23). The simplest example of a reward function (without meeting constraints) is:

$$r_t = -P_t^{HVAC} \tag{3.4}$$

where r_t is the reward and P_t^{HVAC} is the HVAC power at time t.

The main goal of using RL agents as controllers is to learn a mapping from the thermodynamic states of the HVAC model to control setpoints that minimize the total HVAC power while meeting constraints. This mapping is known as the optimal policy, $\boldsymbol{\pi}^*$. In other terms, the policy is the learned control strategy of a RL-based controller. When an agent learns this policy, the exploration-exploitation trade-off is very important. This is the balance between exploiting the knowledge that has already been learned by an agent and exploring new states, which could lead to better policies than the states already known[33, 34].

Some algorithms try to learn this policy directly, while others try to learn a value function to assess the performance of the policy. These functions give the value of being in a certain state. The most basic value function is the state-value function, $V^*(\boldsymbol{s})$, which is defined as the maximum expected reward when starting from a state $\boldsymbol{s}$ and following the optimal policy π^*. The state-value function is given by the equation:

$$V^*(\boldsymbol{s}) := \max_{\boldsymbol{a}} \sum_{\boldsymbol{s'}} P(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a}) \left[r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'}) + \gamma V^*(\boldsymbol{s'}) \right] \tag{3.5}$$

where $P(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})$ is the probability of transitioning to any state $\boldsymbol{s'}$ from the current state $\boldsymbol{s}$ given action $\boldsymbol{a}$. The term $r(\boldsymbol{s},\boldsymbol{a},\boldsymbol{s'})$ represents the reward received after transitioning. Also, a discount

factor, γ, is implemented for the future rewards. This discounts the value of future expected rewards exponentially. Research shows that discounting future rewards leads to better overall results [35], as it is less certain that these future rewards will actually be received. This is especially important in the stochastic HVAC system of this thesis.

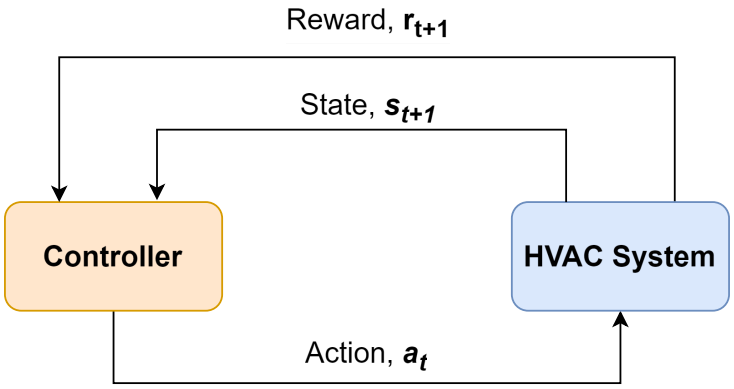


Figure 3.4: The RL framework, applied to the current system.

3.2.2 The Markov Decision Process

In RL, MDP's are used to model the decision-making structure. An MDP is a mathematical framework where the outcomes are partly random and partly controlled by a decision maker, which is the HVAC controller in this thesis. The MDP is thus a structure to describe the HVAC model as an environment for a RL agent.

An MDP is defined as a 5-tuple: $(\mathcal{S}, \mathcal{A}, P_a, R_a, \gamma)$. In this tuple, $\mathcal{S}$ is the state space, a vector space that contains all the possible states in which the HVAC model can be. $\mathcal{A}$ is the action space, the vector space containing all possible combinations of setpoints that can be adjusted by the agent. $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that, under the current state and action (s_t & a_t) the HVAC system will end up in a certain next state s'. This is done by a single simulation step of the EnergyPlus model. R_a is the immediate reward in a certain state and γ is a discount factor for the future rewards. In this section, the implementation of the first 4 of the components of the tuple is discussed in more detail. γ is a hyperparameter of the system and is therefore subject to hyperparameter tuning. Therefore, it is not further discussed in this section.

3.2.2.1 State space

The state space is a set of states for the model which contains all possible states in which an environment can possibly be. More formally: $s_t \in \mathcal{S}, \forall t$, where s_t is the state of the model at time t and $\mathcal{S}$ is the state space.

State definition To fully define the state space, the states of the system should be defined. This consists of the variables that are included in the state, their upper and lower limits, and whether these variables are continuous or discrete. The state consists of the following groups of variables:

- Thermodynamic properties of the fluids in the HVAC loops.

- HVAC component power.
- Disturbance variables.

To fully describe the state of the system, while keeping the dimensionality of the state space as low as possible, as few thermodynamic properties as possible are put in the state. To determine the properties that should be put in the state, the state postulate is used. According to this, all thermodynamic properties can be calculated if 2 variables are known. Since EnergyPlus does not model changes in pressure (which is a major simplification), only 1 variable per point in the system is required. Because the component models in EnergyPlus are based on temperatures, as well as the constraints in the optimization problem, it is only logical to use the temperature at all points (CoL1, Col2, ..., AL4) as the states of the system. Next to that, the humidity of the AL is added to the state since it cannot be accurately calculated due to the continuous pressures. The mass flows of all 3 loops are also taken as states.

Next to this, the power of each HVAC component is added to the state space, such that the agent will always know where power is used (and could be saved). Finally, the disturbance variables (Equation (2.29)) are added to the state.

The definition of the complete set of states is defined as follows:

$$\boldsymbol{s} = \left[T^{\text{CoL1}}, \ldots, T^{\text{AL4}}, \phi^{\text{AL1}}, \ldots, \phi^{\text{AL4}}, \dot{m}^{\text{CoL}}, \dot{m}^{\text{ChL}}, \dot{m}^{\text{AL}}, P^{\text{CT}}, \ldots, P^{\text{AL Fan}}, P^{\text{ITE}}, T^{\text{outdoor}}\right]^{\text{T}} \tag{3.6}$$

It should be clear that, since all these states are physical properties, the state space is continuous.

Markov property The Markov property is fulfilled when a future state only depends on the current state and actions taken, and not on the preceding states. This property is met in the EnergyPlus simulations since the component models are simulated using only current information. In reality, however, the property might not be fulfilled because of the so-called 'thermal inertia' of the system.

3.2.2.2 Action space

The action space is the vector space containing all possible actions done by the agent. Mathematically it is defined as: $\boldsymbol{a_t} \in \boldsymbol{\mathcal{A}}, \forall t$, where $\boldsymbol{a}_{\text{t}}$ is the action at time t and $\boldsymbol{\mathcal{A}}$ is the action space.

In Equation (2.28), the variables in the system that can be directly actuated have been defined (and it is restated here as Equation (3.7) for the reader's convenience). These variables can be used to control the system and can thus possibly be used as action variables. However, some of these variables have been excluded from the action space.

$$\boldsymbol{x_t} = \left[T^{\text{CT}}_{\text{SP},t}, \dot{m}^{\text{CoL}}_{\text{SP},t}, T^{\text{ChL1}}_{\text{SP},t}, \dot{m}^{\text{ChL}}_{\text{SP},t}, \dot{m}^{\text{AL}}_{\text{SP},t}, \phi^{\text{AL4}}_{\text{SP},t}\right]^{\text{T}} \tag{3.7}$$

First of all, it was seen in initial experiments that, with the baseline settings, the value of $T^{\text{CT}}_{\text{SP}}$ was barely ever met. Therefore, changing the setpoint does not affect the system that much since the cooling tower will just try to cool at its maximum capacity for most temperature setpoints. Also, the cooling tower has a single-speed fan, when a different speed is required EnergyPlus assumes it is turned on and off multiple times per timestep. In reality, this rapid

switching is bad practice. Therefore, it is chosen to leave this setpoint as in the baseline, since the added action increases the complexity of training the algorithm, while the added benefits are assumed to be quite low. Using the same reasoning, $\dot{m}^{\text{CoL}}_{\text{SP},t}$ is also omitted from the action space.

Next to that, meeting the setpoint of the relative air humidity is just a constraint of the system. Therefore, it would be nonsensical to start modifying this setpoint. It would be interesting to adjust the internal control of the humidifier to meet this setpoint more efficiently if it was possible. However, this is not possible in EnergyPlus.

When removing these variables from Equation (3.7), the action vector can be defined as:

$$\boldsymbol{a_t} = \left[T^{\text{ChL1}}_{\text{SP},t}, \dot{m}^{\text{ChL}}_{\text{SP},t}, \dot{m}^{\text{AL}}_{\text{SP},t}\right]^{\text{T}} \tag{3.8}$$

Similar to the state space, these variables are continuous.

3.2.2.3 State transition dynamics

The transition of the current state to the next state is calculated by performing a single simulation step in the EnergyPlus model. Since the HVAC system is stochastic, the state transitions are stochastic as well. This is caused by the 2 disturbance variables, T^{outdoor} & P^{ITE}. Because these variables are stochastic by nature and not influenced by the rest of the system, they introduce stochasticity to the rest of the system.

3.2.2.4 Reward function framework

The goal of a reward function is to reward the agent during training in such a way that it can learn a good policy. In the case of this system, a good policy is a policy that minimizes P^{HVAC} while preventing the violation of constraints in the optimization problem from Equation (2.23).

A tuneable reward function framework is proposed, which allows for the investigation of the effects of different terms in the function on the controller behaviour. The function consists of 4 parts, a reward for minimizing P^{HVAC}, a reward for meeting $T^{\text{AL1}}_{\text{SP}}$ (Equations (2.25) & (2.26)), a penalty for violating $T^{\text{AL2}}_{\text{max}}$ (Equation (2.27)) and a penalty to avoid severe oscillations of the action variables of the agent. This leads to the following reward framework:

$$r_{t_i} = \underbrace{r_{\text{P}}\left(P^{\text{HVAC}}_{t_i}\right)}_{\text{HVAC Power reward}} + \underbrace{\lambda_{T^{\text{AL1}}_{\text{SP}}} \cdot r_{\text{T}}\left(T^{\text{AL1}}_{t_i}\right)}_{\text{CRAH setpoint reward}} + \underbrace{\lambda_{T^{\text{AL2}}_{\text{constr}}} \cdot p_{\text{T}}\left(T^{\text{AL2}}_{t_i}\right)}_{\substack{\text{Server outlet} \\ \text{temperature penalty}}} + \underbrace{\boldsymbol{\lambda}_{\text{a}} \cdot \boldsymbol{p}_{\text{a}}\left(\Delta \boldsymbol{a_{t_i}}\right)}_{\substack{\text{Action fluctuation} \\ \text{penalty}}} \tag{3.9}$$

where the terms in this function are defined as follows:

- r_{t_i}, the total reward at time t_i.
- r_{P}, the reward, related to power, which is a function of the HVAC power at time t_i.
- $\lambda_{T^{\text{AL1}}_{\text{SP}}}$, a scaling factor for the reward related to the setpoint temperature of the CRAH.
- r_{T}, a reward for meeting the temperature setpoint for the air leaving the CRAH (point AL1 in the HVAC system) at time t_i.

- $\lambda_{T^{\text{AL2}}_{\text{constr}}}$, a scaling factor for the penalty on violating the server outlet temperature constraint.
- p_{T}, the penalty for violating the server outlet temperature constraint, which is a function of the air temperature after the servers (point AL2) at time t_i.
- $\boldsymbol{\lambda}_{\text{a}}$, a vector containing scaling factors for the penalty on fluctuations in each of the actions.
- $\Delta \boldsymbol{a}_{t_i}$, the difference between the action at time t_i and t_{i-1}: $\boldsymbol{a}_{t_i} - \boldsymbol{a}_{t_{i-1}}$
- $\boldsymbol{p}_{\text{a}}$, a vector of penalties on fluctuations of the actions, which is a function of

HVAC power reward The reward on P^{HVAC} should be high when the power usage is low, and low when the power usage is high. Thus $r_{\text{P}} = -P^{\text{HVAC}}$. To make the scaling of the other parts of the reward function more intuitive, this reward is normalized between 0 and 1, where the reward is 0 at the maximum HVAC power and 1 when no HVAC power is used. This leads to the following equation (as shown in Figure 3.5):

$$r_{\text{P}}\left(P^{\text{HVAC}}_{t_i}\right) = 1 - \frac{P^{\text{HVAC}}_{t_i}}{P^{\text{HVAC}}_{\text{max}}} \tag{3.10}$$

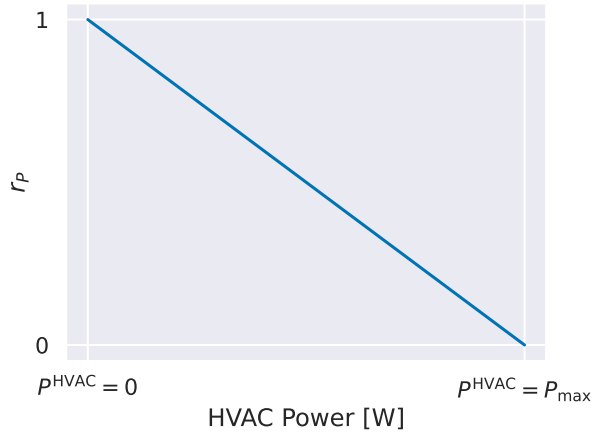


Figure 3.5: Reward for the HVAC power

CRAH setpoint reward To encourage the agent to meet the CRAH leaving air temperature setpoint, a Gaussian reward is defined as:

$$r_{\text{T}} = e^{-\frac{1}{2}\left(\frac{T^{\text{AL1}} - T^{\text{AL1}}_{\text{SP}}}{\sigma_{\text{SP}}}\right)^2} \tag{3.11}$$

where σ_{SP} is a tuneable parameter for the width of the reward peak. Figure 3.6 shows this reward for several values of σ_{SP}.

The choice of this Gaussian function as a reward for meeting the setpoint is based on the reward function in [45]. The parameter σ_{SP} allows for tuning of the temperature range around the setpoint which is rewarded. Unlike in [45], no penalty is given for deviating too much from this temperature. This choice is made since having too high temperatures will already be penalized for T^{AL2}, which is related to T^{AL1}.

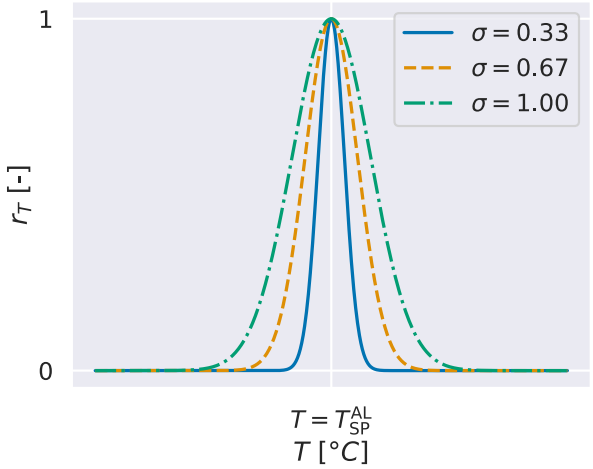


Figure 3.6: The room temperature setpoint reward, where σ is a tunable parameter

Server outlet temperature penalty To meet the server outlet temperature constraint:

$$T^{\text{AL2}} \leq T^{\text{AL2}}_{\text{constr}} \tag{3.12}$$

this penalty has been added to Equation (3.9). As mentioned before, the penalty consists of a function of T^{AL2}. All functions can of course be implemented here, and a few candidates are displayed in Figure 3.7. As it is no problem for the temperature to be below its constraint, all of these functions are (close to) 0 when the constraint is met. When the constraint is violated, these functions all penalize the temperature constraint.

The first candidate (shown in Figure 3.7a) is the Rectified Linear Unit (ReLU), which is defined as:

$$p_{\text{T, ReLU}} = -\max\left(0, \Delta T\right), \quad \text{where} \quad \Delta T = T^{\text{AL2}} - T^{\text{AL2}}_{\text{constr}} \tag{3.13}$$

This is a straightforward penalty function, which makes it easy to tune. However, the fact that it just linearly penalizes the severity of reward violations might cause problems during the training of the RL agent. As more severe violations of the constraint prove an increasingly higher risk to the server [7], another penalty function might be required to train a well-behaving agent.

The next candidate for the penalty function is shown in Figure 3.7b. This is the softplus penalty, which is basically a smooth approximation of the ReLU function. It is defined as follows:

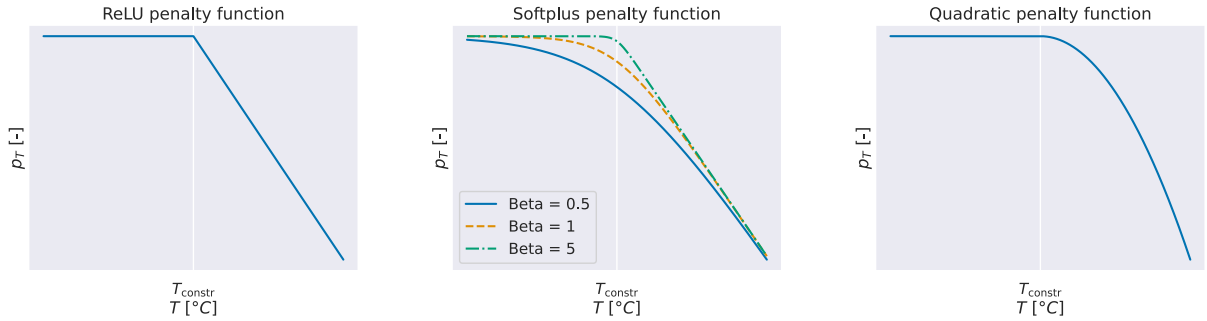
$$p_{\text{T, softplus}} = -\frac{\ln\left(1 + e^{\beta \cdot (\Delta T)}\right)}{\beta} \tag{3.14}$$

In this function, β is a tuneable parameter for the "sharpness" of the curve of the function. This function already provides a penalty when the temperature is at the setpoint, which may make the agent learn to stay away from this setpoint temperature after training. This could make the agent more robust to sudden changes in the environment. However, the softplus function might also penalize the agent too much during safe operation, which could lead to sub-optimal behaviour regarding the energy efficiency of the system. Next to that, the addition of β introduces a new parameter to tune, which leads to increased complexity of the system.

The final candidate function is the ReLU squared function, which is displayed in Figure 3.7c. This function is defined as follows:

$$p_{\text{T, ReLU2}} = -\left[\max\left(0, T^{\text{AL2}} - T^{\text{AL2}}_{\text{constr}}\right)\right]^2 \tag{3.15}$$

By squaring the ReLU function, the larger violations of the temperature constraint are penalized increasingly harder. However, a drawback might be that, after a large constraint violation during exploration, the agent might learn too safe behaviour, again leading to sub-optimal energy efficiency.



(a) The ReLU temperature penalty function

(b) The softmax temperature penalty function

(c) The ReLU squared temperature penalty function

Figure 3.7: The three candidates for the temperature constraint penalty function

Action fluctuation penalty During exploratory experiments, it was seen that the agent sometimes learned that it would be beneficial to lower the chilled water temperature as much as possible to cool the room quickly, and then on the subsequent timestep, it would turn off all HVAC equipment. Although, on average, this led to energy savings, this behaviour is not desirable in real life, as it can deteriorate the expected lifetime of the HVAC equipment. To mitigate this problem, a penalty on action fluctuations has been added to Equation (3.9). In Figure 3.8, 3 different candidates for this penalty are plotted. The penalty is calculated and weighted for each individual action and then added.

The first candidate, shown in Figure 3.8a, is a penalty on the absolute difference between the current and previous action. This is defined as:

$$p_{\text{a, abs}} = -\left|\Delta \boldsymbol{a}_{t_i}\right| \tag{3.16}$$

The application of this penalty is quite straightforward. The only variables to be tuned are the parameters in $\boldsymbol{\lambda}_{\text{a}}$. However, this reward has a potential drawback in that it penalizes every action that is not exactly the same as the previous action. However, small changes in actions should be allowed, and are often even desired. Although the penalty for small deviations in the action is quite small, it could still cause the agent to learn a policy where the actions are as constant as possible.

The second candidate function tries to mitigate this problem by squaring the difference between the current and previous action:

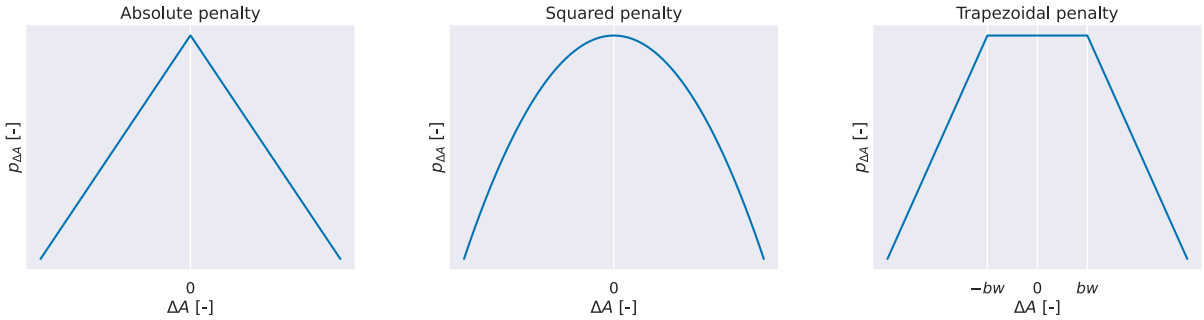
$$p_{\text{a, quad}} = -\left(\Delta \boldsymbol{a}_{t_i}\right)^2 \tag{3.17}$$

This way, small deviations between the actions are penalized less, and larger differences are penalized more than when using Equation (3.16). However, it still does not completely solve the problem, as there is still a penalty for every change between the current and previous action.

The final candidate is a trapezoid penalty function, as shown in Figure 3.8c. It is defined as follows:

$$p_{\text{a, trap}} = -\left|\max\left(0, \Delta \boldsymbol{a}_{t_i} - BW_a\right)\right| \tag{3.18}$$

This penalty has a certain bandwidth (BW_a) where action changes are not penalized. When actions are out of this bandwidth, they do get penalized linearly. This way, large deviations in the action are penalized, but for smaller changes in actions, the reward function is not affected at all. However, this comes at the cost of an additional parameter to tune.



(a) The absolute action penalty function

(b) The absolute action penalty function

(c) The absolute action penalty function

Figure 3.8: The candidates for the penalty on $\Delta \boldsymbol{a}$

To summarize, the reward consists of a reward for minimizing P^{HVAC}, a reward for meeting $T_{\text{SP}}^{\text{AL1}}$, a penalty for violation of the T^{AL2} setpoint, and a penalty for fast action oscillations. Table 3.1 summarizes the hyperparameters in the reward function. Each parameter is tunable and can lead to different results in the behaviour of the policy. The tuning of this reward is discussed in subsection 3.4.2.

Table 3.1: Summary of the reward hyperparameters.

Parameter	Description
$\lambda_{T_{\text{SP}}^{\text{AL1}}}$	The weight of the reward for avoiding setpoint deviations.
σ_{SP}	The standard deviation of the Gaussian setpoint deviation reward.
$\lambda_{T_{\text{constr}}^{\text{AL2}}}$	The weight of the penalty for violating $T_{\text{constr}}^{\text{AL2}}$.
p_{T}	The type of penalty function for violating $T_{\text{constr}}^{\text{AL2}}$.
β_{softplus}	Applies if p_{T} is *softplus*, the "sharpness" of the curve.
$\lambda_{\text{a},T_{\text{chill,SP}}}$	The vector of weights for the penalty on action fluctuations.
p_{a}	The penalty function for action fluctuations.
$bw_{\text{a,trapezoidal}}$	Applies if p_{a} is trapezoidal, the width where no penalty is applied.

3.2.2.5 MDP summary

To summarize, the MDP defined in this section consists of the 5-tuple $(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, P_{\text{a}}, R_{\text{a}}, \gamma)$. Here, $\boldsymbol{\mathcal{S}}$ consists of the temperatures and mass flow rates of all the points in the system, the humidity

in the AL, the disturbance variables P^{ITE} and T^{outdoor} and the power of the HVAC components. $\mathcal{A}$ consists of the mass flow rates of the ChL & AL and the chilled water setpoint temperature. The state transition model is handled by the EnergyPlus model, which is stochastic due to the stochastic nature of the disturbance variables. For the reward, a reward function framework has been designed. The framework can be tuned in tuning experiments to investigate how each penalty affects the behaviour of the controller. The reward provides a reward at every timestep, therefore it provides dense feedback. Finally, γ is treated as a hyperparameter. A summary of the properties of the MDP is given in Table 3.2.

Table 3.2: Summary of the MDP properties

Aspect	Description	Details/Examples
State Space	Continuous, Large	$\boldsymbol{s} \in \mathbb{R}^{33}$
Action Space	Continuous, Small	$\boldsymbol{a} \in \mathbb{R}^{3}$
Transition Dynamics	Stochastic	Simulated by EnergyPlus, based on HVAC system behavior.
Reward	Dense	Tuneable framework to balance P^{HVAC} reduction and constraint handling.

3.2.3 Reinforcement Learning algorithm selection

The selection of a RL algorithm which learns to optimally control the HVAC system is very important for the performance of the controller. To help with this selection, selection criteria are set up, based on practical requirements and a classification of the algorithm based on the MDP. The classification parameters are based on the books [33, 34] and the classification framework from [60]. Using the requirements and classification, the choice of Proximal Policy Optimization (PPO) as a suitable algorithm is motivated. Finally, a detailed explanation of how the PPO algorithm works is provided.

3.2.3.1 Algorithm selection criteria

For the selection of a RL algorithm, several criteria have been distilled. These criteria follow from the MDP and are defined with the future implementation of RL on real HVAC systems in mind. The selected algorithm should match these criteria:

- The algorithm should be able to handle a **relatively large, continuous state-space**, as the HVAC system has continuous state parameters.
- The algorithm should be able to handle a **continuous action-space**, as the HVAC system also has continuous action variables, which can not be logically discretized.
- The algorithm should also be able to work with **stochastic** systems.
- The algorithm should be **sample efficient**, as an agent should be able to converge to an optimal policy as quickly as possible when it would be applied on a real DC, to reduce the period in which it is still learning and constraints are violated. Next to that, the

timesteps in reality are quite large, meaning that sample inefficient agents lead to large energy expenses during a long time.

- A suitable RL algorithm should have **stable learning**, as unstable learning increases the risk of creating a controller that is not energy efficient, or worse, even well-tuned agents will not always able to keep the temperatures in the system low enough after training, which is disastrous in the mission-critical environment of a DC.
- The algorithm should be **model-free**, as the goal of this thesis is to create a controller without the requirement of a predefined model.
- The algorithm should be an **actor-critic** algorithm. This type of algorithm combines the strengths of both policy-based and value-based methods: the "actor" learns the policy directly, while the "critic" estimates the value function. This has the best of both worlds, as the critic helps to stabilize learning by reducing the high variance typically seen in policy-based methods [33], while the actor is better at learning continuous action spaces.

3.2.3.2 Algorithm choice

To summarize, the algorithm that is most suitable for the use on the HVAC model should be able to handle a continuous action-space; should be able to handle stochastic models; be preferably of the actor-critic nature; should be model-free and has to be able of online learning.

Numerous algorithms that match these requirements have been considered, but the final algorithm choice is the PPO [61] algorithm. This is a state-of-the-art actor-critic algorithm, which shows good performance when compared to similar algorithms [62]. Next to that, PPO is known to be quite robust to its hyperparameters, which makes its implementation easy. Finally, actor-critic algorithms are known to be sample inefficient, however, PPO employs a bounded update of its policy (which is discussed later in this section) that ensures that the algorithm will not make large adjustments to its policy, and therefore learns in a relatively stable manner.

3.2.3.3 PPO working principle

For the further implementation and tuning of an RL agent using PPO, on the environment, the way this algorithm works should be understood. Therefore, this subsection discusses the working principle and mathematical background of the PPO algorithm.

The goal of the PPO algorithm is to approximate π^* by training a NN. This NN is updated using the gradient of the policy, making it a Policy Gradient (PG) method [63]. However, there are a number of ways in which PPO improves upon normal PG methods. To update the policy network, PPO minimizes a so-called "surrogate loss function". A look at this surrogate loss function helps better understand how PPO improves:

$$L_t(\theta) := \widehat{\mathbb{E}}_t \left[L_t^{\mathrm{CLIP}}(\theta) - c^{\mathrm{VF}} L_t^{\mathrm{VF}}(\theta) + c^{\mathrm{H}} S \right] \tag{3.19}$$

where θ are the parameters of the policy (or actor) NN. The right-hand side of this function consists of 3 terms, each with a separate function. The first term L_t^{CLIP} is the clipped loss function of a general PG method. The goal of this clipping is to prevent the agent from changing too much when an update is applied. The second term is a term for promoting the loss function

of the Value function network (or actor-network), L_t^{VF}. This function is tuneable by the hyper-parameter c^{VF}. The third term is the so-called entropy bonus. This can be tuned by c^{H} and promotes randomness in the system. This term can be increased to increase the exploration of the agent.

Clipped loss function As was already seen in subsection 3.2.3, policy-based algorithms which use PG are prone to unstable learning. This is caused by the fact that, when updating its policy during learning, often too large steps are made. As this is undesirable in many cases, such as the mission-critical DC, PPO solved this by limiting the "motivation" for the algorithm to make large updates. To understand this, first, an understanding of how a general PG algorithm updates its policy is required:

$$L_t^{\text{PG}} := \widehat{\mathbb{E}}_t \left[r_t^{\pi}(\theta) \hat{A}_t \right] \tag{3.20}$$

where $r_t^{\pi}(\theta)$ is the ratio between the probability of this policy being taken under the new, updated policy and the old policy:

$$r_t^{\pi}(\theta) = \frac{\pi_{\theta}\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{3.21}$$

and $\hat{A}_t$ is the advantage of taking this action over the current optimal policy:

$$A^*\left(s_t, a_t\right) := Q^*\left(s_t, a_t\right) - V^*\left(s_t\right) \tag{3.22}$$

Where $Q^*\left(s_t, a_t\right)$ is the total discounted reward when action a_t is taken and the optimal policy is followed from the subsequent state. So, in summary, PG tries to minimize the ratio between the current policy and an updated policy, given the advantage of taking the current action against the expected optimal action. The important thing to notice here is that the advantage can change sign. So when the advantage is negative, the algorithm will try to minimize the probability of this action happening, while it will try to maximize it when the advantage is positive.

As the advantage can not be determined analytically, advantage estimations are used ($\hat{A}$). In PPO, the Generalized Advantage Estimator (GAE) is used, which is defined as [64]:

$$\hat{A}^{\text{GAE}}(\gamma, \lambda^{\text{GAE}}) := \sum_{l=0}^{\infty} (\gamma \lambda^{\text{GAE}})^l \delta_{t+l}^{\text{V}} \tag{3.23}$$

where γ is the reward discount factor of the RL framework; λ^{GAE} is a parameter which is used to balance between a bias on the accuracy of V ($\lambda^{\text{GAE}} = 0$) or high variance due to relying on future terms ($\lambda^{\text{GAE}} = 1$). Finally:

$$\delta_t^{\text{V}} := r_t + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \tag{3.24}$$

where V is the output of the value (or critic) network. This advantage estimator is used for the surrogate loss function of the actor and critic networks.

The PPO algorithm minimizes the size of the policy updates by clipping the loss function, where ϵ is a tuneable hyperparameter:

$$L_t^{\text{CLIP}}(\theta) := \widehat{\mathbb{E}}_t \left[\min\left(r_t^{\pi}(\theta) \hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t \right) \right] \tag{3.25}$$

This function puts a limit on how much the policy is changed when there is a very large advantage, or when there is a really bad advantage. This adds to the stability of the agent.

Value function loss The second term in Equation (3.19) is the loss of the value function, which is also used to update the value network. This is defined as:

$$L_t^{\mathrm{VF}}(\theta) := \left(V_\theta(s_t) - \widehat{V}_t\right)^2 \tag{3.26}$$

where $V_\theta(s_t)$ is the output of the value network, and $\widehat{V}_t$ is an estimation of the true value function:

$$\widehat{V}_t := \sum_{l=0}^{B} \gamma^l r_{t+l} \tag{3.27}$$

Using the combination of these loss functions, PPO thus ensures no large policy updates are made, which eventually leads to faster and more stable convergence [62].

Algorithm structure and sample collection As the PPO algorithm is an on-policy algorithm, it alternates between interacting with the environment to collect samples and optimizing the policy and value networks using the collected data. This is shown in Algorithm 1.

The outer `for` loop performs this iteration between collection and optimization for a certain amount of times. The data collection is done in the `for` loop in lines $2-5$. PPO can run in parallel environments, which could improve the exploration and decrease the correlation between samples. For every actor in the N environments, the agent interacts with the environment for T timesteps. When the data is collected, the policy and value networks are updated according to line 6, using the loss functions from Equation (3.19) and (3.26).

Algorithm 1 PPO, Actor-Critic Style (algorithm adopted from [61])

1: **for** iteration = 1, 2, ... **do**
2: **for** actor = 1, 2, ..., n^{envs} **do**
3: Run policy $\pi_{\theta_{\text{old}}}$ in environment for $n^{\text{trajectory}}$ timesteps
4: Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
5: **end for**
6: Optimize surrogate L w.r.t. θ, with n^{epochs} epochs and minibatch size $n^{\text{minibatch}} \leq n^{\text{envs}} n^{\text{trajectory}}$
7: $\theta_{\text{old}} \leftarrow \theta$
8: **end for**

Hyperparameters The equations and algorithm of the previous paragraphs contain a number of hyperparameters that can and have to be tuned during the implementation of the algorithm. Table 3.3 contains a summary of those hyper parameters.

The last 2 hyperparameters in the table have not been discussed in detail yet. These are the network architectures of the actor and critic networks. In the definitions above, π_θ and V_θ have only been defined as function approximators which are parameterized by θ. So any function approximator can be used for this. To narrow the scope of the thesis, these network architectures are limited to simple Multi-Layer Perceptron (MLP) NNs unless the implementation indicates that more sophisticated approximators are required.

Table 3.3: Summary of the PPO hyperparameters

Hyperparameter	Description
γ	The reward discount factor.
λ^{GAE}	Parameter to balance variance and bias in the GAE.
ϵ	The PPO clipping factor
c^{VF}	Coefficient to include value function loss in the policy loss.
c^{H}	Coefficient for the entropy bonus.
n^{envs}	Number of parallel environments.
$n^{\text{trajectory}}$	Number of trajectory timesteps collected per iteration.
n^{epochs}	Amount of training epochs for the policy and value approximators.
$n^{\text{minibatch}}$	The size of the minibatches for training.
α	The learning rate of the policy and value networks.
π_θ^{arch}	The architecture of the policy network.
V_θ^{arch}	The architecture of the value function network.

Figure 3.9: Timeline of the training, validation, and testing data.

3.3 Weather and IT load data and data partitioning

This section discusses the data that is used for the 2 disturbance variables and elaborates on how and why the data is partitioned into training, validation, and testing data. These variables are the reason the HVAC model is stochastic. To ensure the agent is robust to this stochasticity and performs well on unseen data, the data is partitioned into 3 categories, as shown in Figure 3.9. If it is not robust, the RL-based controller will not be able to effectively control the system. Firstly, weather and IT load data from $2018 - 2021$ is used for the training of the agent. To validate if a trained agent generalized well, or has overfitted on the training data, its performance is validated using a validation data set with 2022 data. This set is used for the validation during tuning of the agent. Finally, to test tuned agents on unbiased data, a 2023 test set is defined.

3.3.1 Weather data

For weather data, EnergyPlus employs weather files, containing information on the weather at a certain location throughout the year. The only factor of influence in this EnergyPlus model is the outdoor temperature, as this is the only variable the cooling tower model takes into account, and the rest of the system is adiabatic to the outside world. However, the complete weather files are still implemented. For the weather data in this thesis, the weather files of Amsterdam from $2018 - 2023$ have been retrieved from Oikolabs [65]. Amsterdam has been chosen as the Dutch climate is quite moderate, and Coolgradient has Dutch clients.

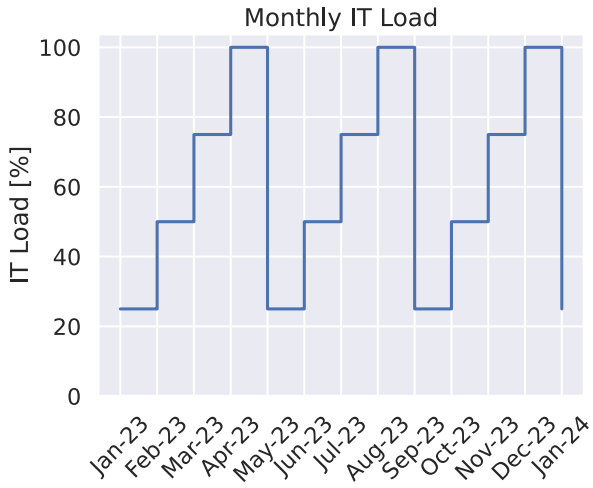


Figure 3.10: The IT Load profile as used in [44].

3.3.2 ITE load data

In the DC model in [44], a stepwise P^{ITE} is used which is continuous for a month before taking a sudden step to a higher P^{ITE}, as shown in Figure 3.10. This approach works well enough to get an insight into the total energy usage of a data center with varying levels of IT load over the course of a year. However, in the case of controlling a HVAC system using RL, the IT load should be more realistic on a smaller timescale.

This more realistic IT load is required for 2 reasons. The first reason is related to the fact that a controller should be robust enough to handle a varying, stochastic value of P^{ITE}. When the system is trained on this stepwise function, which is (almost) deterministic, it is not robust to the stochastic nature of real IT load variations. The other problem is related to the overfitting of the RL algorithms. To summarize, a more realistic IT load is required both for a more realistic simulation and for creating a more robust agent.

Real data of P^{ITE} is hard to find, as it is often part of DC companies Intellectual Property (IP). Therefore, as no real data could be found, the IT load data has been constructed synthetically. Yearly data has been created for each separate training, validation, and testing year. The goal of the yearly data is to represent the stochastic nature of the IT load, without it changing unrealistically fast. From conversations with experts from Coolgradient followed that P^{ITE} typically varies throughout the year, and also has a clear variation in the load from day to night. To adhere to this behaviour, the synthetic P^{ITE} is constructed from a base load that follows a random walk throughout the year, plus an influence of the time of the day. Noise is added as well. This leads to the following definition of P^{ITE}:

$$P^{\text{ITE}}(t) = P_{\text{base}}(t) + P_{\text{day,night}}(t) + P_{\text{noise}}(t) \tag{3.28}$$

The derivation of the terms in this equation can be found in Appendix B.

3.4 Reinforcement Learning tuning experiments

To fairly compare the behaviour of the RL agent to the baseline, it should be tuned to work optimally. This tuning is split up into 2 separate tuning experiments, both with a slightly different goal. The first type of experiment is hyperparameter tuning. In these experiments, the hyperparameters of the algorithm are tuned to ensure they are in an optimal configuration. Their method is given in subsection 3.4.1. The other type of tuning experiments are the reward tuning experiments. In these experiments, the weights and other parameters in the reward function of Equation (3.9) are tuned. This is done to investigate the effects of changing these parameters on the control behaviour of the agent. Using the results of these experiments, a reward that makes a trained RL controller outperform the baseline is chosen. The method of these experiments is discussed in subsection 3.4.2

Justification for reward and algorithm tuning split It is important to realise that the hyperparameters of the RL agent and the reward function itself influence each other. Therefore, splitting the experiments will not lead strictly to the best-performing agent. However, there are multiple important reasons why this choice has been made. Firstly and most importantly, changing the parameters of the reward makes it hard to compare 2 different experiments, as they will have a different total reward by definition. Secondly, the computational cost of hyperparameter tuning experiments increases roughly exponentially with an increasing number of parameters to tune [66]. Therefore, splitting the experiments into separate experiments reduces their computation time. Finally, it is expected that, for different rewards, roughly the same hyperparameters will lead to the best results, as the environment itself does not change drastically. For example, if a certain policy network architecture would lead to a good reward for one reward setting, it is also probably able to learn good behaviour for another reward setting.

It is believed that the splitting of experiments is justified, because of the reasons mentioned above. However, the split is of course a limitation to the final results. Therefore, one should see the goal of the experiments not as much as finding the optimal configuration, but more as finding a good configuration of hyperparameters, which leads to good training, and then using this configuration to get a better understanding of the influence of the parameters in the reward function.

3.4.1 Hyperparameter tuning

In these experiments, the hyperparameter tuning of the RL agent is performed. Hyperparameter tuning is a common practice in ML algorithms to improve performance by adjusting parameters such as the learning rate and network architecture. However, for RL algorithms, hyperparameter tuning is less common. This is mainly due to the dynamic nature of RL, compared to the static datasets typically used in ML. The dynamic environment of RL makes hyperparameter tuning more challenging. Despite these challenges, recent research suggests that hyperparameter tuning can significantly enhance the performance of RL algorithms [67]. Therefore, this research performs hyperparameter tuning for the designed RL framework, to find a close to optimal configuration of the hyperparameters.

For hyperparameter tuning experiments, there are a number of important choices: which parameters to tune, on which ranges to tune them, and what kind of sampler to use for the hy-

perparameter configurations. This section discusses these choices.

Hyperparameter selection The hyperparameters that are tuned are the hyperparameters related to the PPO algorithms. Initially, the default ranges are searched. The network architecture of the policy and value function approximators are split into the number of layers and the number of neurons per layer.

Sampler selection One of the most important choices is deciding which sampler to use. As hyperparameter optimization trains and tests a RL agent for each different hyperparameter configuration, it is a very computationally expensive type of optimization. Therefore, it is very important to have good sample efficiency. This is further complicated by the fact that the amount of samples increases with an increasing number of parameters that should be optimized.

There exists a broad range of samplers, each with its own advantages and disadvantages, in Appendix C, a number of the most common samplers and their (dis)advantages are discussed in more detail.

For these experiments, the Tree-structured Parzen Estimator (TPE) sampler will be employed. This is a Bayesian sampler, which is known to be sample efficient in large hyperparameter spaces, as it has the advantages of the good system exploration of pseudo-random samplers, and being able to adapt to intermediate results. Bayesian samplers work by constructing a surrogate model, often using Gaussian processes. These models predict the performance of the hyperparameter configurations and give an estimate of the uncertainty of the surrogate model. By focussing on promising areas, a lot more sampling can be done in these areas. This is very beneficial for the sample efficiency, and thus the computational complexity. A disadvantage of Bayesian sampling is that it can focus on suboptimal regions when the initial surrogate model is very inaccurate. However, as the model updates with each new sample, this is often not a large problem in reality [68]. The reason for choosing the TPE sampler specifically is because it is incorporated in Optuna [69], which is the Python library used for the hyperparameter tuning.

Experiment outline The experiments will train the RL agent on the EnergyPlus simulation environment, cycling through the training data of the disturbances, for 10 episodes for each hyperparameter configuration. Then, the trained controller will be evaluated on the validation disturbance data. Each hyperparameter configuration will use the same reward and the total reward of the validation run will be used to compare the performance of agents.

The experiments start with a small hyperparameter study of 40 samples. The goal of this study is to validate the hyperparameter ranges. If this study already shows that the RL agent performs a lot better in certain hyperparameter ranges, or the best results are found at the limit of the range for a certain parameter, the ranges will be adjusted accordingly.

After these initial experiments, a large hyperparameter study starting with 200 samples will be performed. The results will be analyzed, and if it is required, additional experiments with either all hyperparameters or a subset of hyperparameters will be performed.

3.4.2 Reward tuning

The goal of the reward tuning experiments is to investigate the effects of changing the parameters in the reward function on the behaviour of the agent. From this investigation, settings for the reward function which leads to the best behaviour can be chosen.

The reward tuning experiments are an optimization problem. However, while it seems similar, this optimization problem should not be confused with the general optimization problem from this thesis from Equation (2.23). The goal of that minimization problem is the ultimate goal of the HVAC system controller, which is to minimize P^{HVAC} while meeting a number of constraints. So it is a single objective, constrained minimization problem. The subtle difference is that the reward tuning should find a reward that leads to behaviour that minimizes P^{HVAC} *and* minimizes the number of constraint violations. This makes the reward tuning the following multi-objective, unconstrained minimization problem:

$$\min_{\boldsymbol{\alpha} \in \mathbf{A}} \left(P^{\text{HVAC}}(\boldsymbol{\alpha}), f_{T_{\text{SP}}^{\text{AL1}}}(\boldsymbol{\alpha}), f_{T_{\text{constr}}^{\text{AL2}}}(\boldsymbol{\alpha}), f_{a_{\text{fluct}}}(\boldsymbol{\alpha}) \right) \tag{3.29}$$

where $\boldsymbol{\alpha}$ is the vector of reward tuning hyperparameters (not to be confused with α, which is the learning rate of the neural networks). It should be an element of the hyperparameter space $\mathbf{A}$. $P^{\text{HVAC}}(\boldsymbol{\alpha})$ is the total yearly HVAC power as a function of the hyperparameters, and $f_x(\boldsymbol{\alpha})$ are functions for measuring the severity of the constraint violations or undesired behaviour over the course of a year. These 3 different functions are based on the 3 types of penalties in the reward function. The remainder of this section will focus first on defining these functions. After that, the sampling from the hyperparameter space will be discussed.

Constraint violation metrics To evaluate the agent's performance over the course of a year, specific metrics will be used to measure the severity of the following three types of unwanted effects in the system:

- The deviation from the setpoint at point AL1.
- The violation of the temperature constraint at point AL2.
- The fluctuation of the action variables.

To provide a measurement of the unwanted effects over the course of a whole year, it is desirable to couple the effect to a single number for each of the 3 effects. Therefore, the severity of these effects is measured by the Euclidean distance, or L_2-norm:

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{t=1}^{T} x_t^2} \tag{3.30}$$

where T is the number of timesteps in a year, and x_t is a measure of each specific unwanted effect. This norm is chosen as it gives more weight to larger constraint violations, while still penalizing small violations. This is in line with the requirements of the system. Next to the L_2-norm, the L_∞-norm (the maximum value of x_t in a year) is also logged for analysis purposes.

To start, the deviation of T^{AL1} from $T_{\text{SP}}^{\text{AL1}}$ should be minimized both when the temperature is

higher and lower than the setpoint. Therefore, the L_2-norm of the setpoint deviation is defined as:

$$f_{T_{\text{SP}}^{\text{AL1}}}(\boldsymbol{\alpha}) = \sqrt{\sum_{t=1}^{T} \left(T_t^{\text{AL1}} - T_{\text{SP}}^{\text{AL1}}\right)^2} \tag{3.31}$$

For the violation of $T_{\text{constr}}^{\text{AL2}}$, it only should be taken into account how much higher T^{AL2} is than the constraint temperature. Therefore, the following definition of the L_2-norm is used in this case:

$$f_{T_{\text{constr}}^{\text{AL2}}}(\boldsymbol{\alpha}) = \sqrt{\sum_{t=1}^{T} \left(\max\left(0, T_{\text{constr}}^{\text{AL2}} - T_t^{\text{AL2}}\right)\right)^2} \tag{3.32}$$

Finally, for the action fluctuation, the agent must not have large, high-frequency oscillations. Therefore, the difference between the current and previous action is taken as the constraint measure:

$$f_{a_{\text{fluct}}}(\boldsymbol{\alpha}) = \sqrt{\sum_{t=1}^{T} \left(a_t - a_{t-1}\right)^2} \tag{3.33}$$

This finalizes the definition of the minimization problem in the reward tuning experiments. The next section will focus on the next step in these experiments: the sampler selection.

Sampler selection This study uses the Bayesian TPE sampler as well, as the reward hyperparameter space is also high dimensional, and the adjustment to intermediate results is even more desirable in the multi-objective optimization which is performed in these experiments.

Opposed to the single-objective optimization, where the sampler tries to sample hyperparameters which lead to a better reward, the multi-objective TPE tries to sample hyperparameters on the so-called Pareto front. This front consists of all non-dominated solutions. A configuration is non-dominated if it leads to behaviour for which there is no other configuration that outperforms the first configuration on all optimization metrics. An example of a Pareto front is given in Figure 3.11.

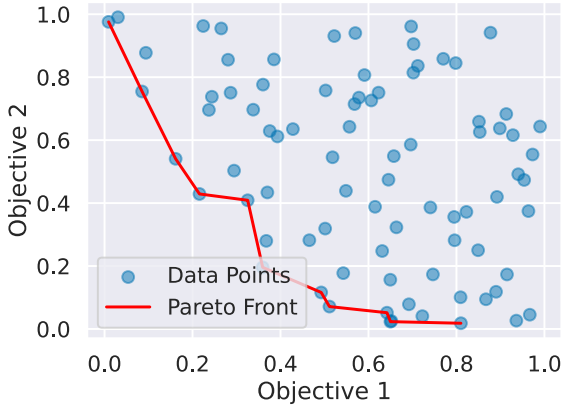


Figure 3.11: Example of a Pareto front in 2 dimensions.

Experiment outline The goal of the reward tuning experiments is to form a Pareto front of the 4 performance metrics. To achieve this, the agent is trained with different hyperparameter configurations for 200 trials. These configurations are sampled using the TPE sampler. Each configuration will again be trained on 10 episodes using the training disturbance data and is evaluated on the validation disturbance data.

The algorithm hyperparameters that are found in the algorithm hyperparameter tuning experiments are used for all reward configurations. The results of the reward tuning experiments will be compared to the performance of the baseline controller. Depending on the outcomes, additional tuning experiments may be performed to further investigate specific phenomena observed in the results.

3.5 Summary

This chapter details the methodology that will be used to create a RL-based controller for the HVAC system defined in chapter 2.

First, a baseline controller has been defined, which is based on static temperature setpoints and bases its mass flow rates on the energy balance.

After that, a framework for the RL agents has been created. The HVAC system has been described as an MDP, and a tuneable reward function has been proposed, which has penalties for violation of constraints. The chosen algorithm is PPO.

To ensure that the trained agents are robust controllers, that are not overfitted to the disturbance variables, data of the disturbance variables is partitioned into training, validation, and testing data. For the outdoor temperature, historical data from Amsterdam is used. For P^{ITE} a load profile is constructed using a random walk, day and night effects, and noise.

Finally, 2 types of hyperparameter tuning experiments are proposed to actually train and tune the RL-based controller. The first type is algorithm hyperparameter tuning, which uses a Bayesian sampler to find well-performing hyperparameters of the PPO algorithm. Afterward, the hyperparameters in the reward function will be tuned. The goal of these experiments is to gain insights into the effects of tuning reward parameters on the performance of the controller. These insights can be used to find an optimal reward that balances the performance of a tuned controller with respect to reducing both P^{HVAC} and constraint violations.

The upcoming chapters first implement the simulation model, baseline controller, and RL framework into a Python framework. Afterward, the numerical results of the experiments outlined in this chapter are presented.

Chapter 4

Numerical results

In chapter 2, a HVAC system has been defined and a control optimization problem has been constructed for this system. Following this, chapter 3 converted this system into a simulation model, designed a conventional baseline controller, and proposed a RL framework which can be used to improve upon this baseline controller.

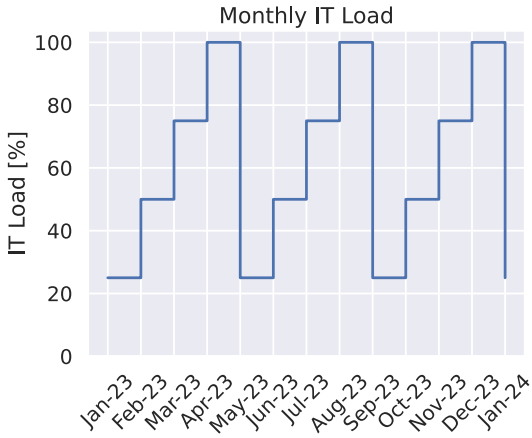
In this chapter, the results of the experiments which were outlined in those chapters are presented and discussed. In section 4.1, the sensitivity of the simulation to changes in the timestep is determined. With the desired timestep set, control experiments have been conducted. First, section 4.2 discusses the general Python implementation of the EnergyPlus model for control purposes. After that, the results of the baseline experiments are discussed in section 4.3. After the baseline has been set, initial RL experiments have been performed to get to know the system, spot any problems at an early stage, and get an initial guess of the best hyperparameters. The results of these experiments are given in section 4.4. With this initial estimation of the hyperparameters, the tuning experiments could be conducted. The hyperparameter tuning results are given in section 4.5, while the reward tuning results are shown in section 4.6. The results of these experiments indicated that the framework could be sensitive to its initialization. Therefore, a seed sensitivity study is performed in section 4.7. After these experiments have resulted in several well-tuned RL agents, their performance during training, validation, and testing is compared to the performance of a baseline agent in section 4.8.

4.1 Timestep sensitivity study results

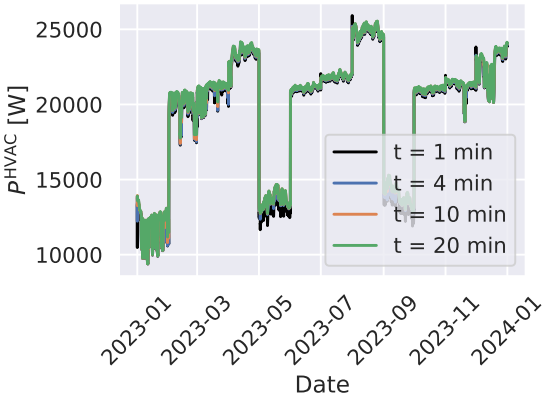
This section provides the results of the timestep sensitivity analysis, as outlined in subsection 2.3.3. The goal of these experiments is to analyze how much the timestep affects the outcome of the model and, based on this, select a suitable timestep for the remainder of the experiments in the thesis.

4.1.1 Data filtering

In Figure 4.1b, the simulation results of P^{HVAC} over the course of a year are shown for 4 different timestep sizes (only a selection of timestep sizes is shown for clarity of the figure). As can be seen, the value of the power takes a large jump every month, which corresponds to the stepwise changing P^{ITE} used in this study (Figure 4.1a). Generally, such a large step would not affect the results of a timestep study. However, Figure 4.2a shows large peaks in the difference between the HVAC power of these 4 timestep sizes compared to the HVAC power using the

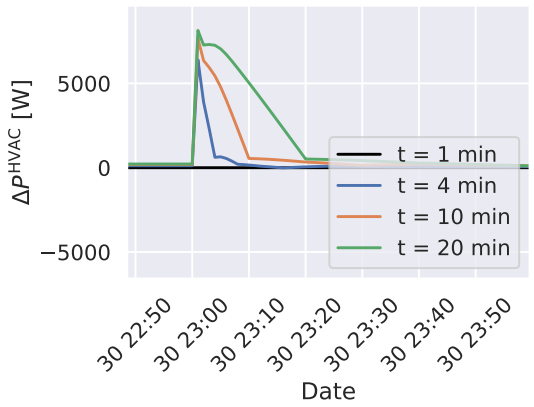


(a) Monthly stepwise P^{ITE} (also shown in Figure 3.10, restated here for convenience)

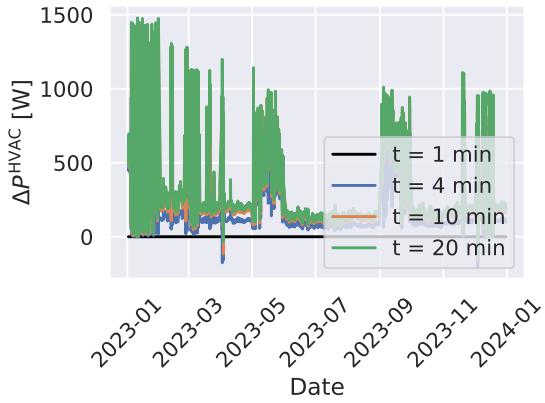


(b) P^{HVAC} over the course of a year, for 4 different timestep sizes.

Figure 4.1: The ITE load and HVAC power over the course of a year.



(a) The difference for P^{HVAC}, zoomed in on a change of month.



(b) The filtered difference between P^{HVAC} and the simulation with the smallest timestep.

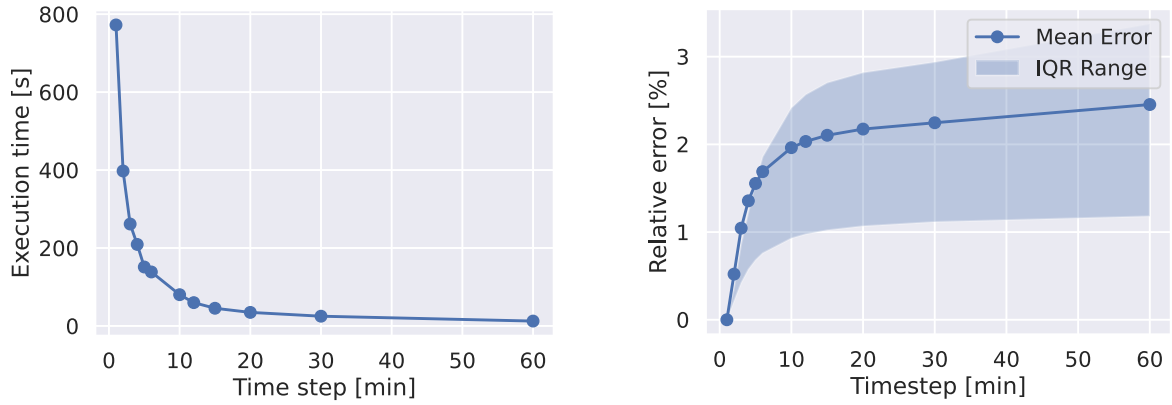
Figure 4.2

smallest timestep (more than 5 kW difference). As the peaks get wider for increasing timestep size, this is probably related to the controller of the system, and not the simulation itself.

To mitigate this problem, the days on which these external conditions change rapidly are simply filtered out of the data. Figure 4.2b shows the difference between the 4 timestep sizes to the smallest timestep for P^{HVAC}, where the first and last day of each month have been filtered out. The large peaks are now no longer present in the data. Since only 2 days per month are neglected, this is an acceptable loss of data for the rest of this study.

4.1.2 Effects on the simulation

For the effects of the timestep on the accuracy of the simulation, its effect on the execution time and the mean error over a year of 3 important parameters (the total HVAC power, a temper-



(a) The execution time against t^{step}.

(b) The relative error of P^{HVAC} against t^{step}.

Figure 4.3: Timestep study results for the execution time and P^{HVAC}.

ature, and a mass flow) has been calculated. Additionally, to indicate the distribution of the error, the Inter Quartile Range (IQR) (which is the range between the first and 3rd quartile of all error values in the year) has been plotted as well.

Execution time The effect of the size of t^{step} on the total execution time of the simulation is shown in Figure 4.3a. The total execution time of a single year of simulation time increases rapidly with decreasing timestep size. This has to be kept in mind while selecting a timestep size. Another important note is that this is solely the EnergyPlus execution time. Training an RL agent will produce additional overhead for each timestep.

HVAC power The first parameter for which the mean of the filtered error over a year has been calculated is P^{HVAC}. The results are presented in Figure 4.3b. As can be seen, the mean error increases with increasing t^{step}, although it starts to increase less quickly for $t^{\text{step}} > 10$ minutes. The maximum mean error is at around 2.5%, which is quite an acceptable value since the simulation of both the baseline and the RL agent are performed with the same timestep.

The IQR range is quite close to the mean error, which indicates that the error is not too widely distributed. For the smaller timesteps, however, even the 3^{rd} quartile is below the mean error. This indicates that the mean is affected by some large outliers. However, for these small timesteps, the error is at or below 1%, which is still deemed acceptable.

Temperature The second variable for which the error has been determined is T^{ChL2}. For this variable, a temperature without a setpoint has been chosen to avoid the controller affecting the simulation results too much. In Figure 4.4a can be seen that the error is symmetrically distributed above and below zero for all timesteps. The error is negligibly small for all timestep sizes.

Mass flow In Figure 4.4b, the mean errors $\dot{m}^{\text{ChL}}$, the third variable to be considered, is shown. Except for the outlier at $t^{\text{step}} = 60$, the mean error stays relatively small. The distribution of the error increases slightly with increasing timesteps. However, this is all within 0.5% of the total mass flow.

(a) The relative error of T^{ChL2} against t^{step}.

(b) The relative error of $\dot{m}^{\text{ChL}}$ against t^{step}.

Figure 4.4: The timestep study results for the temperature and mass flow.

The high mean error at $t^{\text{step}} = 60$ is probably caused by a small number of data points where there was an outlying, large error, as the IQR is still quite comparable to the rest of the timesteps. This has not been further investigated, as a timestep of 60 minutes is too large for control effectiveness. This is discussed in more detail at the end of this section.

4.1.3 Discussion & conclusion on the timestep sensitivity

The mean error and the error distribution for all 3 important parameters that have been taken into account are, with only a few percent, relatively small. Although there are indications that there are quite large outliers in the error over the course of a year, overall, the simulation does not seem to be very sensitive to the timestep size.

What is very sensitive to the timestep size, however, is the time the simulation takes. This is found to increase linearly with the number of timesteps in a year, as was expected. It is also important to note that these simulation times were just for an EnergyPlus simulation. When the simulation also has to interact with Gym and models have to be trained in the RL experiments, the simulation times are expected to increase.

Another factor of concern is the comparability to reality. As in reality, often data is collected every 10 minutes, this would seem a logical timestep to use for the simulations as well. A way larger timestep would let the controller react too slowly, while a much smaller timestep may let the controller react way too fast. In Figure 4.3a, it can be seen that the EnergyPlus simulations take 50 seconds with this timestep. Also, the mean errors and IQR of the error are all acceptable for this timestep size. Therefore, a timestep of **10 minutes** is used in the remainder of the experiments.

4.2 Python implementation

For the Python implementation, the OpenAI Gym [70] library has been used as an interface between a controller and the (EnergyPlus) environment. The Gym library has a standard Appli-

cation Programming Interface (API) for the communication between agents and environments. Since this is the standard interface that is used in RL, many off-the-shelf RL libraries can be used with Gym. Gym provides a set of standard environments that can be used for easy development and benchmarking of new algorithms. It also provides the ability to create custom environments, as has been done in this study. Although the Gym API is thus mainly meant for RL, other controllers, such as the baseline, can also be easily employed on this system, as it simply provides a feedback loop.

The Gym API has some key methods: `__init__()`, which initializes the environment and sets the correct state and action space; `step()`, which performs one simulation step; `reset()`, which resets the environment after a training episode has ended and `close()`, which closes the environment after the simulation.

The custom environment used in this study should be able to receive and send information from and to EnergyPlus between every simulation step on one end, and communicate with a controller according to the Gym API on the other end. This is visualized in Figure 4.5, which expands the environment of Figure 3.4 into the custom Gym environment and the EnergyPlus model. For the creation of the custom environment, the testbed from [45] has been modified to simulate the HVAC model of this thesis. This testbed consists of a Gym environment that is connected to a patched version of EnergyPlus. This allows the Python code to send the action a_t to EnergyPlus, perform one timestep of simulation, and then receive the states, s_{t+1}, back from the model. This is then integrated into the Gym API. The modification of this testbed consists of 2 steps: changing the Input Data File (IDF) of the EnergyPlus model to allow the Python code to send and receive the state and action variables, and the creation of a model-specific class which handles the creation of the state and action spaces and reward computation in Python. This is done according to the instructions provided in [45]. Further details of this implementation are not discussed here to keep the report (a bit more) comprehensive.

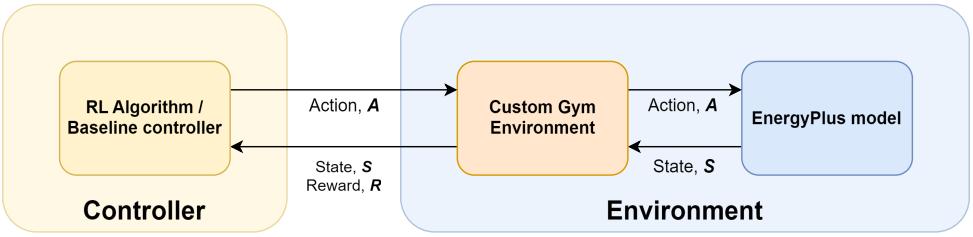


Figure 4.5: The RL framework for the Python implementation.

For most simulations in this study, the author's laptop has been used, while for the tuning experiments, a VM has been used. Their specifications are summarized in Table 4.1.

4.3 Baseline study results

Using the timestep size which has been determined in section 4.1, the baseline experiments have been performed according to the method in section 3.1. The settings presented in Table 4.2 are used for these experiments. In this section, first, a correct value of T_{SP}^{ChL1} is selected. Using this configuration, the baseline experiments are performed. First, the yearly results are analyzed. Afterward, a more in-depth analysis of the component behaviour is performed to understand how the baseline controller makes the HVAC system behave.

Table 4.1: Specifications for the hardware used in this study.

Component	Laptop	Virtual Machine
Device Name	ThinkPad P51 Signature Edition	-
Processor	Intel Core i7-7700HQ, 4 cores, 8 threads, 2.80 GHz base, 3.80 GHz boost	2 vCPUs
RAM	16 GB DDR4	16 GiB
Storage	1 TB SSD	OS Disk: 1 TB
Operating System	Windows 10 Home 64-bit, Version 22H2, OS Build 19045.4651	Linux
Location	-	Sweden Central (Zone 1)

Table 4.2: Settings of the conventional controller in the baseline experiments.

Setpoint	Value	Justification
T^{AL1}	24°C	This is the typical underfloor temperature setpoint in DCs, which is right between the upper and lower limits of 21 and 27°C, as determined in section 2.4.
$T^{\text{AL2}}_{\text{constr}}$	30°C	This setpoint is set to prevent the air leaving the servers from becoming too hot and exceeding the constraint temperature. This temperature has been determined in section 2.4 and is typical for the hot aisle temperature.
ϕ^{AL4}	0.5	A relative humidity of 50% is standard in server rooms [52].
T^{ChL1}	$18, \ldots, 23$°C	Experiments with varying chilled water temperature setpoints are performed. This is done to compare the effect of different chilled water temperatures over the course of a year with each other. The best performing setting is compared to the RL agent.
$T^{\text{CoL1}} - T^{\text{outdoor}}$	1.5°C	This is the standard value of this setpoint in EnergyPlus. To minimize the variables that are changed, this is kept at the standard value.

4.3.1 Chilled water setpoint selection

Simulations using a range of fixed values for $T^{\text{ChL1}}_{\text{SP}}$ in the baseline controller have been performed, with as goal to select the best-performing value for the setpoint regarding P^{HVAC}. The controller determined the mass flows based on the equations outlined in subsection 3.1.1. These simulations have been performed on the test dataset, so the weather data from 2023 and `IT_Load_5.csv`.

Figure 4.6 shows the total yearly energy usage of the HVAC model using the baseline controller for various values of the chilled water setpoint. The controller becomes more efficient when the setpoint temperature increases until it becomes too close to the air temperature and the efficiency rapidly decreases. The initial decrease is likely caused by a higher efficiency of the

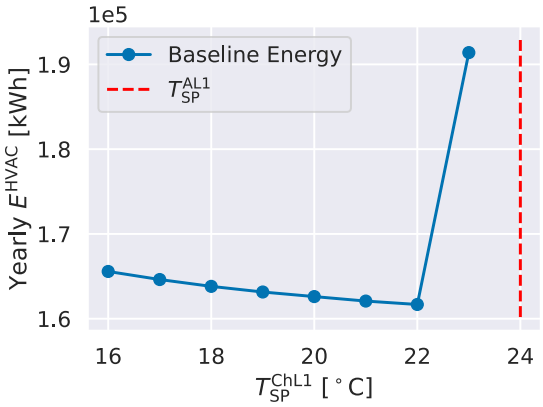


Figure 4.6: The energy consumption of the baseline controller for various values of T_{SP}^{ChL1}.

Table 4.3: Performance metrics of the baseline.

Physical Quantity	Norm Type	Variable	2018	2019	2020	2021	2022	2023
HVAC Energy	Total	P^{HVAC} [MWh]	157	165	164	180	173	162
Temperature Setpoint Deviation	2-norm	T^{AL1} [°C]	184	196	263	345	212	201
Temperature Constraint Compliance	2-norm	T^{AL2} [°C]	28.5	27.1	40.1	79.9	38.0	48.4
Action Fluctuation	2-norm	T_{SP}^{ChL1} [°C]	0	0	0	0	0	0

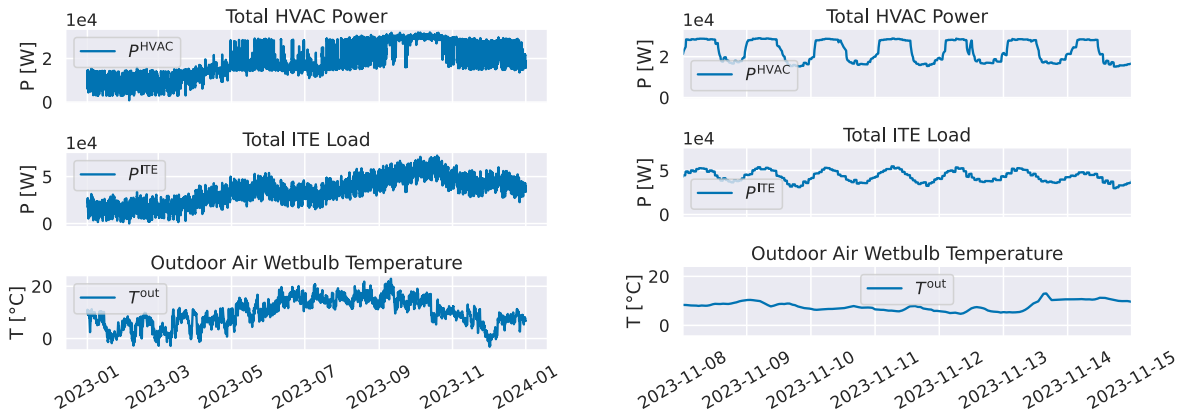
chiller at higher temperatures. When the setpoint increases, however, the mass flow rates also increase which likely causes the rapid increase of the power required. For a fair comparison to the RL agent, the baseline controller with $T_{SP}^{ChL1} = 22°C$ is selected.

4.3.2 Yearly performance

In Table 4.3, the performance of the baseline regarding P^{HVAC} and how well it satisfied the constraints, based on the metrics from subsection 3.4.2 is shown for all years of which data is used. This performance will be compared to the performance of the tuned RL agent, where the goal is of course to score lower on all parameters, and especially the HVAC power and temperature constraint compliance. In the remainder of this section, only the data of 2023 is considered.

It is interesting to see in this table that the baseline is also not fully capable of complying with the temperature constraint of T^{AL2}, as the 2-norm is still 44.6°C over the course of 2023. To understand better how much the constraints are violated, the infinity-norm for this parameter is checked, which is: $L_\infty = 1.66°C$. So the maximum constraint violation is only relatively small.

Relation HVAC power to disturbances To better understand what causes the energy usage in the system when using the baseline controller, Figure 4.7a shows P^{HVAC} and the disturbance



(a) The HVAC Power and disturbance variables over the course of a year.

(b) The HVAC Power and disturbance variables over the course of a week.

Figure 4.7: Overview of the total HVAC Power.

variables over the course of a year. As can be seen, the HVAC power is low when the ITE load and outdoor temperature are low, which is as expected. The apparent noise in the system is mainly caused by the variation in the load between day and night, as can be seen clearly in Figure 4.7b which shows the simulation results zoomed in to a single week.

From Figure 4.7b, it looks like P^{HVAC} is mainly proportional to P^{ITE} while it does not respond much to T^{outdoor}. This could be explained by how the baseline determines the mass flow in the loops in Equation (3.1) and Equation (3.2). The mass flow (and thus pump and fan power) mainly depends on P^{ITE}.

To further investigate the relationship between these variables, a contour plot has been created in Figure 4.8. This plot shows the relation between these 3 parameters for all timesteps. This plot shows that the HVAC power indeed mainly increases with increasing P^{ITE}, and slightly with increasing T^{outdoor}. This is especially visible when the ITE load is already high, as the contour line is under a steeper angle there, indicating the power depends on both variables. Another interesting observation is that there is a rapid change in HVAC power at $P^{\text{ITE}} \approx 40$kW, where the contour lines are close together. Potential causes for this are investigated in the component-level analysis. Note that this thesis uses the wet bulb temperature as the outdoor temperature, which is a better measure for heat exchangers. This is generally lower than the more common dry bulb temperature, therefore the maximum temperature in the year is only 23°C.

To conclude, the baseline controller performs quite well regarding the constraints. Next to that, the main influence on the HVAC power is P^{ITE} with a smaller influence of T^{outdoor}.

4.3.3 Component-level results

To further investigate the behaviour of the baseline controller and which components use the most power, the performance is compared component-wise. Figure 4.9 shows the power of each individual HVAC component compared to the ITE load. As the data originally was quite noisy, due to the dynamic behaviour of the system, a moving average is used. Several conclu-

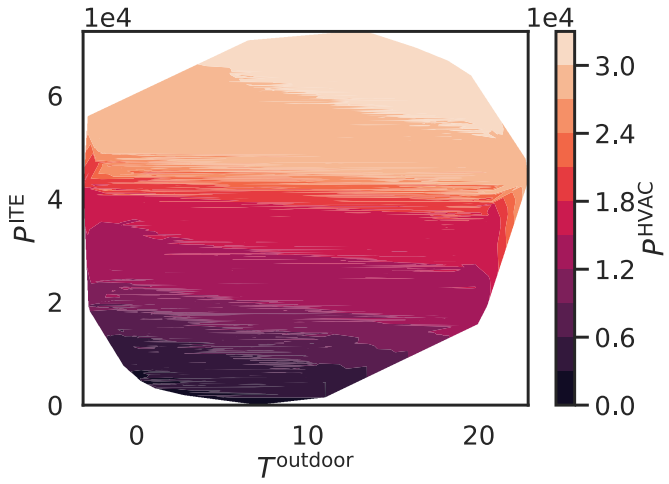


Figure 4.8: Contour plot relating the IT load and outdoor temperature to the HVAC power in the baseline study.

sions can be drawn from this plot:

- The chiller and air fan use the highest power. The chiller power increases gradually with increasing P^{ITE}. The air fan power increases quickly, according to the third order relation between this power and the fan mass flow from Equation (2.5), until it hits a limit at $P^{ITE} \approx 40$ kW. This is probably caused by the fan reaching its maximum mass flow.
- Both the cooling tower and cooling tower pump have almost continuous power usage, except for when the IT load is very low. This indicates that they are running at full power for most IT loads.
- The power of the humidifier is not even visible in this plot, it is therefore negligible.

Now that the ratios of the power of the different components compared to each other are known, we can look at how the power used by each of these components relates to both disturbance variables. Figure 4.10 contains contour plots for each component that show this relationship. Several conclusions can be drawn from this:

- Only in extreme cases, where either the IT load is very low (small amount of energy to extract) or where the outdoor temperature is low (large temperature difference), the Cooling Tower can switch off part of the time. This indicates that there is little energy to be saved here by using smarter settings of the cooling tower.
- The chiller is dependent on both the disturbance variables, and increases linearly when these variables increase. This is expected, as the chiller power depends on its required cooling capacity, which is determined by P^{ITE} and its COP, which depends on T^{CoL1}.
- The ChL pump power depends solely on the IT load, which is as expected as it mainly depends on its mass flow, and in the baseline controller, the mass flow solely depends on the IT load as long as the temperature setpoints are met.

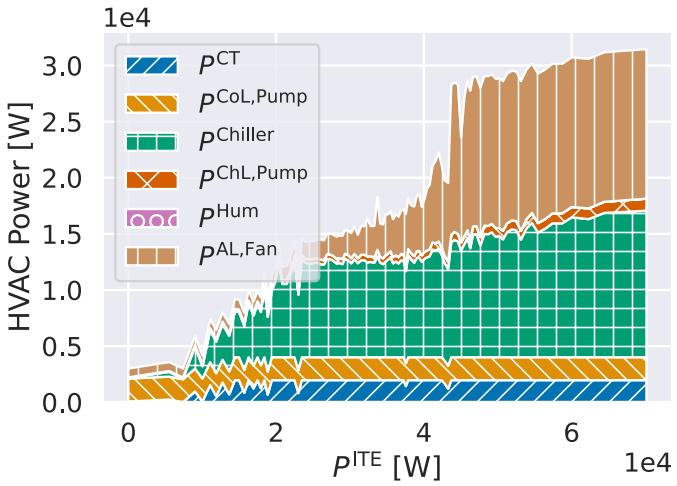


Figure 4.9: The filtered HVAC power per component against the ITE load.

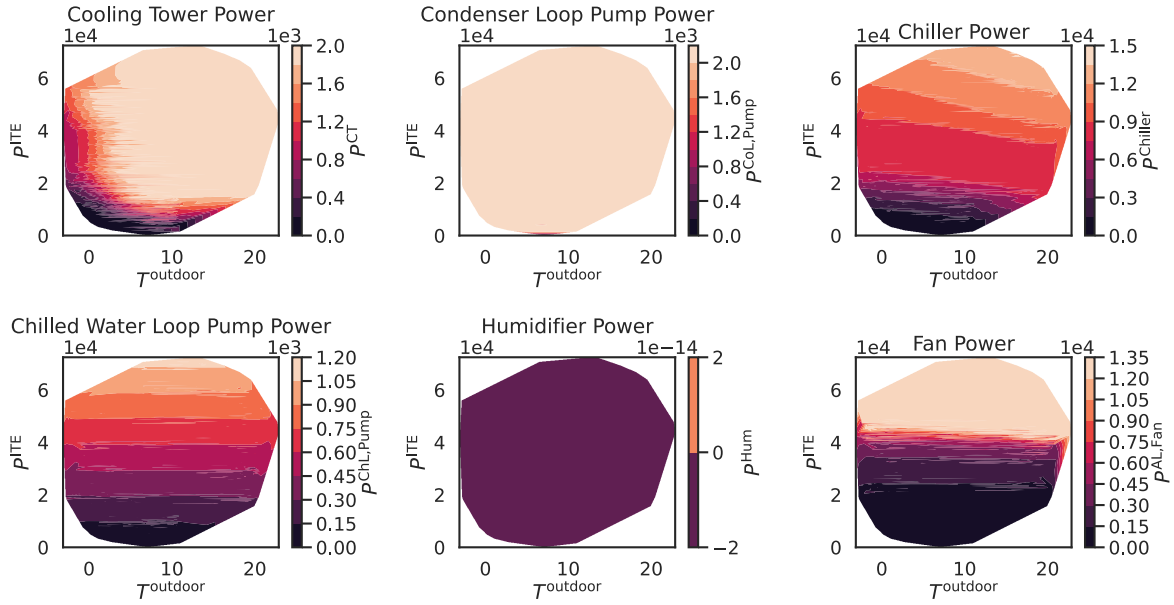
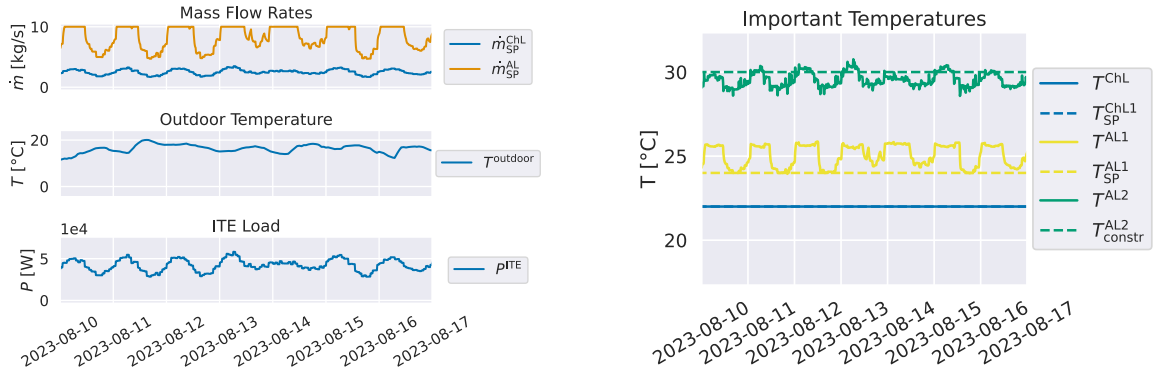


Figure 4.10: Contour plots showing how the power of each HVAC component related to the disturbance variables

- Similarly, the fan power depends only on the IT load. Next to that, it corresponds with the findings from Figure 4.9 and indicates that the maximum mass flow is reached when the IT load is this high.

In Figure 4.11, the behaviour of the control variables, disturbance variables, and several important temperatures for a week. In Figure 4.11a, it can be seen that just as expected, both mass flows (the control variables) are adjusted proportional to mainly the ITE power. However, $\dot{m}^{AL}$ reached its limit when P^{ITE} becomes too high. In Figure 4.11b it can be seen that the air loop is then also not able to maintain its setpoints and meet the constraints. This is thus also the cause



(a) The action and disturbance variables.

(b) Important temperatures

Figure 4.11: The behaviour of the system during a week where the maximum airflow is reached.

of the constraint violations by the baseline. The baseline can not adjust enough to this situation to solve the problem. This could be done by e.g. increasing $\dot{m}^{\text{ChL}}$.

4.3.4 Discussion & conclusion on the baseline controller

The analysis of the baseline controller's performance provides several key insights into the behaviour of the HVAC system under the baseline controller.

Firstly, the power consumed by the HVAC system mainly depends on the IT load and only has a minor dependence on the outside temperature. This is as expected, as the mass flows of the ChL and AL are directly proportional to this power in the baseline controller. However, the effect of the outdoor temperature, especially on the chiller, was expected to be slightly higher.

Secondly, the baseline controller can effectively maintain temperature constraints under relatively low IT load, however, when this increases, the controller reaches the limits of the equipment and is not able to adjust to it, leading to exceeding the constraint temperature and deviating from the air setpoint temperature.

One might say that the equipment is just badly scaled and the baseline agent will perform better if the equipment has a larger range, and this is indeed true. However, this bad scaling of equipment is something that also happens in reality, and it will be interesting to see how the RL agents will control the system under these conditions, and if they can find solutions for this problem. Problems like this are actually one of the reasons "smarter" controllers such as RL agents could improve the efficiency of the HVAC system.

4.4 Initial Reinforcement Learning experiment results

Now that the baseline controller has been implemented, the RL agents can be trained and tuned. Before the tuning experiments started, the framework has been implemented in Python and several initial experiments were performed to get an intuitive understanding and filter out anything that causes problems for the learning of the agent.

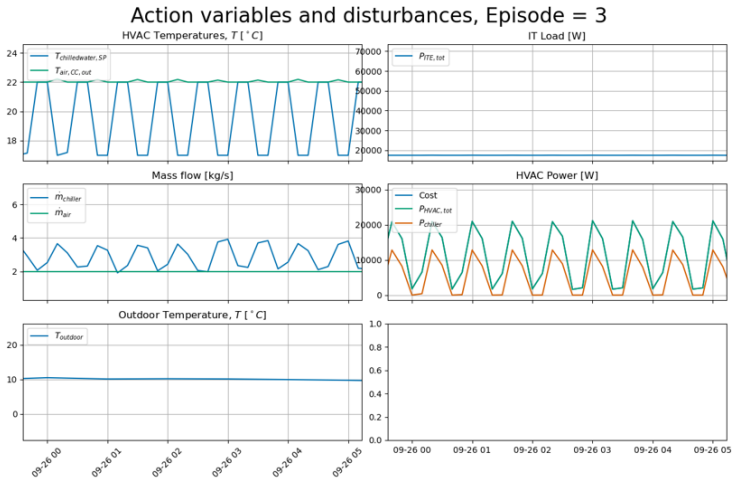


Figure 4.12: High-frequency oscillations of the action variables, leading to peaks in HVAC power

Python implementation There are a lot of different libraries which offer off the shelf RL algorithms that work with Gym, such as e.g., Ray RLLib [71], Stable Baselines 3 [72], KerasRL [73] or RL Coach [74]. These libraries all offer standard PPO algorithms, so the choice mainly depended on the ease of use, documentation and ability to modify the code of the library if that would be necessary.

The library of choice is Stable Baselines 3, as it has algorithm-specific agent classes, which make it very easy to implement, has good documentation and clearly commented code, and provides customizable callbacks that can be used for customization such as normalization of inputs or logging variables.

Initial experiments To keep this thesis comprehensive, only the most important findings of these experiments are summarized here.

- The actions and states should be **normalized** between 0 and 1. This is achieved by scaling the actions and states between the limits provided by the action and state space.
- Using several different hyperparameters and reward settings, it was found that the agents had always converged after **10 episodes**. Therefore, this number of episodes will be used for the tuning experiments. The convergence of the reward will be monitored during these experiments, such that the number of episodes can be adjusted if required.
- If the value function network was **too large**, the agents did not converge. This has been taken into account by limiting the amount of neurons per layer to 64.
- When the penalty on the action fluctuation was not present, high-frequency oscillations in the actions (and thus HVAC power) occurred. This is shown in Figure 4.12.

The algorithm hyperparameter tuning and reward hyperparameter tuning experiments have been set up using these insights.

Table 4.4: The settings of the reward used for hyperparameter tuning (a description of these parameters is given in Table 3.1)

Reward setting	Value
$\lambda_{T_{\text{SP}}^{\text{AL1}}}$	0.1
σ_{SP}	3.0
$\lambda_{T_{\text{constr}}^{\text{AL2}}}$	0.8
p^{T}	ReLU
$\lambda_{a_{T_{\text{SP}}}}$	0.1
$\lambda_{a_{m_{\text{chill}}}}$	0.0
$\lambda_{a_{T_{\text{air}}}}$	0.0
p^{a}	Trapezoidal
$bw_{\text{a,trapezoidal}}$	1.0

4.5 Hyperparameter tuning results

The hyperparameters have been tuned according to the method described in section 3.4. This section first describes the implementation of the sampling in these experiments in Python. Consequently, results of an initial run to test and adjust the hyperparameter ranges are given in subsection 4.5.1. The results of the actual tuning with a lot more samples are given in subsection 4.5.2. All experiments try to minimize P^{HVAC} and constraint violations simultaneously by minimizing the reward function. The reward settings used for this hyperparameter tuning were set in initial experiments and are shown in Table 4.4.

Python implementation For the algorithm and reward hyperparameter tuning experiments, the Optuna [69] library is used. Optuna is the leading library in hyperparameter optimization and is known for its efficiency and flexibility. It supports various optimization algorithms, including the TPE sampler, which is used in this study. Optuna's framework allows easy integration with different ML libraries and provides an interface for defining and executing optimization trials. Its ability to handle both single-objective and multi-objective optimization makes it an ideal choice for both the algorithm and reward hyperparameter tuning in these experiments.

4.5.1 Initial hyperparameter search

Initially, a short hyperparameter search using the default ranges for PPO parameters (given in Table 4.5) was performed to investigate if these ranges should be modified. The range for the learning rate, α, has been adjusted to $\alpha \in \left[10^{-5}, 3 \cdot 10^{-3}\right]$, as it was found during earlier experiments that the learning of the agent could become unstable for learning rates of $\alpha = 10^{-2}$.

Figure 4.13 shows a parallel coordinate plot of these initial experiments. The hyperparameter combinations of the best-performing agents have been highlighted in this plot. This plot indicates the optimal ranges and the hyperparameter ranges are adjusted accordingly. For example, all top-performing agents have λ^{GAE} in the upper half of its range, so it is now set to $[0.95, 1]$. For parameters with preferences at the range borders, the ranges have been extended in that direction. Table 4.6 gives these updated ranges on which the extensive hyperparameter search is performed.

Table 4.5: The initial ranges of the hyperparameters used in this study.

Hyperparameter	Lower limit	Upper limit
γ	0.9	0.999
λ^{GAE}	0.9	1
ϵ	0.1	0.4
c^{H}	0	0.01
c^{VF}	0.5	1
α	10^{-5}	$3 \cdot 10^{-3}$
$n^{\text{layers, pi}}$	1	2
$n^{\text{neurons, pi}}$	16	64
$n^{\text{layers, vf}}$	1	2
$n^{\text{neurons, vf}}$	16	64
n^{epochs}	5	30
n^{steps}	128	2048
$n^{\text{minibatch}}$	32	n^{steps}

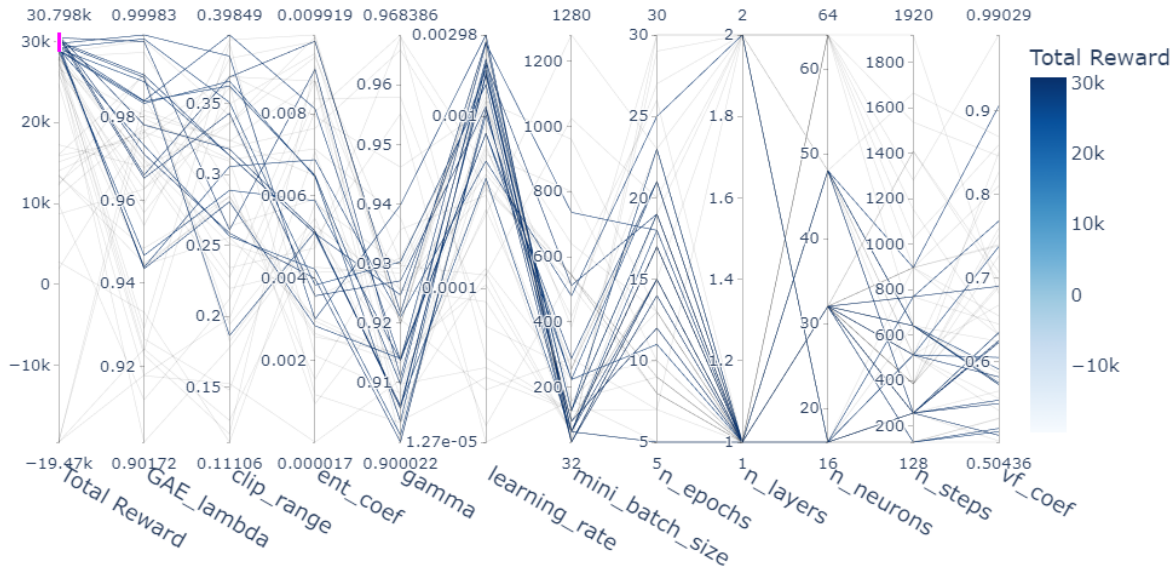
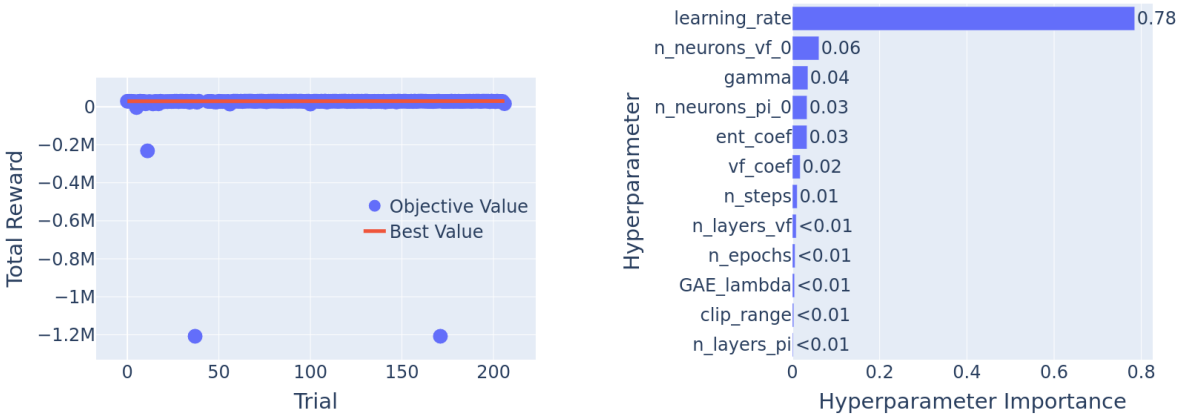


Figure 4.13: A parallel coordinate plot of the initial hyperparameter search. The axis on the left contains the total reward of an episode. The lines are then connected to the hyperparameters for that run. The best episodes are highlighted here, as can be seen in the *blue* selection on the *left* axis.

In this initial search, both the actor and critic network structure were assumed to be the same, with the same number of neurons for each layer. In the extensive search, these parameters are all tuned separately, as the networks serve different goals and might both have different optimal architectures.

Table 4.6: The updated ranges of the hyperparameters used in this study.

Hyperparameter	Lower limit	Upper limit
γ	0.85	0.95
λ^{GAE}	0.95	1
ϵ	0.2	0.5
c^{H}	0.002	0.01
c^{VF}	0.4	0.9
α	$5 \cdot 10^{-4}$	10^{-2}
$n^{\text{layers, pi}}$	1	3
$n^{\text{neurons, pi}}$	4	64
$n^{\text{layers, vf}}$	1	3
$n^{\text{neurons, vf}}$	4	64
n^{epochs}	10	25
n^{steps}	128	1024
$n^{\text{minibatch}}$	16	n^{steps}



(a) The cumulative reward against the trials in the study.

(b) The influence of each hyperparameter on the total reward.

Figure 4.14: The history and hyperparameter importance of the tuning experiments.

4.5.2 Extensive hyperparameter search

Using the updated ranges, a more extensive hyperparameter search of 200 hyperparameter samples has been performed. Figure 4.14a shows the total reward of the evaluation run of each of these samples. The figure shows that a number of hyperparameter settings have led to a very bad reward. This section first investigates what the best hyperparameter combination is. Afterward, the cause of the very bad performance is investigated.

4.5.2.1 Best parameters

The hyperparameters of the best-performing studies are shown in Table 4.7. A closer analysis of these results shows that most of the best-performing hyperparameters are similar. For example, $\lambda^{\text{GAE}} \approx 0.98$ for all combinations, and γ is at the lower end of its range. Although most other parameters are spread out more than this, they are still all in a close range.

Table 4.7: Top 5 trials of the hyperparameter search.

Reward	λ^{GAE}	ϵ	c^{H}	c^{VF}	γ	α	n^{batch}	n^{epoch}	$n^{\text{lay}}_{\text{pi}}$	$n^{\text{lay}}_{\text{vf}}$	$n^{\text{neur}}_{\text{pi,0}}$	$n^{\text{neur}}_{\text{pi,1}}$	$n^{\text{neur}}_{\text{pi,2}}$	$n^{\text{neur}}_{\text{vf,0}}$	n^{steps}
30303.87	0.99	0.45	$6.5 \cdot 10^{-2}$	0.75	0.85	$7.9 \cdot 10^{-4}$	48	25	3	1	17	6	15	60	512
30218.88	0.97	0.43	$5.8 \cdot 10^{-2}$	0.77	0.86	$8.0 \cdot 10^{-4}$	80	23	3	1	7	27	19	29	384
30176.69	0.99	0.34	$6.9 \cdot 10^{-2}$	0.75	0.86	$1.2 \cdot 10^{-3}$	80	24	3	1	9	13	41	54	512
30134.26	0.98	0.44	$7.8 \cdot 10^{-2}$	0.67	0.86	$1.0 \cdot 10^{-3}$	64	22	3	1	5	9	18	33	384
30134.11	0.99	0.44	$6.8 \cdot 10^{-2}$	0.75	0.86	$7.6 \cdot 10^{-4}$	64	24	3	1	10	12	22	59	512

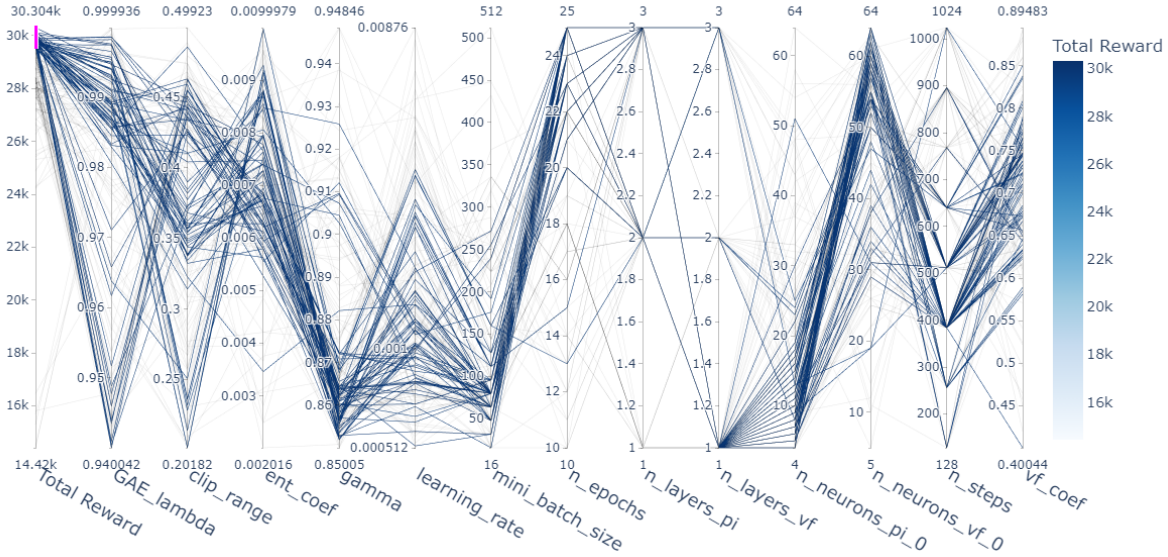
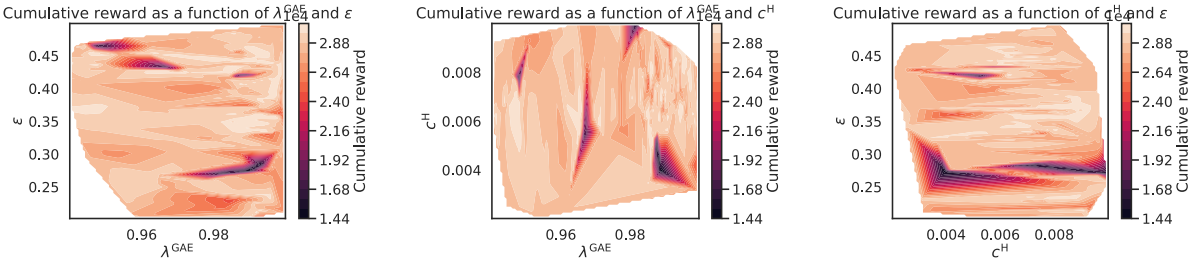


Figure 4.15: Parallel coordinate plot of the extensive hyperparameter study, with the 4 worst performing agents filtered out.

When looking at the network architectures of these best trials, however, it can be seen that the number of neurons per layer of the policy network changes drastically. This indicates that not yet enough trials have been performed to find the optimal network architecture. Therefore, it is decided to perform an additional hyperparameter search where only the network-related parameters are adjusted, this study is described in subsubsection 4.5.2.2.

The remainder of this section focuses on further analyzing these combinations of hyperparameters. It aims to judge the combinations that are found and to identify any risks, such as a high sensitivity to hyperparameter changes for certain combinations of hyperparameters.

To give a better overview of which hyperparameter combinations lead to good results, a parallel coordinate plot is shown Figure 4.15, with the 4 worst performing agents filtered out. Among the patterns this plot shows are the patterns which were found in Table 4.7, with high values for λ^{GAE}, moderate values for the clip range, low γ, etc. However, also other patterns lead to good results. On the left side, several other groups of well-performing agents can be distinguished. The first group consists of agents with high λ^{GAE}, low ϵ, and high entropy coefficients c^{H}. Another group includes agents with low λ^{GAE}, high ϵ, and moderate c^{H}. However, there are no well-performing agents with both low λ^{GAE} and ϵ.



(a) λ^{GAE} against ϵ. **(b)** λ^{GAE} against c^{H}. **(c)** c^{H} against ϵ.

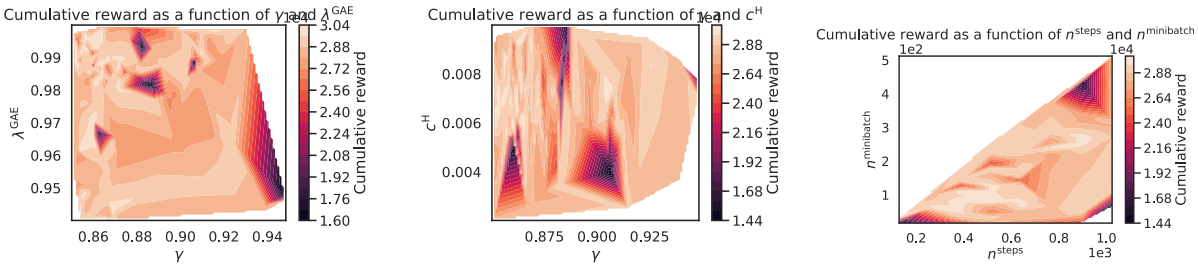
Figure 4.16: Contour plots, relating λ^{GAE}, ϵ and c^{H} to the cumulative reward.

Further analysis of hyperparameter interactions These combinations can be easily observed, however, there might also be other combinations that can not conveniently be spotted in this plot. Therefore, further investigation is required. To do so, contour plots relating 2 variables and the cumulative reward have been created, which are good at showing non-linear relationships between the variables. As there are many hyperparameters, and they can not all be analyzed, the remainder of this section highlights the most important relations.

In Figure 4.16, contour plots of the variables for which relations were found in the parallel coordinate plot are shown and these plots confirm the relations. For example, in Figure 4.16a, it is seen that there is indeed a small spot of good performance for the combinations of λ^{GAE} and ϵ. However, it can also be seen that there are no data points for low values of both parameters. Therefore, no conclusion can be drawn on whether the lack of well-performing agents with this combination is caused by the sampling of the TPEsampler or because this combination performs badly. As the expected combinations indeed lead to good results, it is assumed this goes for other combinations as well.
Next to the fact that these combinations indeed lead to good results, also several bad performing regions are spotted, seen as the blue spaces, such as a high λ^{GAE} and low c^{H}. Interestingly, for a clipping parameter between 0.25 and 0.30, bad results are found for all values of c^{H}, indicating that these variables are not correlated.

Another conclusion that can be drawn from these plots is that there is quite some variation in the cumulative reward for hyperparameters that are close together, while the same random seed has been used for all experiments. This could be caused by the fact that there are of course more hyperparameters that affect the outcome of a simulation than just the 2 that are displayed. Another reason could be that the simulations are very sensitive to hyperparameter changes.

Relation γ and λ^{GAE} Figure 4.17a depicts the relation between γ and λ^{GAE}. Interestingly, low values of γ lead to the best results. Next to that, the best results are found for high values of λ^{GAE}. This is interesting since, typically, the value of the GAE estimator is lower than the discount factor. It is expected that this behaviour is caused by the fact that EnergyPlus models the reaction of the components to changes in their setpoints instantly. This allows the agent to respond very quickly. Therefore, γ can be quite low, because the PPO algorithm does not have to take the expectation of reward in the distant future into account to optimize the cumulative reward. The future cumulative reward will be maximized if the policy simply learns to get to the state with the highest reward in the current timestep.



(a) γ against λ^{GAE}. **(b)** γ against c^{H} **(c)** n^{steps} against $n^{\text{minibatch}}$.

Figure 4.17: Contour plots, relating several combinations of variables to the cumulative reward.

Consequently, there is less variance in the future discounted reward, which explains the high values of λ^{GAE}. When recalling the estimation of the advantage using $\hat{A}^{\text{GAE}}$ in Equation (3.23), one can see that both γ and λ^{GAE} discount future values. λ^{GAE} is mainly used as an addition to γ to balance the variance of future rewards and bias on the current reward. Since γ already greatly reduces the variance of future rewards, λ^{GAE} is allowed to be quite high, which causes less bias.

Relation γ and c^{H} In Figure 4.17b, it can be seen that the agent performs badly for a combination of the low value of γ and low c^{H}. This is likely caused by the fact that a low γ leads to the current reward being the main influence on the surrogate loss function of the PPO algorithm (Equation (3.19)). c^{H}, on the other hand, promotes randomness in the policy by adding a bonus for the randomness of the policy on the surrogate loss function, leading to more exploration. If it is low, the algorithm is rewarded most for getting the HVAC system in the best-known state, which is often not the best state to be in.

Relation n^{steps} and $n^{\text{minibatch}}$ Finally, Figure 4.17c shows the relation between the collection steps and the minibatch size. It can be seen that the best-performing minibatch size is roughly proportional to the amount of steps, indicating that about 6 minibatches per collection step are optimal. n^{steps} mainly determines the convergence speed of the PPO algorithm, as the policy and value networks update more frequently when n^{steps} is lower (as seen in 1). However, a larger value for n^{steps} can lead to more stable learning, as more data is collected. On the other hand, the advantage approximation is based on the value following the (not yet) optimal policy. Too large n^{steps} might thus lead to slower convergence. From these results follows that around 500 collection steps with 6 minibatches is optimal for final performance. The convergence speed and learning stability are not taken into account in this hyperparameter study.

4.5.2.2 Network architecture search

As mentioned before, another hyperparameter study with 200 trials was performed where only the network architecture could be adjusted (so the number of layers and neurons per layer). For the other hyperparameters, the optimal values from Table 4.7 are used. In Table 4.8, the top 5 performing studies are shown.

Interestingly, these results are entirely different from the architectures found in the previous study. Firstly, the optimal layers are found to be just 1 layer for each network. This is more in

Table 4.8: Top 5 trials for the network architecture study

Reward	$n^{\text{layers, pi}}$	$n^{\text{layers, vf}}$	$n_0^{\text{neurons, pi}}$	$n_0^{\text{neurons, vf}}$
28683	1	1	28	18
28312	1	1	26	8
28272	1	1	28	1
28183	1	1	27	4
28147	1	1	43	10

line with the expectations, as the EnergyPlus component models are relatively simple, and the loss function of the algorithm mainly depends on the current state due to the low values of γ. Therefore, not a very abstract relationship is expected, which should be sufficiently describable by a single network layer. Secondly, it was found that the policy network layer should be relatively wide, but just a bit smaller than the amount of states (which is 33). The value function network can be smaller than this. This can be explained by the fact that the policy network has all 3 action variables as an output, while the value function just has one output. The network architecture of the best-performing agent is used in the remainder of this thesis.

4.5.2.3 Cause of Bad Results

This section investigates the cause of the poor performance of certain agents. Analyzing the worst-performing agents shows several important relationships. Table 4.9 displays the hyperparameters of the 4 worst-performing training runs. All these agents have a high learning rate. Initial experiments indicated that a high learning rate can lead to poor results, as the networks update too much based on current, biased changes in the loss function. However, high values of α did not always result in poor performance, which suggests that the issue is likely caused by a combination of a high learning rate and other factors.

Further analysis showed that all poorly performing agents have large policy networks, indicating potential overfitting on initial data. Additionally, the number of neurons per layer is suboptimal. For instance, the last two rows of Table 4.9 show that the number of neurons ranges from small to large, despite the neural networks having many input parameters (33 states) and few output parameters (3 actions).

To conclude, the bad performance is likely caused by a combination of the agent networks updating in too large steps due to the high learning rates and large policy networks that have a bad architecture. This can make the networks overfit on the currently updated timesteps, and the large networks allow for finding more complex relations than exist in reality. Avoiding this is key in the implementation of agents.

4.5.3 Discussion & conclusion on hyperparameter tuning

From the hyperparameter search, a number of interesting conclusions can be drawn. Firstly, γ can be a lot lower than the default range, and the optimal value might even be lower than the value found. This is likely caused by a combination of the stochasticity of the system, getting a reward every timestep, and the quick response time of EnergyPlus, which make it unfeasible to put a large weight on future rewards. Consequently, λ^{GAE} can be relatively high, as the

Table 4.9: Network architectures and learning rate of the agents with very bad results

Total Reward	α	$n^{\text{layers,pi}}$	$n_0^{\text{neur,pi}}$	$n_1^{\text{neur,pi}}$	$n_2^{\text{neur,pi}}$	$n^{\text{layers,vf}}$	$n_0^{\text{neur,vf}}$	$n_1^{\text{neur,vf}}$
$-3.9 \cdot 10^3$	$7.4 \cdot 10^{-3}$	3	61	22	30	1	30	-
$-2.3 \cdot 10^5$	$9.5 \cdot 10^{-3}$	3	36	61	20	2	7	41
$-1.2 \cdot 10^6$	$6.9 \cdot 10^{-3}$	2	9	59	-	2	58	19
$-1.2 \cdot 10^6$	$9.8 \cdot 10^{-3}$	3	8	14	43	1	31	-

Table 4.10: Best hyperparameters

λ^{GAE}	ϵ	c^{H}	c^{VF}	γ	α	n^{steps}	n^{batch}	n^{epoch}	$n^{\text{layers,pi}}$	$n_0^{\text{neurons,pi}}$	$n^{\text{layers,vf}}$	$n_0^{\text{neurons,vf}}$
0.99	0.45	$6.5 \cdot 10^{-2}$	0.75	0.85	$7.9 \cdot 10^{-4}$	512	48	25	1	28	1	18

variance due to higher reward is already relatively low. Another important hyperparameter is the learning rate α. It was found that this parameter can be relatively high (around $8.0 \cdot 10^4$), allowing for quick updates of the networks, but it should be avoided to have this parameter too high, as a combination of high learning rates and large neural networks leads to severe overfitting. Finally, the additional network architecture search showed that the optimal networks should have a single layer, where the policy network has slightly more neurons than the value function network.

Combining this knowledge, the optimal set of hyperparameters is found by taking the best hyperparameter combination of the network search, which is displayed in Table 4.10.

An observing reader could also have noticed that the maximum reward in the network architecture search is a lot lower than in the other hyperparameter studies ($2.8 \cdot 10^4$ versus $3.0 \cdot 10^4$). At first, it was expected that this study just did not find results as good as the other study. However, the first hyperparameter search and the network architecture search were performed on different virtual machines. The optimal hyperparameters of the first study led to even slightly worse results than $2.8 \cdot 10^4$ on the second virtual machine. Therefore, it is expected that this is caused by the machine-specific initialization of the random seed. Some small hyperparameter studies were performed with a different random seed which still indicated similar combinations of hyperparameters are optimal, so this seed sensitivity is not expected to influence the results of the hyperparameter study. However, further investigation of this sensitivity is required as it does affect the learning robustness of the agent. This is done in section 4.7.

4.6 Reward tuning results

The tuned algorithm can now be used for the tuning of the reward function. This section presents the results of the reward tuning experiments that have been conducted following the method described in subsection 3.4.2. First, general results of the reward tuning experiments are provided. Following this, a more in-depth investigation into the influence of the hyperparameters on the performance of the RL-based controller is performed. Finally, 3 reward configurations are selected. These configurations will be used to train 3 controllers which are then tested on the test data and compared to the baseline controller in the next section.

In these experiments, the hyperparameters of Table 4.10 have been used, while the reward parameters were treated as hyperparameters. The range of these parameters was determined using initial experiments and are based on agents which gave reasonable results, these ranges are shown in Table 4.11. Next to that, it was found that high frequent action oscillations only occurred when there was no action penalty at all. To reduce the amount of hyperparameters, therefore only one action fluctuation penalty is applied, to $T_{\text{chill,SP}}$

$$r_{t_i} = \underbrace{r_{\text{P}}\left(P_{t_i}^{\text{HVAC}}\right)}_{\text{HVAC Power reward}} + \underbrace{\lambda_{T_{\text{SP}}^{\text{AL1}}} \cdot r_{\text{T}}\left(T_{t_i}^{\text{AL1}}\right)}_{\text{CRAH setpoint reward}} + \underbrace{\lambda_{T_{\text{constr}}^{\text{AL2}}} \cdot p_{\text{T}}\left(T_{t_i}^{\text{AL2}}\right)}_{\substack{\text{Server outlet} \\ \text{temperature penalty}}} + \underbrace{\boldsymbol{\lambda}_{\text{a}} \cdot \boldsymbol{p}_{\text{a}}\left(\Delta \boldsymbol{a}_{\boldsymbol{t}_i}\right)}_{\substack{\text{Action fluctuation} \\ \text{penalty}}} \tag{4.1}$$

Table 4.11: The ranges of the reward parameters

Parameter	Lower limit	Upper limit	Options
$\lambda_{T_{\text{SP}}^{\text{AL1}}}$	0	1.5	-
σ_{SP}	0	5	-
$\lambda_{T_{\text{constr}}^{\text{AL2}}}$	0	3	-
p_{T}	-	-	ReLU, ReLU2, Softplus
β_{softplus}	0	10	-
$\lambda_{\text{a},T_{\text{chill,SP}}}$	0	0.5	-
p_{a}	-	-	Linear, Quadratic, Trapezoidal
$bw_{\text{a,trapezoidal}}$	0	7	-

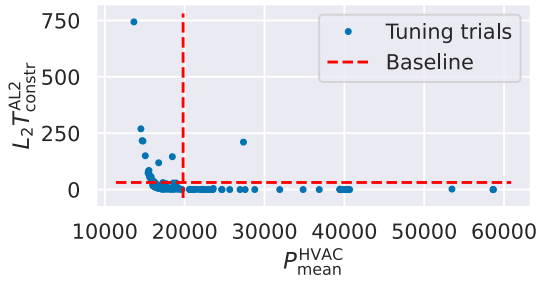
The agents are all trained on the training data and then evaluated on the validation data of 2022. In this analysis, the performance of the agents is compared to that of the baseline (also over 2022, note that the total values are thus slightly different than in section 4.3) on the 4 factors: $P_{\text{mean}}^{\text{HVAC}}$, $L_2\left(T_{\text{constr}}^{\text{AL2}}\right)$, $L_2\left(T_{\text{SP}}^{\text{AL1}}\right)$ and $L_2\left(\Delta a\right)$.

4.6.1 General results

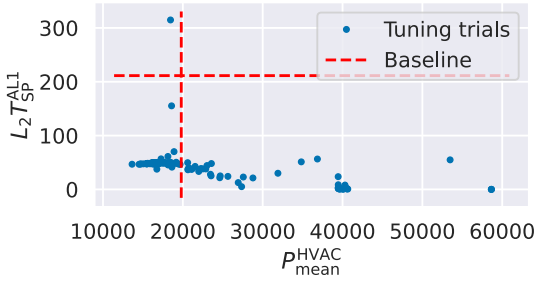
Figure 4.18 shows the results of the reward tuning experiments. On the horizontal axis, the average HVAC power in the validation run for each trial is displayed, while the y-axes display the 2-norms of the 3 performance parameters. The dotted lines portray the performance of the baseline. From these results, several conclusions can already be drawn, while other phenomena require further investigation.

Firstly, in Figure 4.18b, it can be seen that almost all agents work better than the baseline regarding the deviation from the cold air setpoint in the room, $T_{\text{SP}}^{\text{AL1}}$. When taking a closer look at the only trial which resulted in worse performance than the baseline, it was found that $\lambda_{T_{\text{SP}}^{\text{AL1}}}$ was very low in this case ($1.1 \cdot 10^{-2}$). This means the algorithm only received small rewards for meeting the setpoint, thus the surrogate loss function was barely affected by deviations from the setpoint. Consequently, the policy was not updated to meet the setpoint. This indicates that, above a certain threshold, the RL agent will always be able to outperform the baseline in this regard.

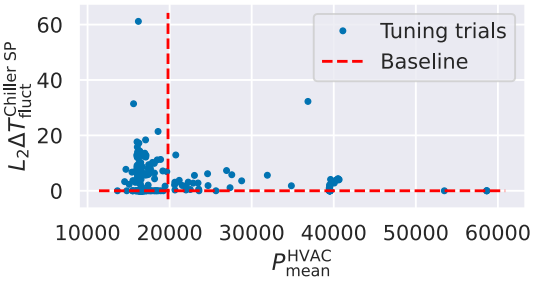
Secondly, from Figure 4.18c follows that the baseline outperforms all RL agents regarding the action fluctuation. This is trivial, as the baseline has a constant setpoint, and one of the



(a) Comparison of the value of $P^{\text{HVAC}}_{\text{mean}}$ to the 2-norm of the constraint violations.



(b) Comparison of the value of $P^{\text{HVAC}}_{\text{mean}}$ to the 2-norm of the setpoint deviation.



(c) Comparison of the value of $P^{\text{HVAC}}_{\text{mean}}$ to the 2-norm of the action fluctuation.

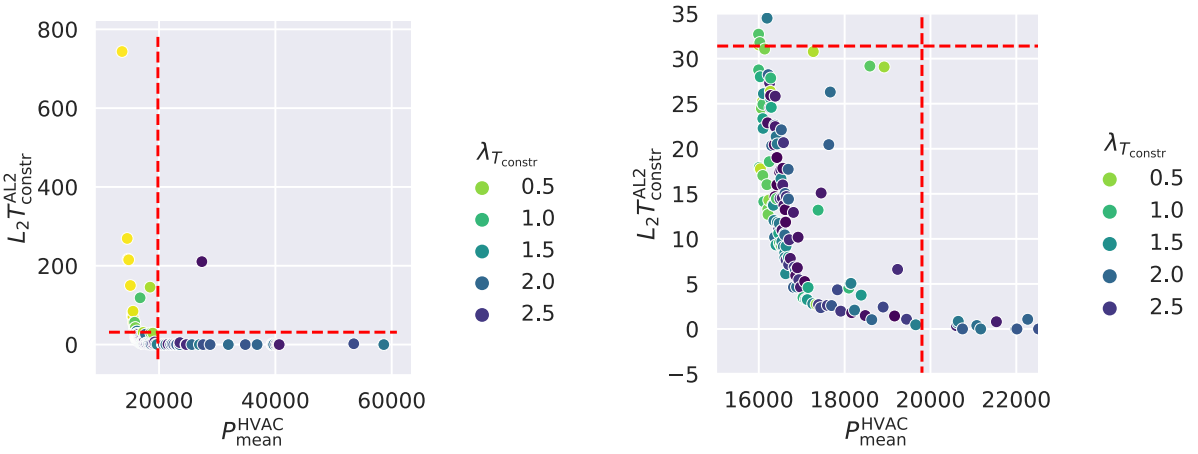
Figure 4.18: The overall reward tuning results.

strengths of the RL agents is that it can actually adapt its setpoint. The goal of this metric is to prevent severe, high frequent oscillations in the system. However, this only happens for L_2 values above 10^3. Since all values are well below this, the agents all perform well enough in this case.

Combining this knowledge and the knowledge that severe action oscillations *did* occur when no penalty on the action fluctuation was applied, it can be concluded that the RL agent performs well regarding $T^{\text{AL1}}_{\text{SP}}$ and $T^{\text{Chiller SP}}_{\text{fluct}}$ as long as there is at least some penalty on this behaviour. Therefore, these variables do not have to be taken into account for the selection of optimal reward settings. However, when looking at Figure 4.18a, it can be seen that there is a clear trade-off between the $L_2 T^{\text{AL2}}_{\text{constr}}$ and the reduction of P^{HVAC}. The following sections focus on deducing which settings cause the agent to either save more energy or adhere to the temperature constraint better. Using that understanding, a selection of the optimal reward settings is made.

4.6.2 Trade-off between HVAC power and constraint violations

As the PPO algorithm simply updates the networks to maximize the reward it gets, without any knowledge of the reward itself, the penalty has a large influence on its performance. Therefore, to find out which parameters influence this trade-off, the parameters related to the constraint temperature penalty are a logical first choice. These are the weight and type of penalty function of the constraint penalty. To start, the weight of the penalty is expected to have a large effect on this system, as the less weight is placed on the constraint violations, the lower the advantage is of taking actions that prevent constraint violations at the cost of P^{HVAC}. Thus $\hat{A}^{\text{GAE}}$ is not



(a) All parameters

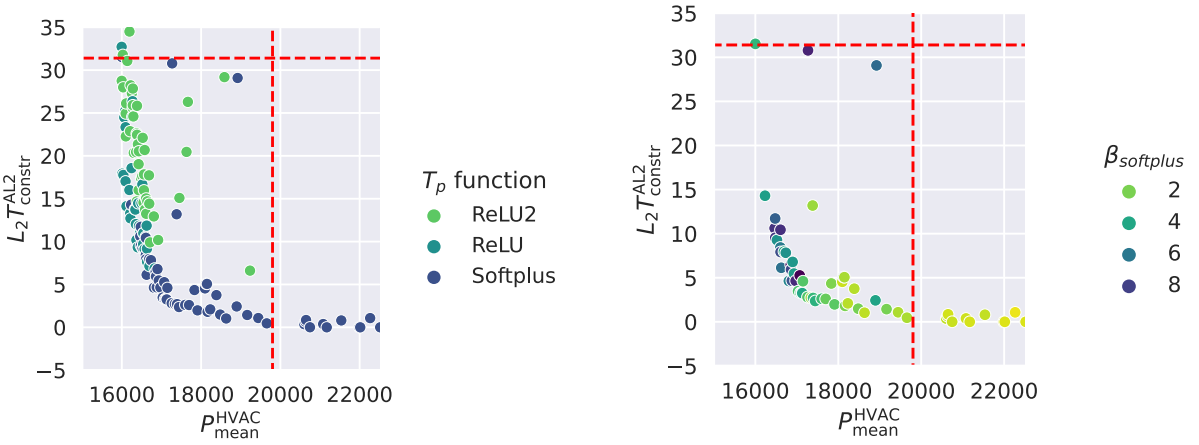
(b) Zoomed in

Figure 4.19: The effect of $\lambda_{T^{AL2}_{constr}}$ on the trade-off between P^{HVAC} and T^{AL2}_{constr} violations.

affected as much by constraint violations as for higher weights. To investigate if this weight indeed has this effect, Figure 4.19 shows the results of the trials again with P^{HVAC} on the x-axis and $L_2\left(T^{AL2}_{constr}\right)$ on the y-axis, and the results colored according to the value of $\lambda_{T^{AL2}_{constr}}$. In Figure 4.19a, it can be seen that, generally, the higher the weight on penalizing constraint violations, the lower the severity of the violations is. This comes at the expense of higher power usage. When zooming in at the area where the agents outperform the baseline both on energy usage and constraint violations (Figure 4.19b), this relation is less clear. As this is the region which is the most important, further investigation is required.

The other factor affecting the penalty on T^{AL2}_{constr}, is the type of function that is used to penalize the reward. In Figure 4.20a, again a similar plot of the results is made, this time coloring the different types of penalty functions. When recalling subsection 3.2.2, it was expected that the ReLU function would be easiest to tune, while ReLU2 has better energy efficiency and Softplus has better constraint handling. When looking at the figure, it can be clearly seen that indeed, Softplus is the best at constraint handling, at the expense of some energy efficiency of the system. The ReLU2 is not significantly better at reducing P^{HVAC}, while it is worse at handling constraints than the ReLU penalty.

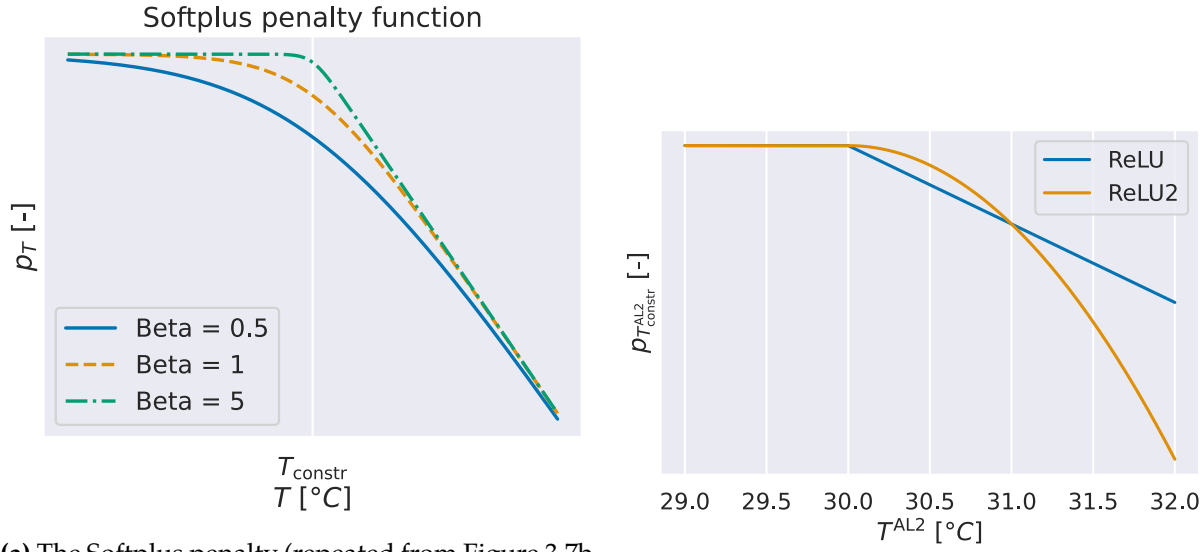
In the case of the Softplus penalty, a wide range of mean HVAC power is found. To further investigate what causes this, Figure 4.20b displays the effect of modifying the β-parameter, which (when recalling subsection 3.2.2) affects the smoothness of the softplus function. For higher values of β, the softplus function becomes sharper and resembles the ReLU function more. From the figure follows that, indeed, high values of β lead to performance similar to that of the ReLU function, while lower values lead to very good constraint performance, albeit at the expense of HVAC power. The reason why the Softplus function is so good at handling constraints can be best explained by looking at Figure 4.21a. Softplus already provides a penalty for temperatures below the constraint, so the $\hat{A}^{GAE}$ will give less of an advantage for keeping T^{AL2} close to its constraint. This gives the controller a kind of buffer to the actual constraint temperature. This comes at the expense of requiring more power, as the air in the room has to be cooled down more, explaining the higher HVAC power in this case. On a side note, a similar effect might be accomplished by simply shifting a ReLU function to penalize already before the



(a) Effect of the type of constraint penalty.

(b) The effect of the smoothness of Softplus.

Figure 4.20: The effect of the constraint penalty function on the trade-off between P^{HVAC} and T^{AL2}_{constr} violations.



(a) The Softplus penalty (repeated from Figure 3.7b for convenience).

(b) The ReLU and ReLU2 penalties compared.

Figure 4.21: The penalty functions.

constraint temperature. However, this is left open to further research.

The fact that ReLU2 leads to a higher L_2-norm for the constraint violations than ReLU can easily be explained by Figure 4.21b. As can be seen, for smaller constraint violations (between 30 and 31 °C), the ReLU2 function gives a smaller penalty on the constraint violations. Thus, the PPO algorithm gets a smaller penalty on small violations of the constraints. Allowing T^{AL2} to slightly violate the constraint was expected to reduce P^{HVAC}. However, from Figure 4.20a follows that this energy saving is negligible compared to the added constraint violation.

4.6.3 Discussion & conclusion on reward tuning

The reward tuning search showed that the RL agents can outperform the baseline controller for the cold air temperature setpoint deviation metric, even with a small reward for this setpoint. Although the baseline had lower action fluctuations due to its stationary chilled water setpoint, the RL agents avoided severe high-frequency oscillations with a small penalty on action variation. There is a trade-off between energy use (P^{HVAC}) and constraint handling ($T^{\text{AL2}}_{\text{constr}}$), influenced by the weight and type of constraint violation penalty. Lower penalty weights lead to higher violations but reduced energy use, as higher room temperatures decrease cooling system mass flows. The ReLU function best balances HVAC power saving and constraint violations, while the Softplus function, with its buffer, performs best at constraint handling.

Reward parameter selection Obviously, there are many trials on the optimal Pareto front. To best show how the versatility possible with RL agents compared to the baseline, the following 3 agents are selected (their performance and settings are summarized in Table 4.12):

- **Constraint-Optimized Agent (COA):** an agent which performs *similarly* for P^{HVAC} and *better* for $T^{\text{AL2}}_{\text{constr}}$.
- **Balanced Agent (BA):** an agent that outperforms the baseline for both parameters.
- **Energy-Optimized Agent (EOA):** an agent which performs *better* for P^{HVAC} and *similar* for $T^{\text{AL2}}_{\text{constr}}$

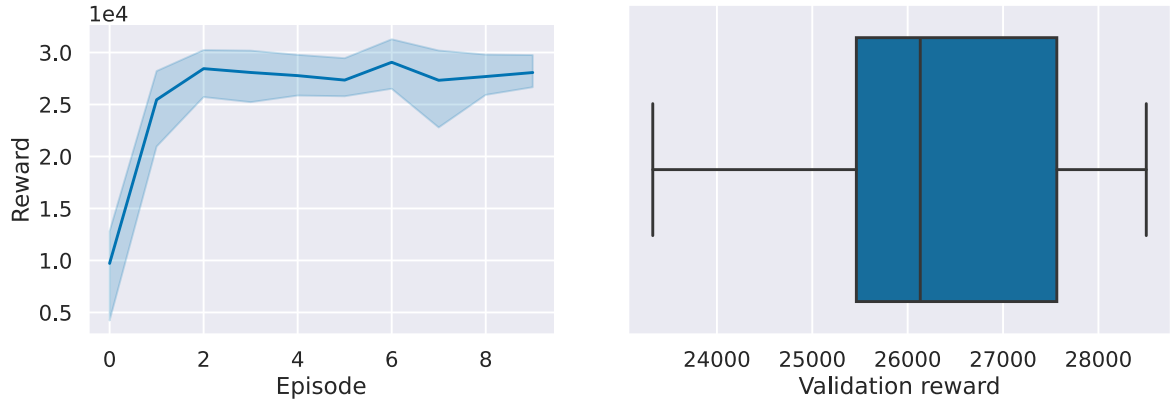
Table 4.12: The parameters of the 3 agents that are further investigated.

Parameter	COA	BA	EOA
$P^{\text{HVAC}}_{\text{mean}}$	19656	16808	15863
$L_2\left(T^{\text{AL2}}_{\text{constr}}\right)$	0.46	4.6	40.4
$\lambda_{T^{\text{AL1}}_{\text{SP}}}$	0.83	0.97	0.43
σ_{SP}	4.3	2.1	2.4
$\lambda_{T^{\text{AL2}}_{\text{constr}}}$	1.7	2.1	0.27
p_{T}	Softplus	Softplus	ReLU
β_{softplus}	1.25	5.4	-
$\lambda_{\text{a},T_{\text{chill,SP}}}$	0.24	0.19	0.43
p_{a}	Trapezoidal	Trapezoidal	Linear
$bw_{\text{a,trapezoidal}}$	3.8	1.6	-

4.7 Learning robustness results

In section 4.5, it was found that 2 agents with the same settings, but different seeds for the random number generator, would lead to a different final reward on the validation data. This seed is used to ensure all random number generators in the framework give repeatable results. These random number generators are used for:

- Initialization of the weights and biases of the policy and value networks.



(a) The reward during the training runs for 20 different seeds, uncertainty region contains 100% of collected data.

(b) Boxplot of the distribution of the rewards of the validation runs.

Figure 4.22: Results of the robustness tests

- Sampling of actions from the policy probability distribution (output of the policy network). This determines the actions the PPO algorithm takes during training.
- Sampling of hyperparameters in tuning experiments (although this is not related to the training robustness of an agent, it influences overall results).

As just changing the seed affects the outcome, further investigation into how robust the agents are to changes in this seed is required.

To test this robustness, agents with the reward used for hyperparameter tuning (Table 4.4) are trained with different seeds. This section presents the results of these tests and discusses the effect of the outcome on the thesis.

Training level results Figure 4.22a shows the spread of the yearly cumulative reward during training of 20 simulations with different initialization seeds. As can be seen, the general learning curve is similar for all simulations. There is, however, quite a large spread. The spread is around 20% of the total reward towards the end of the simulation and it is even larger in the first episode.

Figure 4.22b shows the distribution of the validation reward, which is approximated using kernels. As can be seen, this distribution is also spread widely, with the largest and smallest reward deviating 9% from the mean validation reward.

Discussion These tests indicate that, while the random seed does not destabilize the agent's learning process, the results vary significantly depending solely on the seed. Ideally, the PPO algorithm should converge to the same, optimal policy, regardless of its initialization.

Obviously, a true optimal policy will seldom be found in reality, due to the dynamic nature of RL. The PPO algorithm collects samples using the policy and consequently updates this policy and the value function. Next to that, the surrogate loss function of the policy network depends

on the value function, which is in turn approximated using the collected data using the suboptimal policy. Therefore, the initially collected samples affect the found policy. However, that the seed dependence is so large in this research also indicates that the agent does not explore enough during training and thus stays at its suboptimal solutions. This is a known problem in ML and in classical ML, regularization techniques are often applied to reduce this sensitivity. However, these techniques, such as dropout layers, are less suitable for RL algorithms such as the PPO, as such techniques often increase variance, while that is actually already the problem in RL due to its dynamic nature.

As the conventional regularization techniques are not suitable for RL algorithms, the easiest way to reduce seed sensitivity is by changing the hyperparameters or reward function. Based on the used hyperparameters, and the underlying principles of the PPO algorithm, the following possible causes of the sensitivity have been identified:

- **Data correlation** The data collected by the agent may be too correlated since only one environment is used instead of multiple parallel environments. In a multi-environment setup, different actions are sampled and thus the agent updates its policy based on diverse experiences, potentially balancing between good and bad initial explorations.
- **Too large penalty** A high penalty for constraint violations might influence the agent's behavior during initial episodes. Severe penalties in early episodes could discourage the agent from exploring strategies that approach the constraints closely, where optimal rewards might lie.
- $c^{\mathbf{H}}$ **too low** As the entropy coefficient encourages randomness in the surrogate loss function of the PPO, it makes the agent explore more. It could be the case that the value found in the hyperparameter tuning leads to better results with that specific seed, but is too low for sufficient exploration.
- $n^{\mathbf{steps}}$ **size** A final explanation would be that the trajectory collection, n^{steps}, is too small, which leads to very fast convergence but can make the agent avoid good exploration.

Identifying the cause of this problem requires further investigation, which is out of the scope of this thesis. However, this problem with the sensitivity does expose a weakness in the hyperparameter tuning experiments. The hyperparameter tuning searches for a single experiment with the highest reward. It was assumed that this would be the best set of hyperparameters. However, it also promotes searching for hyperparameters that are sensitive to small parameter changes, such as the seed. If there is a large spread in possible outcomes, there is a large chance that there is at least one very positive outcome, while the average reward might not be the best if experiments with more random seeds would be performed.

An ideal solution for this would be to perform multiple simulations using the same hyperparameters and different seeds during the optimization. However, this would increase the computational time of the experiments drastically, and therefore, it is chosen not to do this for this thesis.

Decreasing this sensitivity is left as an open problem for further research. However, it does have a large effect on the outcome of the experiments in this thesis. Therefore, the 3 optimized agents, COA, EOA, and BA are trained using several seeds so that the variability can be taken into account during further training.

4.8 Analysis of tuned controllers and comparison to baseline

In section 4.6, 3 agents were selected from the Pareto front. These agents are either as energy efficient as the baseline, but better at handling constraints (the COA); more energy efficient than the baseline, and as good as the baseline at handling constraints (the EOA); or slightly better at both (the BA). In this section, first, an analysis of how these agents compare to the baseline during training is given. After this, their performance on the validation data is analyzed. Then, the agents are tested on the test data. Their results and the learned control strategies of these agents are analyzed in depth. In all these analyses, the findings about the seed sensitivity in section 4.7 are taken into account.

4.8.1 Training phase

The 3 agents have been trained for 10 episodes on the training data files (2018-2021). Taking the seed sensitivity into account, the agents have been trained using 10 different random seeds. Their performance during training is compared to the performance of the baseline controller on the 4 performance metrics in Figure 4.23. Below is an analysis of the agents' performance for each metric.

HVAC power Regarding P^{HVAC}, the agents behave entirely as expected, as shown in Figure 4.23a. The COA uses a lot more power during the first year of training but learns to use on average just as much power as the baseline. This is what it was selected for. The BA uses more power than the baseline controller in the first year but learns to use less energy every year after that. Finally, the EOA uses less energy than the baseline controller starting from the first year.

From the spread, it can be seen that the seed again influences the convergence of all agents. However, especially for the BA and EOA, the RL-based controllers always outperform the baseline controller regardless of this variance in performance.

$\boldsymbol{T}^{\textbf{AL2}}_{\textbf{constr}}$ In Figure 4.23b, the performance of the agents regarding the constraint handling during training is shown. In the first year of training, all agents are not good at handling the constraints and are penalized heavily for it. This is in line with the expectations, as the agents are still in their exploratory phase and have not yet learned how the HVAC system responds to changes in the control variables.

After the first year, both the COA and BA are around as good at handling these constraints as the baseline controller, with some agents even outperforming the baseline a lot, depending on the seed. This performance is not as good as expected, and the validation run will tell if this is caused by exploratory behaviour during training, or if the selected agent just was an outlier regarding the constraint handling.

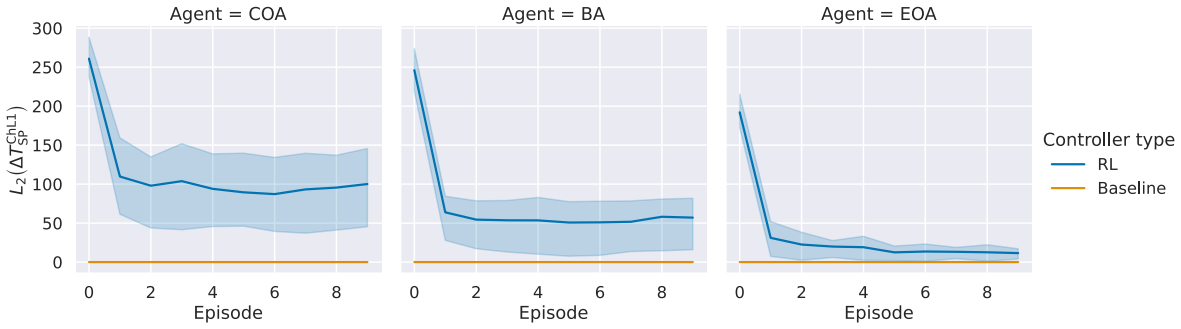
The EOA has a larger spread caused by the seed. Here, only the most advantageous seed leads to constraint handling comparable to that of the baseline. For this agent, the validation set will also tell if the behaviour is caused by the exploratory behaviour or not.

Although it is a bit out of the scope of this thesis, which focuses on the performance of trained agents, it should be noted that the constraint violations at the start of training are severe. Figure 4.24 show that very high temperatures are reached during the first training year, which can

(a) The HVAC power.

(b) The $T^{\mathrm{AL2}}_{\mathrm{constr}}$ violations.

(c) The $T^{\mathrm{AL1}}_{\mathrm{SP}}$ deviation.



(d) The Δa-fluctuation.

Figure 4.23: The performance of the agents during training, compared to the baseline controller in the training years.

be a limitation for the application of these types of agents directly in reality. When the controllers are implemented by pretraining on a model and then applying it to reality, this is less of a problem. However, there is still no guarantee of not violating constraints when adjusting from the simulation to reality. This is, however, a subject for further research.

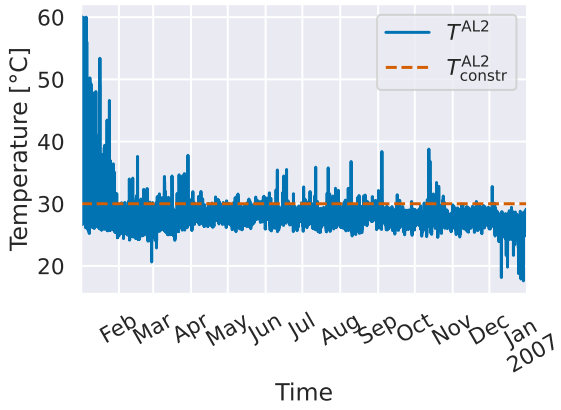


Figure 4.24: The constraint violations during the first year of training of the COA

T_{SP}^{AL1} After the agents deviate from the cold air setpoint more than the baseline controller does in the first year, as shown in Figure 4.23c, they all learn to meet the setpoint more during all following years. Interestingly, the EOA deviates more from the setpoint than the other agents. This can be explained by the fact that $\lambda_{T_{SP}^{AL1}}$ is lower in this agent. It is not known if this is one of the causes of the agent being better at saving energy, or just a side effect of a further relatively unimportant parameter.

Action fluctuation Finally, Figure 4.23d shows the fluctuation of the chilled water setpoint temperature. This gradually decreases and does not lead to any severe oscillations.

4.8.2 Validation & test performance

The agents have also been validated using the validation set, again using the 10 different seeds. In Figure 4.25, boxplots are shown for the performance of the agents on this year.

The tuned and trained agents are then finally tested on the data of the test year. This shows if the agents can control the HVAC system better than the baseline controller, or if there has been overfitted on the validation data during the many tuning trials. Again, the agents trained on 10 different seeds are all tested on this set. The results of the controllers on the test data are shown in Figure 4.26. As can be seen, the performance of the agents is almost the same as the performance on the validation data. This indicates that the tuning experiments have indeed resulted in well-tuned agents and did not overfit on the validation data. The remainder of this subsection first compares the performance of the controllers on the test data to the performance of the baseline based on these boxplots. Then, it determines whether or not the differences found are significant by a statistical analysis.

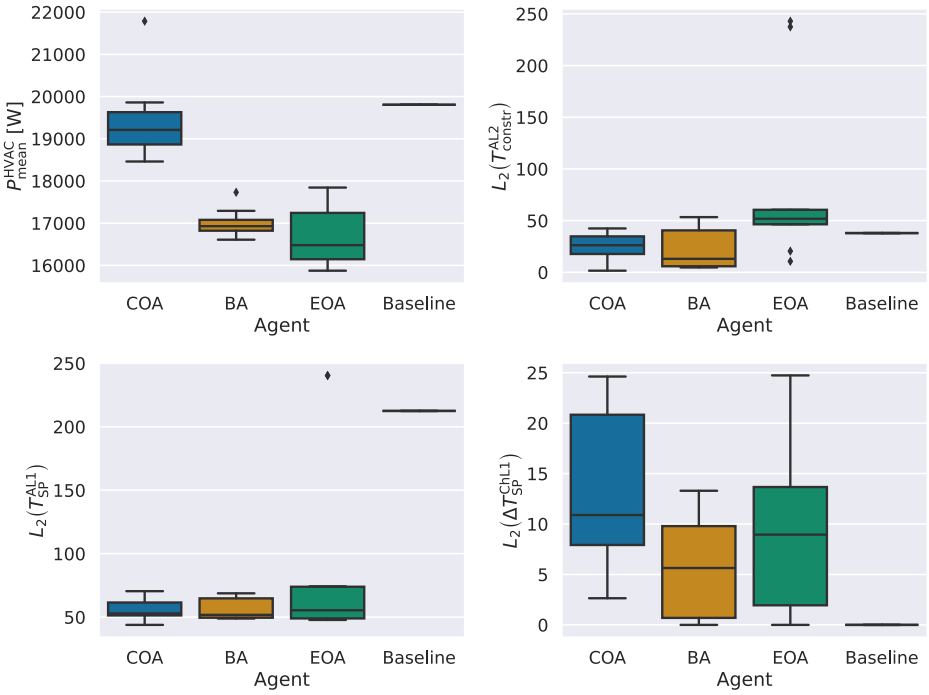


Figure 4.25: Boxplots of the performance metric for the different agents on the *validation* set, which have been trained on 10 different seeds.

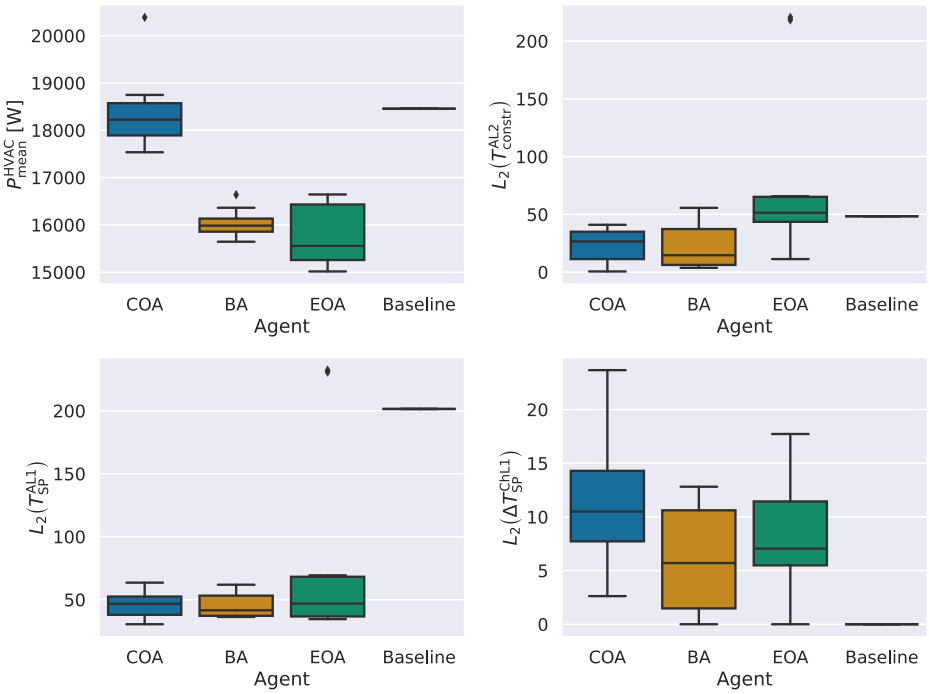


Figure 4.26: Boxplots of the performance metric for the different agents on the *test* set, which have been trained on 10 different seeds.

Visual comparison to baseline Regarding the mean HVAC power, all agents seem to behave as expected. The COA has power usage slightly below that of the baseline, while the BA and EOA are more energy efficient and comparable to each other. Interestingly, the BA seems to be less sensitive to the seed, although too few trials are used to conclude from this.

Regarding $T^{\text{AL2}}_{\text{constr}}$, both the COA and BA seem to show improvements compared to the baseline, and are similar to each other. Their performance is generally slightly worse than expected from the reward tuning, which is likely caused by a favourable initialization of the networks in these experiments. The EOA shows behaviour comparable to the baseline controller, and thus slightly worse than the other controllers, again as expected.

Finally, for the setpoint deviation and action fluctuation, all agents behave comparably to the training performance and do not seem significantly different from each other. It is interesting to see that the action fluctuation is much better for the validation set than during training. This is again likely caused by the exploratory behaviour during the training phase.

Statistical analysis To ensure that the observed differences are indeed significant, and not just coincidental, a statistical analysis has been performed. The detailed results of this analysis are presented in Appendix E, while the general procedure is discussed here.

First, a Shapiro-Wilk test was conducted to test whether the performance metrics were normally distributed or not. For this, a 95% confidence interval has been used. Since several metrics were not normally distributed, such as $P^{\text{HVAC}}_{\text{mean}}$ for the COA, a test suitable for non-normal data should be employed to compare the performance of the agents. Note that, as the baseline is seed-independent, it resulted in 10 times the same result. The results of this test are shown in Table E.1.

Following this, a Mann-Whitney U test has been conducted on each combination of agents for each metric. Again, a 95% confidence interval was used here. The results of this test are shown in Table E.2 and confirm all the observations of the previous paragraph.

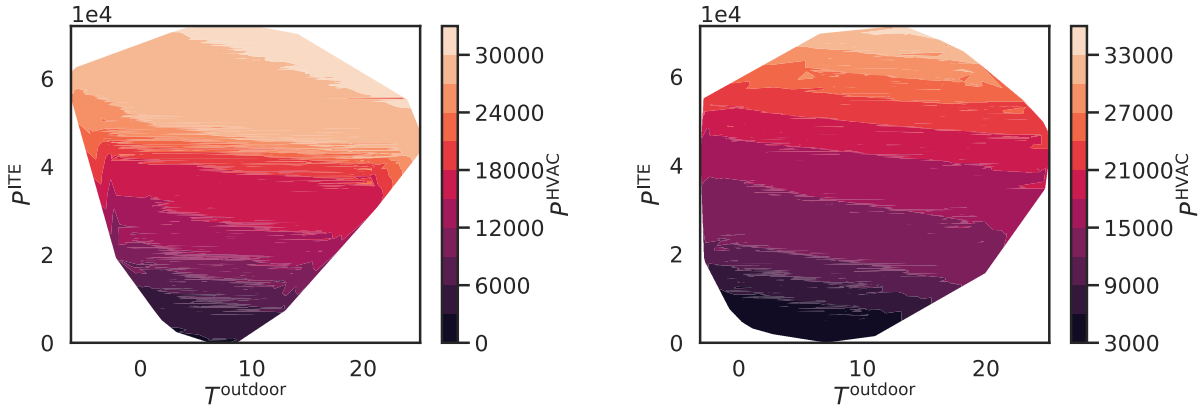
4.8.3 Analysis of the RL based controllers

In this subsection, an in-depth analysis of how the RL-based controllers can outperform the baseline controller is performed. First, the distribution of the HVAC power on a yearly basis is analyzed. Then, the control strategy which is learned by each agent is analyzed.

4.8.3.1 Yearly HVAC power distribution

HVAC power compared to disturbances Figure 4.27 shows contour plots of the HVAC power against the disturbance variables. As can be seen, the BA agent increases the HVAC power more gradually when the IT load increases, meaning the system is less often running at full power. The same pattern is seen for the other agents. This indicates that the energy savings are mainly made at higher IT loads.

HVAC energy per component To better understand how the agents can spread the energy usage better, Figure 4.28 shows the average power used by each HVAC component for every type of controller. Interesting to see is that the agents mainly save energy on the AL fan, at



(a) The HVAC power in the baseline

(b) The HVAC power in the BA

Figure 4.27: Contour plots of the HVAC power against the disturbance variables,

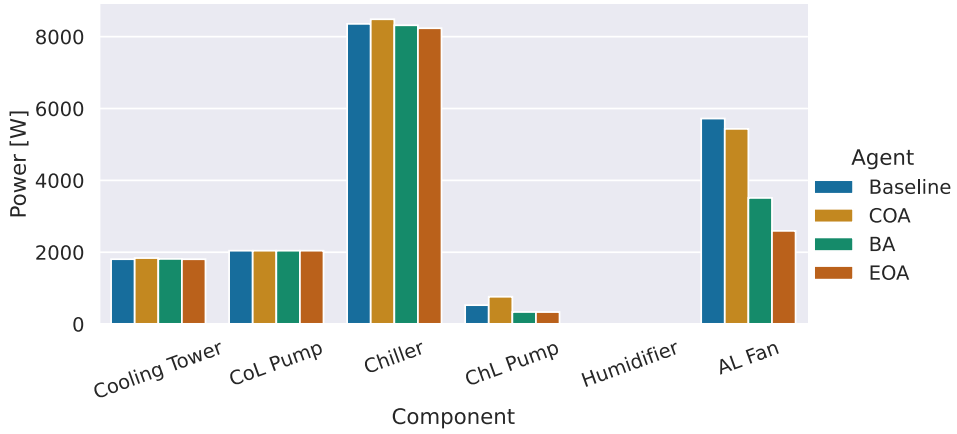


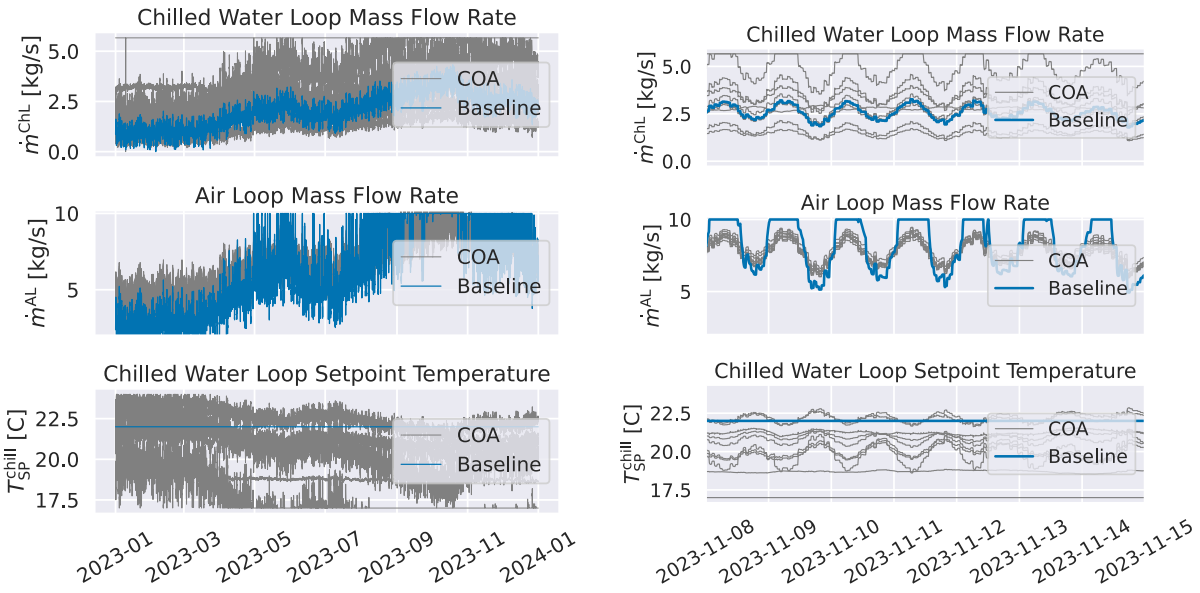
Figure 4.28: The mean HVAC component power in the test set for each agent.

the expense of some chiller power. When recalling the results of the baseline, the AL fan was always running at full power for high P^{ITE}, apparently, the RL agents have found a way to avoid this.

4.8.3.2 Learned control strategies

Up to now, we have concluded that the RL agents are better at handling the upper limit of $\dot{m}^{AL}$. To better understand how this is achieved, the control strategies learned by each agent are explained here. This is done by first showing the different strategies an agent learns for different seeds, followed by an analysis focusing on the single week analyzed in Figure 4.7b of the baseline controller. For figures containing the complete yearly behaviour of the controllers, the reader is referred to Appendix D. These are not shown here to reduce the length of the thesis, and generally no large differences between summer and winter were found.

Control strategies for different seeds As the seed sensitivity leads to a wide range in performance, it also leads to a range of different control strategies. This is illustrated in Figure 4.29, where all action variables of the different COA controllers are displayed. Especially $\dot{m}^{ChL}$ and



(a) Action variables of the COA trained with 10 different seeds.

(b) Action variables of the COA trained with 10 different seeds, shown for 1 week.

Figure 4.29: The behaviour of the COA for the different seeds.

T_{SP}^{ChL1} The exact cause of this is a subject for other research. For now, the trained controller with the most average behaviour is selected for further analysis of each agent type.

Constraint-Optimized Agent control strategy Figure 4.30 displays the behaviour of the action variables, HVAC power, T^{AL2} compared to T_{constr}^{AL2} and the disturbance variables of the COA compared to the baseline controller. It can be seen that the COA uses a slightly *higher* $\dot{m}^{ChL}$ than the baseline. Next to that, it modifies T_{SP}^{ChL1} according to the ITE load. When P^{ITE} increases, the controller decreases the chiller setpoint temperature. This combination decreases the required air loop mass flow, which ensures that the maximum mass flow is not reached. Next to that, the HVAC power is also lower when $\dot{m}^{AL}$ is lower. It is unknown if this daily modification to the setpoint is feasible for real chillers, as it is not done in practice, and also no research on it is performed to the best of our knowledge. However, the changes are quite small, and also relatively slow, so for now it is assumed to be possible.

When comparing the mass flows to Figure 4.28, it becomes clear what causes the power differences. The chilled water mass flow is slightly higher than that of the baseline, increasing $P^{\text{Chiller Pump}}$ slightly. The mass flow in the AL is sometimes lower, and sometimes higher than that of the baseline, which in the end leads to a small energy saving.

Interestingly, T^{AL2} is always kept below its constraint, ensuring a buffer for unexpected events. This is caused by the fact that the softplus penalty function is employed for the reward of this agent, which leads to the controller trying to keep the setpoint temperature slightly lower than the constraint temperature.

To summarize, the COA uses an adaptive chilled water setpoint, which allows $\dot{m}^{AL}$ to be lower. This prevents the mass flow from reaching its limit, saves energy, and still prevents constraints

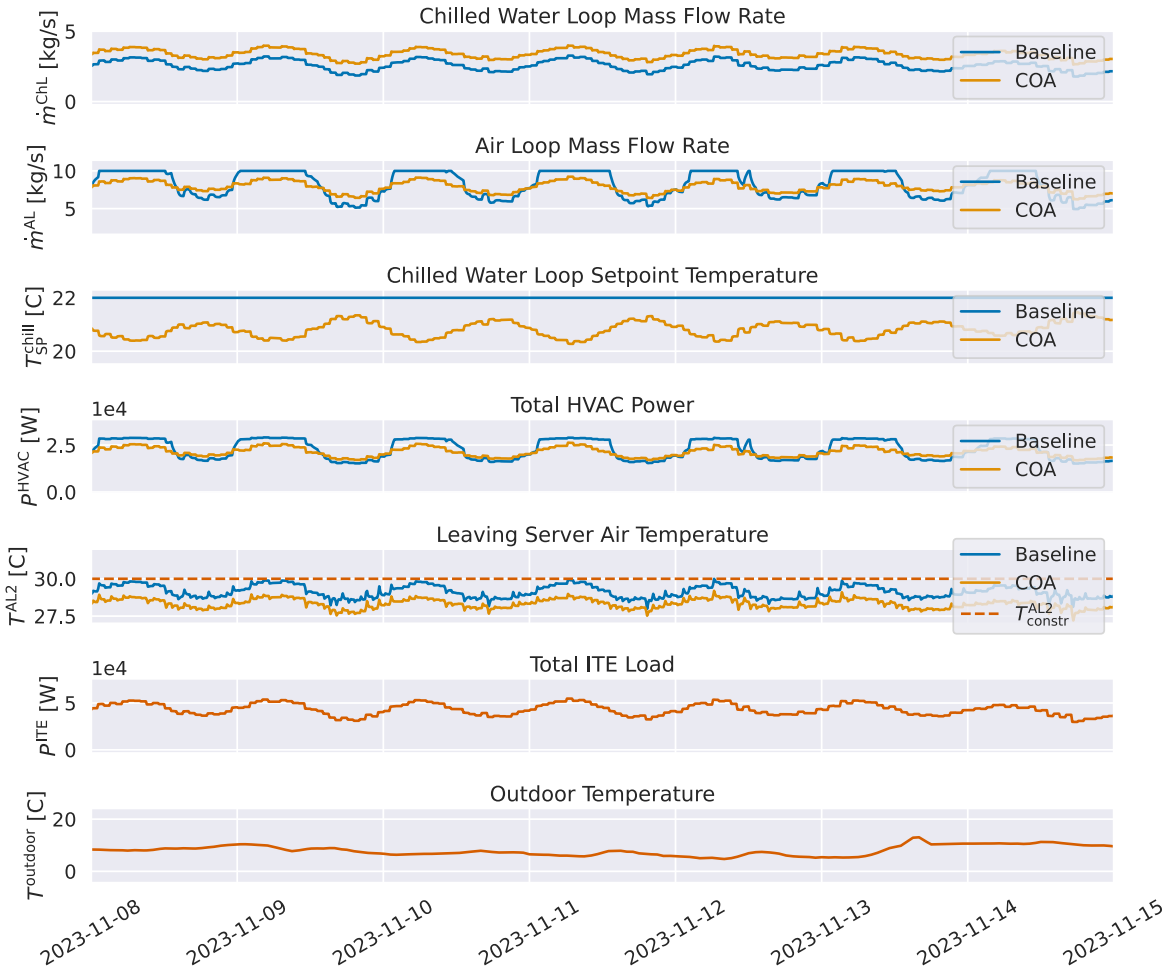


Figure 4.30: The COA control strategy compared to the baseline in the week of the 8^{th} of November.

from being violated.

Balanced Agent control strategy Figure 4.31 shows the control strategy of the BA. Interestingly, this controller has learned a completely different control strategy than the COA. This controller uses a low, slightly varying value for T_{SP}^{ChL1}. This cold temperature allows for a large temperature difference in the CC, which means the mass flows of both the ChL and AL can be lower than for the baseline or COA, while the chiller does not use much more power. This strategy saves power in the pump and fan and ensures that the maximum air mass flow is not reached.

In this controller, T^{AL2} is allowed to be much closer to the constraint temperature than in the COA.

Energy-Optimized Agent control strategy Finally, the control strategy of the EOA is shown in Figure 4.32. This strategy is similar to the strategy of the BA, with the large difference being that T_{SP}^{ChL1} is entirely at the bottom of the allowable range. This is likely caused by the linear

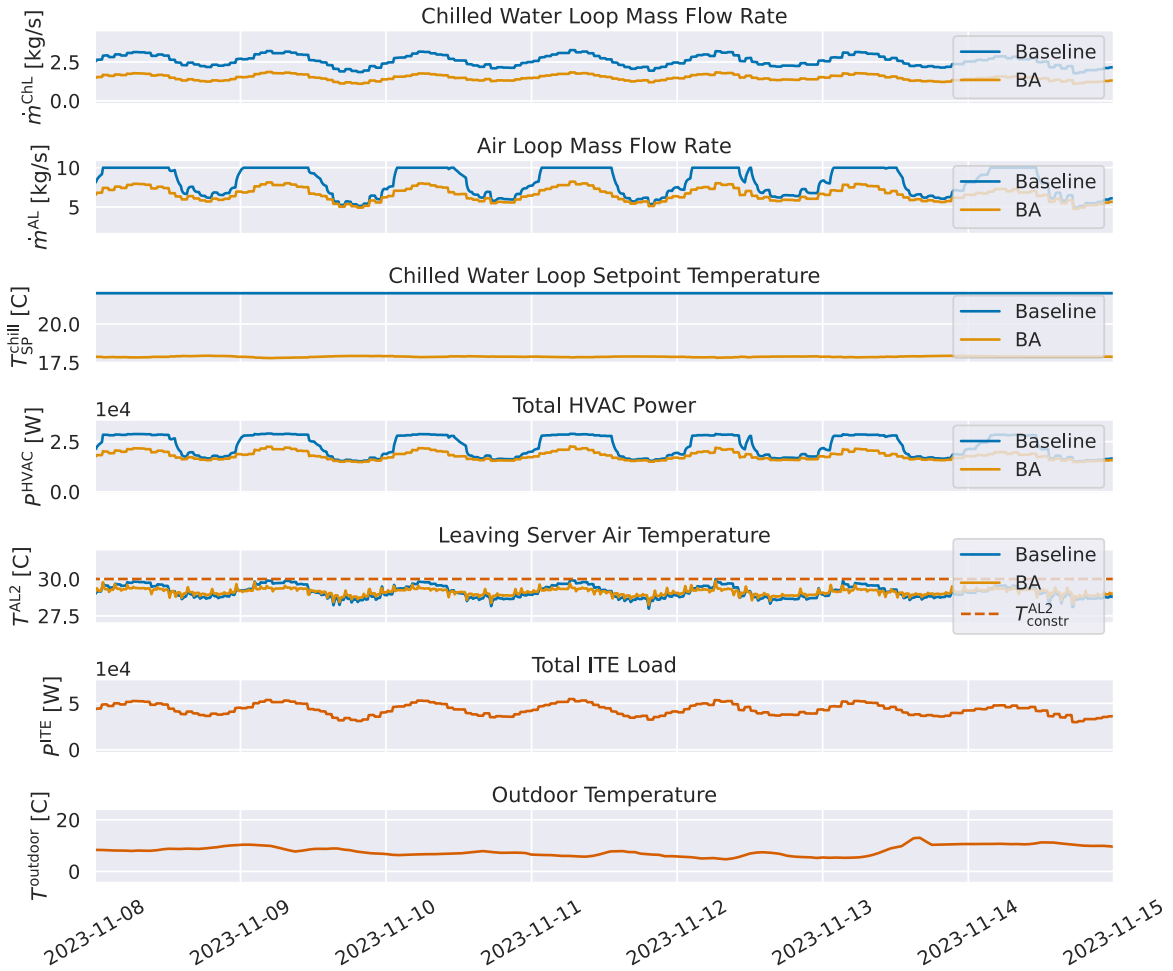


Figure 4.31: The BA control strategy compared to the baseline in the week of the 8^{th} of November.

penalty on action fluctuation for this agent. This means that the advantage of changing the chilled water setpoint compared to the previous timestep is always penalized. The most efficient continuous setpoint for the RL controller is at the lower end of the action space. This allows the mass flows to be even lower, thus decreasing power usage by the Air Fan and Chiller Pump. Next to this, T^{AL2} is consistently closer to the constraint temperature. This is the most efficient operating point of the system, however, it does make it less robust to changes in the disturbances.

4.8.4 Discussion & conclusion on the RL performance

To summarize, the 3 tuned RL agents behave as expected regarding P^{HVAC} during training. Regarding the handling of T_{constr}^{AL2}, the agents show a wide spread, dependent on the initial seed. Similar sensitivity to the initialization is seen on the validation set. However, all of the agents do outperform the baseline. The controllers have also been evaluated on the test set and still outperformed the baseline controller in the manner expected. This indicates that the trained controllers are robust to the disturbance variables and did not overfit on the training and vali-

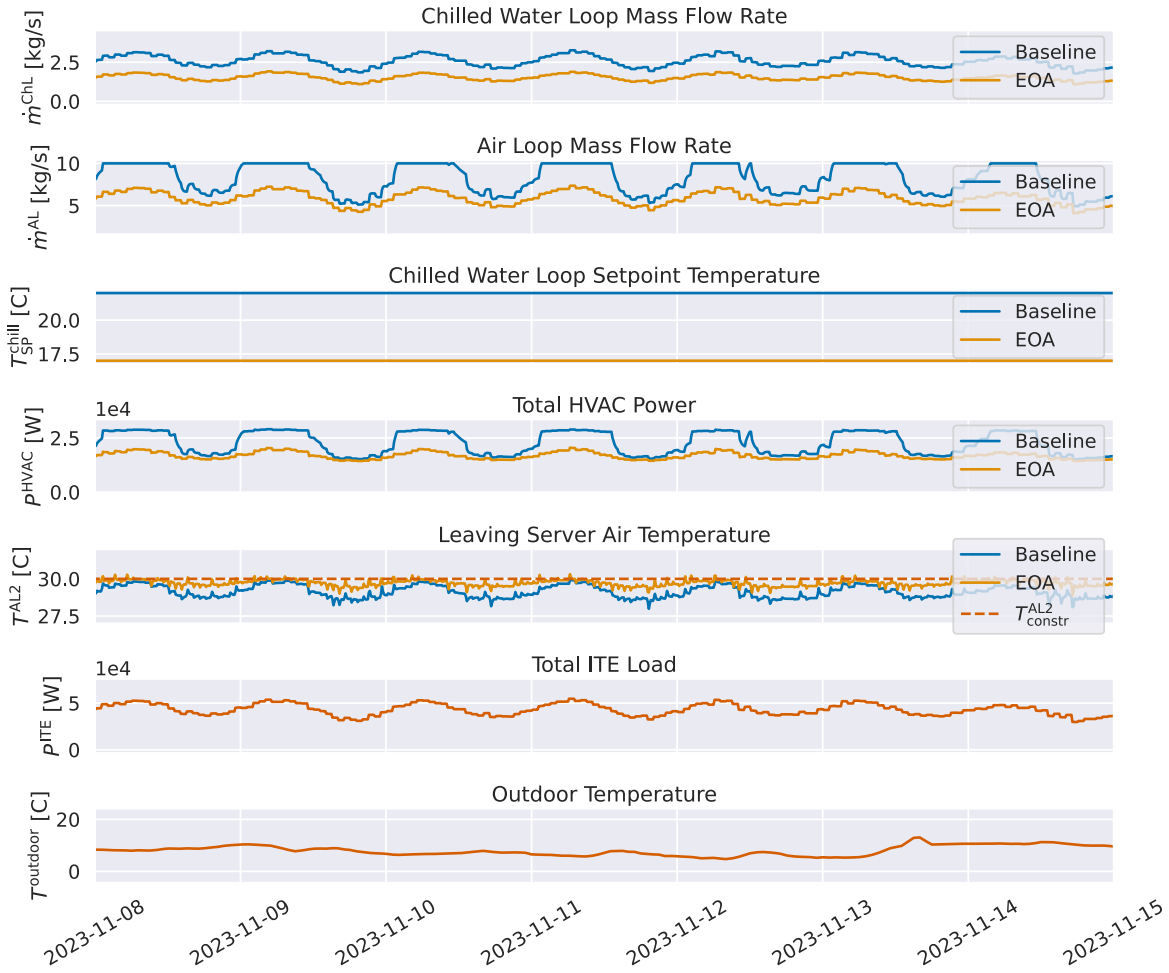


Figure 4.32: The EOA control strategy compared to the baseline in the week of the 8^{th} of November.

dation data.

An in-depth analysis of the RL based controllers showed that these controllers mainly save energy by decreasing the mass flows of the system. Analysis of the control strategy during a week indicated that two types of strategies are learned by the agents. Either adjusting T_{SP}^{ChL1} dynamically or making it very low to allow the mass flow rates to reduce. Since the mass flow rates are proportional to the power used by these components, this reduces energy usage.

Overall, the agents showed a good performance on all data. All 3 controllers were shown to generalize well to different disturbance variables. The largest drawback of the tuned agents is their sensitivity to their initialization. Because of this, one can not predict if an agent that is trained and will employed on a DC HVAC system will greatly improve the constraint handling of the system, or perform similar or even worse than a baseline controller.

4.9 Summary

This section has presented the numerical results of the experiments outlined in this thesis.

First, the sensitivity of the EnergyPlus model to its timestep size was investigated in section 4.1. This study concluded that the model accuracy does not suffer greatly from changes in the timestep size, but the execution time is influenced by its size. A timestep of 10 minutes has been selected to be used in the remainder of the thesis.

Following this, simulations using the baseline controller have been performed in section 4.3. From these experiments, it was found that most power is occupied by the air fan and chiller and that the baseline reaches the maximum mass flow rate of the fan for high ITE loads. It is not able to work around this limitation and thus violates constraints in these situations.

After the baseline was set, first several initial experiments with the RL-based controller were performed in section 4.4. Following this, the algorithm hyperparameter tuning experiments have been performed in section 4.5. From these experiments, a set of optimally performing hyperparameters has been found. Next to that, also hyperparameters which should be avoided have been found.

Using the tuned algorithm, the reward tuning experiment was conducted in section 4.6. This experiment found that RL-based controllers can outperform the baseline regarding the deviation of $T_{\text{SP}}^{\text{Al1}}$ and almost always perform well on action fluctuations. For the most important parameters, P^{HVAC} and $T_{\text{constr}}^{\text{AL2}}$, a trade-off should be made. This is mainly influenced by $\lambda_{T_{\text{SP}}}$ and p_T. From the tuning experiments, 3 controllers have been selected for further analysis.

In the tuning experiments, indications were found that the RL-based controller was sensitive to its initial seed. Therefore, a seed sensitivity study was conducted in section 4.7 which underlined these findings.

Finally, the 3 agents were compared on the validation and test data in section 4.8. For this, the seed sensitivity was also taken into account. It was found that, while the controllers outperform the baseline controller, their performance is dependent on the initialization of the agent. Finally, it was found that the agents learned 2 types of control strategies. On similar to the baseline controller, but with a fluctuating chilled water temperature. The other strategy learned to have a low chilled water setpoint temperature, leading to reductions in the mass flows required by the chilled water and air loops.

The next chapter will present the conclusion of this thesis, reflect on the limitations of the research, and propose a number of directions for future research.

Chapter 5

Conclusion

This chapter presents the conclusion on the results of this thesis and reflects its limitations. Next to that, a number of future research directions are proposed.

This thesis investigates the application of an RL-based controller to reduce HVAC power in DCs while being able to handle critical temperature constraints. It explores how differences in the reward function affect the control strategy of an agent.

This has been approached by defining a simple HVAC system and a formal control optimization problem, where P^{HVAC} should be minimized and temperature constraints should be met. For this system, both a baseline controller and a RL framework for the training of RL-based controllers have been proposed. Both of these controllers have been evaluated on an EnergyPlus simulation model.

From the evaluation of the baseline controller followed that the baseline controller generally performs well, but can not work around situations where the fan mass flow rate reaches the maximum value of its equipment. This led to high power usage and the violation of temperature constraints in a number of situations.

The proposed RL framework has been trained and tuned. This is done using data partitioning for the disturbance variables, to prevent overfitting. From this followed that RL-based controllers can outperform the baseline controllers. However, a trade-off between reducing P^{HVAC} and $T^{\text{AL2}}_{\text{constr}}$ violations is required. The reward function influences this trade-off between efficiency and handling constraints by changing the weight of the penalty on violating temperature constraints and modifying the penalty function itself. A Softplus function motivates the controller to keep a buffer between the air temperature and its constraint, making it more robust to disturbances, while a ReLU penalty function removes this buffer, making the agent more energy efficient by allowing the temperature differences in the system to increase, leading to less power usage.

From these results, 3 controllers were selected for further analysis and comparison to the baseline. These comparisons showed that the agents can outperform the baseline, but their final performance is sensitive to the initialization of the agents and initial exploration. This leads to varying agent behaviour after training for the same hyperparameters.

To conclude, this thesis shows that RL-based controllers indeed have the potential to improve

the efficiency of a simple HVAC system of a DC, while being able to handle temperature constraints. The agents have shown to be robust to new situations in the disturbance variables, but are sensitive to their initialization during training. From the reward tuning follows that the trade-off between reducing HVAC power and preventing the leaving server temperature from exceeding its constraints is mainly affected by the weight and type of the penalty function for the constraint violations.

5.1 Limitations

While this research offers a number of valuable insights, it is not without limitations. The main limitations of the research are discussed in this section.

HVAC system simplicity The first limitation is that the proposed HVAC system has a very simple layout. It has just a single chiller, CT and CRAH. This was chosen to keep the initial research simple, and also with the implementation in EnergyPlus in mind. However, real DCs have more complex HVAC systems, containing more rooms and a larger HVAC network. This will make controlling such a system increasingly difficult.

Modelling simplifications Another large limitation is the simplifications made in EnergyPlus. For example, there are no time delays in the system. This means equipment can instantly be turned on or off, and the maximum capacity of equipment can be accessed instantly. This enables controllers to just consider the current situation, without looking ahead at upcoming situations. This is likely the cause of the low values of γ found in the hyperparameter tuning.

Baseline controller The baseline controller is relatively simple. A more sophisticated baseline controller could have improved its performance, for example by being able to handle maximum mass flows better.

Constraint violation during training The comparison of the RL-based controllers is performed by comparing their performance after training on testing data for the disturbances. However, what is not considered is how much these controllers violate temperature constraints during their training, while temperatures of 30°C above the constraint were seen. This is acceptable in this study since it works with a model. However, when employing RL on a real, physical DC, these violations are unacceptable.

Agent learning robustness The agents proposed here are very sensitive to their initial seed. This is of course very bad for the implementation of the agents, as their trained performance can not be accurately predicted when an agent would be implemented.

5.2 Future research

This thesis shows that RL-based controllers have the potential to reduce energy while preventing constraint violations in a DC's HVAC system. However, before RL can be applied to the HVAC system of a real DC, a lot of additional research is required. During this research, 5 areas have been identified on which additional research is required to achieve this goal. Some of these directions are directly based on the limitations of this research, while others are directions that have not been addressed in this thesis.

Model realism As already has been discussed in the limitations, the EnergyPlus model has large simplifications. Although it proved to be sufficient for investigating the use of RL in DCs, it only captures the general trends of the behaviour of the HVAC system. To further investigate how RL agents behave in DCs, research on more realistic models is required. This includes both models that are better related to the real physics, thus including e.g., hot spots in a server room or delays in the system and models that capture the complexity of the system of a DC better, thus including more thermal zones and more HVAC components.

Safe RL In the limitations, it was seen that temperature constraints are violated severely during training. This should be improved before RL-based controllers can be employed in reality. The field of RL that focuses on minimizing constraint violations during the training of the agent is SRL. Thus, additional research on using SRL for DC HVAC systems is required.

Agent robustness In this thesis, it was already seen that the RL agents are quite sensitive to their initial seeds and hyperparameters. Another important topic that has not been discussed yet is the robustness to changes between the simulation model itself and reality, the so-called simulation-to-reality gap. Before this type of controller could be applied in reality, more research into its robustness to both HVAC model and RL agent parameter changes is required.

Increasing sample efficiency Although the sample efficiency of the RL agents is already quite good and allows for fast training on a model, it takes still quite a long time to converge regarding the simulated time. When a RL-based controller would be implemented in reality, it has to adapt to differences between the model and reality. To improve the speed of this adaptation, more research into increasing the sample efficiency of RL agents will be required.

Agent explainability The explainability of an agent's actions has not been a focus of this thesis. The initial goal was to develop a well-performing RL agent, with the expectation that explainability would become relevant once the agent's performance is satisfactory.

Before RL can be deployed in DCs, it is however important to gain the approval of many stakeholders. Given the mission-critical nature of data centers, it must be absolutely clear that the agent will not perform any dangerous actions. Understanding the reasoning behind an agent's decisions can significantly help in reassuring stakeholders about the agent's reliability and safety.

Moreover, researching the explainability of RL agents can uncover valuable insights that lead to the development of new best practices for conventional controllers. These best practices could be implemented in data centers a long time before RL agents are sufficiently stable to be deployed in the real world. For instance, if an agent demonstrates that adjusting setpoint temperatures based on outdoor conditions can save energy, a simple controller could be designed to follow this behavior. However, a thorough understanding of the agent's decision-making process is essential before such implementations can be realized.

Bibliography

[1] Stijn Grove et al. *European outlook*. Tech. rep. Kickstart Europe, Feb. 2024. URL: www.kickstartconf.eu.

[2] Laura Cozzi, Tim Gould, and Stephanie Bouckaert. *World energy outlook 2023*. Tech. rep. International Energy Agency (IEA), 2023. URL: www.iea.org/terms.

[3] Eren Cam et al. *Electricity 2024 - analysis and forecast to 2026*. Tech. rep. International Energy Agency (IEA), Jan. 2024. URL: www.iea.org.

[4] Emissions Database for Global Atmospheric Research (EDGAR). *Country fact-sheet: United Kingdom*. URL: https://edgar.jrc.ec.europa.eu/country_profile/GBR (visited on 02/20/2024).

[5] Jiacheng Ni and Xuelian Bai. "A review of air conditioning energy performance in data centers". In: *Renewable and Sustainable Energy Reviews* 67 (Jan. 2017), pp. 625–640. ISSN: 18790690. DOI: 10.1016/j.rser.2016.09.050.

[6] T T Chow et al. "Global optimization of absorption chiller system by genetic algorithm and neural network". In: *Energy and Buildings* 34 (June 2002), pp. 103–109.

[7] Zhenjun Ma and Shengwei Wang. "An optimal control strategy for complex building central chilled water systems for practical and real-time applications". In: *Building and Environment* 44 (2009), pp. 1188–1198. DOI: 10.1016/j.buildenv.2008.08.011.

[8] Yung-Chung Chang. "A novel energy conservation method-optimal chiller loading". In: *Electric Power Systems Research* 69 (2004), pp. 221–226. DOI: 10.1016/j.epsr.2003.10.012.

[9] Lu Lu et al. "HVAC system optimization-in-building section". In: *Energy and Buildings* 37 (Dec. 2004), pp. 11–22. DOI: 10.1016/j.enbuild.2003.12.007.

[10] B C Ahn and J W Mitchell. "Optimal control development for chilled water plants using a quadratic representation". In: *Energy and Buildings* 33 (2001), pp. 371–378.

[11] Jian Sun and Agami Reddy. "Optimal control of building HVAC&R systems using complete simulation-based sequential quadratic programming (CSB-SQP)". In: *Building and Environment* 40 (2005), pp. 657–669. DOI: 10.1016/j.buildenv.2004.08.011. URL: www.elsevier.com/locate/buildenv.

[12] Fernando Martínez-García et al. "Adaptive predictive control of a data center cooling unit". In: *Control Engineering Practice* 107 (Feb. 2021). ISSN: 09670661. DOI: 10.1016/j.conengprac.2020.104674.

[13] Seyed Morteza Mirhoseininejad, Ghada Badawy, and Douglas G. Down. "A data-driven, multi-setpoint model predictive thermal control system for data centers". In: *Journal of Network and Systems Management* 29.1 (Jan. 2021). ISSN: 15737705. DOI: 10.1007/s10922-020-09574-5.

[14] Nevena Lazic et al. "Data center cooling using model-predictive control". In: *32nd Conference on Neural Information Processing Systems*. Montreal, 2018.

[15] Kamran Fouladi et al. "Optimization of data center cooling efficiency using reduced order flow modeling within a flow network modeling approach". In: *Applied Thermal Engineering* 124 (2017), pp. 929–939. DOI: 10.1016/j.applthermaleng.2017.06.057. URL: http://dx.doi.org/10.1016/j.applthermaleng.2017.06.057.

[16] G.J. Levermore. *Building energy management systems*. 2nd ed. London: E & FN Spon, May 2000. ISBN: 9780203477342. DOI: 10.4324/9780203477342. URL: www.efnspon.com.

[17] Michael Deru et al. *Innovations in sensors and controls for building energy management: research and development opportunities report for emerging technologies*. Tech. rep. DOE, BTO & NREL, Feb. 2020. DOI: http://dx.doi.org/10.2172/1601591.

[18] Liang Yu et al. "A Review of Deep Reinforcement Learning for Smart Building Energy Management". In: *IEEE Internet of Things Journal* 8.15 (Aug. 2021), pp. 12046–12063. ISSN: 23274662. DOI: 10.1109/JIOT.2021.3078462.

[19] Y. G. Wang, Z. G. Shi, and W. J. Cai. "PID autotuner and its application in HVAC systems". In: *Proceedings of the American Control Conference* 3 (2001), pp. 2192–2196. ISSN: 07431619. DOI: 10.1109/ACC.2001.946075.

[20] Jerry Luo et al. *Controlling Commercial Cooling Systems Using Reinforcement Learning*. Tech. rep. Dec. 2022. URL: http://arxiv.org/abs/2211.07357.

[21] Timothy I Salsbury. "A survey of control technologies in the building automation industry". In: *IFAC Proceedings Volumes* 38.1 (2005), pp. 90–100. DOI: https://doi.org/10.3182/20050703-6-CZ-1902.01397. URL: https://www.sciencedirect.com/science/article/pii/S1474667016374092.

[22] Shengwei Wang and Zhenjun Ma. "Supervisory and optimal control of building HVAC systems: a review". In: *HVAC and R Research* 14.1 (Jan. 2008), pp. 3–32. ISSN: 10789669. DOI: 10.1080/10789669.2008.10390991.

[23] David Weinberg et al. "A Review of Reinforcement Learning for Controlling Building Energy Systems From a Computer Science Perspective". In: *Sustainable Cities and Society* 89 (Feb. 2023), p. 104351. ISSN: 2210-6707. DOI: 10.1016/J.SCS.2022.104351.

[24] Chi Zhang et al. "Building HVAC scheduling using reinforcement learning via neural network based model approximation". In: *BuildSys 2019 - Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. Association for Computing Machinery, Inc, Nov. 2019, pp. 287–296. ISBN: 9781450370059. DOI: 10.1145/3360322.3360861.

[25] Albin Heimerson et al. "Adaptive Control of Data Center Cooling using Deep Reinforcement Learning". In: *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*. 2022, pp. 1–6.

[26] Yuanlong Li et al. "Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning". In: *IEEE Transactions on Cybernetics* 50.5 (Sept. 2020), pp. 2002–2013. URL: http://arxiv.org/abs/1709.05077.

[27] Tianshu Wei, Yanzhi Wang, and Qi Zhu. "Deep reinforcement learning for building HVAC control". In: *Proceedings - Design Automation Conference*. Vol. Part 128280. Institute of Electrical and Electronics Engineers Inc., June 2017. ISBN: 9781450349277. DOI: 10.1145/3061639.3062224.

[28] Basil Kouvaritakis and Mark Cannon. *Advanced textbooks in control and signal processing series editors*. Ed. by Michael J Grimble and Michael A. Johnson. Springer, 2016. ISBN: 978-3-319-24851-6. DOI: 10.1007/978-3-319-24853-0. URL: http://www.springer.com/series/4045.

[29] K S Holkar, K K Wagh, and L M Waghmare. "An overview of model predictive control". In: *International Journal of Control and Automation International Journal of Control and Automation* 3.4 (2010).

[30] Manfred Morari and Jay H Lee. "Model predictive control: past, present and future". In: *Computers and Chemical Engineering* 23 (1999), pp. 667–682.

[31] Nevena Lazic et al. "Data center cooling using model-predictive control". In: *32nd Conference on Neural Information Processing Systems*. Google. Montreal: NeurIPS, 2018.

[32] Qingxia Zhang et al. "A survey on data center cooling systems: Technology, power consumption modeling and control strategy optimization". In: *Journal of Systems Architecture* 119 (Oct. 2021). ISSN: 13837621. DOI: 10.1016/j.sysarc.2021.102253.

[33] Richard S Sutton and Andrew G Barto. *Reinforcement Learning An Introduction second edition*. 2nd ed. London: The MIT Press, 2018. ISBN: 978-0-262-19398-6.

[34] Steven L Brunton and J Nathan Kutz. *Data-Driven Science and Engineering Machine Learning, Dynamical Systems, and Control*. 2nd ed. Washington: Cambridge University Press, 2021, pp. 504–540.

[35] Youssef Fenjiro and Houda Benbrahim. "Deep reinforcement learning overview of the state of the art". In: *Journal of Automation, Mobile Robotics and Intelligent Systems* 12.3 (Nov. 2018), pp. 20–39. ISSN: 20802145. DOI: 10.14313/JAMRIS{_}3-2018/15.

[36] Christian Blad, Simon Bøgh, and Carsten Skovmose Kallesøe. "Data-driven Offline Reinforcement Learning for HVAC-systems". In: *Energy* 261 (Dec. 2022). ISSN: 0360-5442. DOI: 10.1016/J.ENERGY.2022.125290.

[37] Hsin Yu Liu et al. "Safe HVAC Control via Batch Reinforcement Learning". In: *Proceedings - 13th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2022*. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 181–192. ISBN: 9781665409674. DOI: 10.1109/ICCPS54341.2022.00023.

[38] Zhengbo Zou, Xinran Yu, and Semiha Ergan. "Towards optimal control of air handling units using deep reinforcement learning and recurrent neural network". In: *Building and Environment* 168 (Jan. 2020). ISSN: 03601323. DOI: 10.1016/j.buildenv.2019.106535.

[39] Ruihang Wang et al. "Green Data Center Cooling Control via Physics-Guided Safe Reinforcement Learning". In: *ACM Transactions on Cyber-Physical Systems* (Feb. 2023). ISSN: 2378-962X. DOI: 10.1145/3582577.

[40] Zhiwei Cao et al. "Toward Model-Assisted Safe Reinforcement Learning for Data Center Cooling Control: A Lyapunov-based Approach". In: *Proceedings of the 2023 14th ACM International Conference on Future Energy Systems*. Association for Computing Machinery, Inc, June 2023, pp. 333–346. ISBN: 9798400700323. DOI: 10.1145/3575813.3597343.

[41] Abdul Afram and Farrokh Janabi-Sharifi. "Review of modeling methods for HVAC systems". In: *Applied Thermal Engineering* 67.1-2 (2014), pp. 507–519. ISSN: 13594311. DOI: 10.1016/j.applthermaleng.2014.03.055.

[42] Drury B. Crawly et al. "EnergyPlus: energy simulation program". In: *ASHRAE Journal* (Apr. 2000), pp. 49–56.

[43] *EnergyPlus™ version 23.2.0 documentation engineering reference*. Tech. rep. 2023.

[44] Kaiyu Sun et al. "Prototype energy models for data centers". In: *Energy and Buildings* 231 (Jan. 2021). ISSN: 03787788. DOI: 10.1016/j.enbuild.2020.110603.

[45] Liang Li, Kyoko Hasegawa, and Satoshi Tanaka. "Reinforcement Learning Testbed for Power-Consumption Optimization". In: *Methods and Applications for Modeling and Simulation of Complex Systems*. Kyoto, Japan: AsiaSim, Oct. 2018, pp. 45–59. URL: http://www.springer.com/series/7899.

[46] Avisek Naug et al. "PyDCM: Custom Data Center Models with Reinforcement Learning for Sustainability". In: *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 23. ACM, Nov. 2023, pp. 232–235. ISBN: 9798400702303. DOI: 10.1145/3600100.3623732. URL: https://doi.org/10.1145/3600100.3623732.

[47] *ANSYS FLUENT user's guide*. Vol. 13.0. Canonsburg, PA: ANSYS, Inc., Nov. 2010.

[48] Ziynet Boz, Ferruh Erdogdu, and Mustafa Tutar. "Effects of mesh refinement, time step size and numerical scheme on the computational modeling of temperature evolution during natural-convection heating". In: *Journal of Food Engineering* 123 (2014), pp. 8–16. ISSN: 02608774. DOI: 10.1016/j.jfoodeng.2013.09.008.

[49] Abhijit S. Badwe et al. "Detection of model-plant mismatch in MPC applications". In: *Journal of Process Control* 19.8 (Sept. 2009), pp. 1305–1313. ISSN: 0959-1524. DOI: 10.1016/J.JPROCONT.2009.04.007.

[50] Cooltherm UK. *Water-Cooled Chillers*. URL: https://www.cooltherm.co.uk/products/chillers/water-cooled-chillers (visited on 04/08/2024).

[51] Yogesh Fulpagare and Atul Bhargav. "Advances in data center thermal management". In: *Renewable and Sustainable Energy Reviews* 43 (2015), pp. 981–996. ISSN: 18790690. DOI: 10.1016/j.rser.2014.11.056.

[52] *Equipment thermal guidelines for data processing environments*. Tech. rep. Peachtree Corners: ASHRAE, 2021.

[53] *What is a server rack: specifications, usage, history and more*. Tech. rep. AnD Cable Products Inc., 2023.

[54] Robert F. Sullivan. *Alternating cold and hot aisles provides more reliable cooling for server farms*. Tech. rep. 2000. URL: www.upsite.com/TUIpages/.

[55] O VanGeet, William Lintner, and Bill Tschudi. *Best practices guide for energy-efficient data center design*. Tech. rep. National Renewable Energy Laboratory (NREL), Mar. 2011. URL: http://www.thegreengrid.org/en/Global/Content/white-papers/ERE..

[56] AmCraft Manufacturing Inc. *Hot and Cold Aisles in Your Data Center: What to Know*. URL: https://datacenterenclosure.com/hot-and-cold-aisles-in-your-data-center-what-to-know/ (visited on 04/25/2024).

[57] "Electric chiller model based on condenser entering temperature". In: *EnergyPlus™ Version 23.2.0 Documentation Engineering Reference*. U.S. Department of Energy, Sept. 2023. Chap. 14.3.9, pp. 796–805.

[58] *EnergyPlus™ version 23.2.0 documentation input output reference*. Tech. rep. U.S. Department of Energy, 2023.

[59] "Chilled-water-based air cooling coil". In: *EnergyPlus™ Version 23.2.0 Documentation Engineering Reference*. 2023, pp. 821–836.

[60] Fadi Almahamid and Katarina Grolinger. "Reinforcement Learning Algorithms: An Overview and Classification". In: *Canadian Conference on Electrical and Computer Engineering*. Vol. 2021-September. Institute of Electrical and Electronics Engineers Inc., Sept. 2021. ISBN: 9781665448642. DOI: 10.1109/CCECE53047.2021.9569056.

[61] John Schulman et al. "Proximal policy optimization algorithms". In: (July 2017). URL: http://arxiv.org/abs/1707.06347.

[62] Nicolas Heess et al. "Emergence of locomotion behaviours in rich environments". In: *CoRR* (July 2017). URL: http://arxiv.org/abs/1707.02286.

[63] Richard S Sutton et al. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

[64] John Schulman et al. "High-dimensional continuous control using generalized advantage estimation". In: *ICLR*. 2016. ISBN: 1506.02438v6. URL: https://sites.google.com/site/gaepapersupp..

[65] Oikolab. *Weather Data Downloader*. URL: https://weatherdownloader.oikolab.com/app (visited on 03/18/2024).

[66] Yasser A. Ali et al. "Hyperparameter Search for Machine Learning Algorithms for Optimizing the Computational Complexity". In: *Processes* 11.2 (Feb. 2023). ISSN: 22279717. DOI: 10.3390/pr11020349.

[67] Baohe Zhang et al. "On the importance of hyperparameter optimization for model-based reinforcement learning". In: *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. San Diego, California, USA: PMLR, Apr. 2021, pp. 4015–4023. URL: http://arxiv.org/abs/2102.13651.

[68] James Bergstra et al. "Algorithms for Hyper-Parameter Optimization". In: *Advances in Neural Information Processing Systems 24 (NIPS 2011)*. Ed. by J. Shawe-Taylor et al. Grenada, Dec. 2011. ISBN: 9781618395993.

[69] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining*. July 2019. URL: http://arxiv.org/abs/1907.10902.

[70] Greg Brockman et al. *OpenAI Gym*. Tech. rep. OpenAI, 2016.

[71] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. Tech. rep. 2018. URL: http://rllib.io.

[72] Antonin Raffin et al. *Stable-Baselines3: Reliable Reinforcement Learning Implementations*. Tech. rep. 2021, pp. 1–8. URL: https://github.com/DLR-RM/stable-baselines3..

[73] Matthias Plappert. *keras-rl*. https://github.com/keras-rl/keras-rl. 2016.

[74] Itai Caspi et al. *Reinforcement Learning Coach*. Dec. 2017. DOI: 10.5281/zenodo.1134899. URL: https://doi.org/10.5281/zenodo.1134899.

[75] John Niemann, Kevin Brown, and Victor Avelar. *Hot-aisle vs. cold-aisle containment for data centers*. Tech. rep. APC by Schneider Electric, July 2010.

[76] USA Coil Air. *Water cooling and heating coils*. Jan. 2024. URL: https://usacoil.com/water-cooling-and-heating-coils/ (visited on 06/11/2024).

[77] NDetated. *Everything you need to know about water cooled chillers*. URL: https://cntcu.com/blogs/industrial-chiller/everything-you-need-to-know-about-water-cooled-chillers (visited on 06/18/2024).

[78] "One, two, and variable speed cooling towers and evaporative fluid coolers". In: *EnergyPlus™ Version 23.2.0 Documentation Engineering Reference*. 23.2.0. U.S. Department of Energy, Sept. 2023. Chap. 16.1.1, pp. 1026–1033.

[79] Inc. Delta Cooling Towers. *What is a cooling tower and how does it work?* URL: https://deltacooling.com/resources/faqs/what-is-a-cooling-tower (visited on 06/11/2024).

Appendix A

Chiller plant

This appendix contains more background information on a chiller plant.

A.1 Aisle containment strategies

Although the use of hot and cold aisles can already save up to 25% of energy when compared to a mixed aisle layout, there is still room for improvement in the layout. As the air streams are not divided by a physical barrier, mixing will still take place. This occurs for example at the end of the aisles, where the air streams flow out of the aisle and mix, or when hot air rises and moves over the racks to move to the cold aisle [75].

To prevent the mixing of air, a straightforward solution is to create a physical barrier between the hot and cold air streams. In most modern DCs, one of two different approaches are implemented, either hot or cold aisle containment [75]. These approaches share the same main advantage. As there is less mixing of air, the cold air temperature can be safely set to a higher value without having the risk of hotspots at locations where the cold and hot air mix. Rising the cold air temperature does not only save energy by requiring less cooling, there are also other advantages, such as the fact that there are less (de)humidification costs, since there is less humidity removed from the air at higher temperatures, and the possibility for better sized HVAC equipment [75].

In the case of cold aisle containment, the cold aisle is closed off from the rest of the room, as shown in Figure A.1. By containing the cold aisle, the rest of the room is allowed to become hot without any mixing of the 2 air streams. The fact that the main room is filled with hot air poses constraints on the maximum hot air temperature, because of the working conditions for the personnel in the room. This containment can be implemented by simply putting a roof on top of the cold aisle and closing both ends of the aisle (often with a curtain). Since it is so easy to implement, cold aisle containment is a popular choice to improve the efficiency of DCs which do not have any aisle containment strategies in place yet.
As the name suggests, hot aisle containment is exactly the opposite of cold aisle containment. Here, the hot aisles are closed off from the cold aisles, as shown in Figure A.2. The hot air flows to a space above the room, where it is sucked in by the CRAH units. These lead the cold air to the cold aisles, where it cools the server racks. Since the hot aisles are closed off, the temperature is not constraint by the maximum acceptable personnel working conditions anymore. Therefore, the air is allowed to become warmer, which increases the efficiency of the HVAC system even more [75]. Because of the space above the ceiling, it is harder to retrofit hot aisle

Figure A.1: Schematic representation of cold aisle containment [56] (Although the cold air is coming from the roof in this representation, it often comes through the perforated floor tiles in reality).

containment in existing DCs.

When comparing hot and cold aisle containment, it can be concluded that hot aisle containment is the most efficient of the 2, and therefore it is preferred in newly built DCs. However, cold aisle containment still shows large improvements in energy efficiency when compared to no air containment. Since it is easy to implement in existing DCs, it is the preferred method for retrofitting.

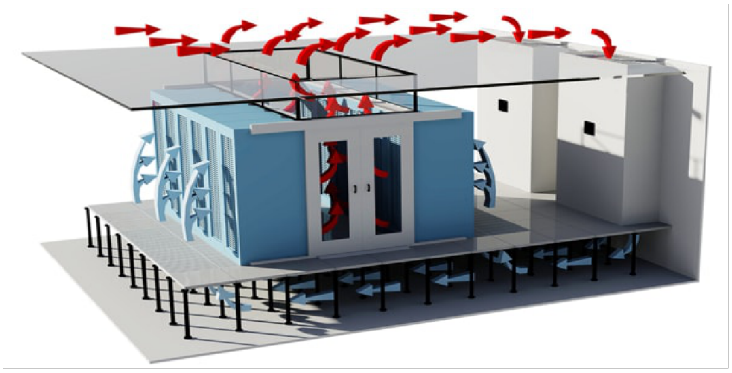


Figure A.2: Schematic representation of hot aisle containment [56].

A.2 HVAC components

Cooling Coil In the CRAH units in a DC, chilled water CCs are used to cool down the air. These coils consist of a series of pipes through which the chilled water flows. These pipes have cooling fins attached through which the air passes. A CC is shown in Figure A.3. The cooling capacity of these coils mainly depends on the mass flow rate and temperature difference between the water and the air passing through the coil. Next to that, the cleanliness of the surface area affects the heat transfer in the coil. For example, when there is moisture on the surface of the coil, its capacity will be reduced [59].

Figure A.3: A chilled water CC [76]

Air fans The fans in a CRAH control the air flow through a server room. Their speed is determining the pressure under the raised floor and therefore the air flow rate. There is a non-linear relationship between the air flow rate and the fan energy usage. Therefore, optimizing their energy usage can be challenging

Water pumps Similar to the fans, water pumps control the pressure and thus the water mass flow rate in the loops. There is also a non linear relationship between the mass flow rate and the power used by a water pump.

Chiller In a HVAC system, the chiller is the device that actively cools water using a refrigeration cycle. Therefore, it is a central component of the system and often also one of the most energy demanding components. The chiller ensures that heat can be extracted from a server room even when the outdoor temperature is higher than the desired room temperature.

There are a lot of different chillers, which all use a refrigeration cycle internally. A basic refrigeration cycle is shown in Figure A.4. This cycle uses a refrigerant with a relatively low boiling point. In the expansion valve, the pressure of the refrigerant is dropped so much that it starts to evaporate and cools down a lot. In the evaporator, heat is extracted from the outside world (in this case the chilled water loop). Then, using a compressor, the refrigerant is compressed until it becomes liquid again. During this liquefaction, the refrigerant heats up. Because of this higher temperature, the refrigerant is able to reject heat to an environment which has a higher temperature than the chilled water loop.

A refrigeration cycle is often categorised by its COP, which is defined as:

$$COP = \frac{Q_{in}}{W} \tag{A.1}$$

where Q_{in} is the heat energy extracted from the cold part of the system, and W is the work done by the compressor. COP is thus a measure of the efficiency of a refrigeration cycle. Chillers typically have a COP between 3.5 and 7 [77].
Different approaches exist for this rejection of heat, and chillers can be divided into 2 groups based on its method of heat rejection: air or water cooled chillers. Air cooled chillers reject their heat directly from the refrigerant to the outdoor air. Water cooled chillers use another water loop to reject their heat to. These other loops use a heat exchanger such as a cooling tower to reject their heat to the outdoor air.

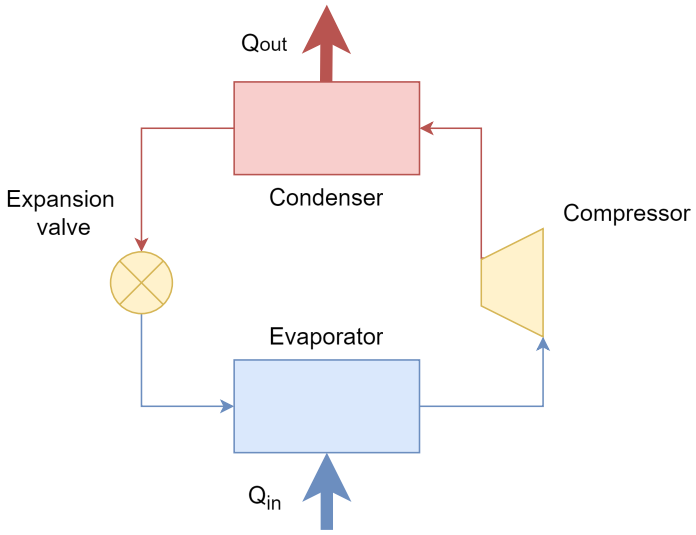


Figure A.4: Schematic overview of a refrigeration cycle.

The water cooled chillers are more efficient and generally have longer life spans than air cooled chillers [77]. The air cooled chillers are often cheaper than water cooled chillers, since they do not require an extra water loop and therefore require less equipment than water cooled chillers. Since energy efficiency is very important in DCs, most have water cooled chillers installed.

Cooling Tower Figure A.5 shows a schematic representation of a CT. This tower exchanges heat with the outdoor air by spraying the hot water on a filler material. Because of the increased surface area by the spray, the heat transfer rate is as high as possible. The cooled down water is then collected and send back to the chiller. In favourable conditions, the water cools down enough by natural convection, and the CT uses no energy. If natural convection does not provide enough cooling, the fan at the top of the CT can be turned on. This ensures air flows through the tower and the water is cooled by forced convection. This does require electricity of course [78].

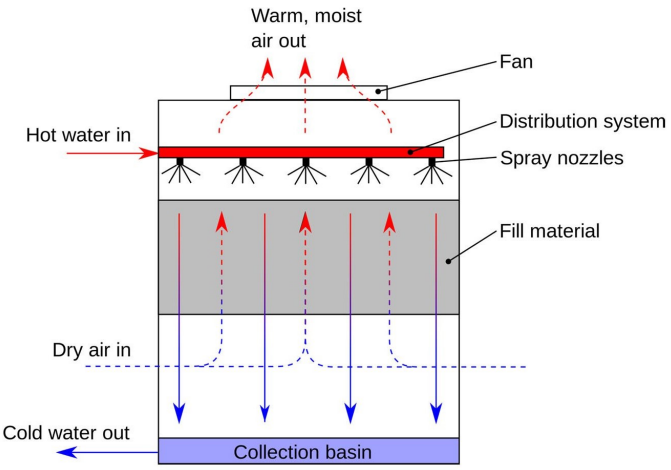


Figure A.5: Schematic working of a CT [79]

Appendix B

IT Load derivation

This appendix gives an in depth explanation on the ITE load, which is defined as P^{ITE}:

$$P^{\text{ITE}}(t) = P_{\text{base}}(t) + P_{\text{day,night}}(t) + P_{\text{noise}}(t) \tag{B.1}$$

where the sub components of the load are defined as follows:

$$P_{\text{base}}(t_i) = P_{\text{base}}(t_{i-1}) + P_{\text{base}}(t_0) \cdot X_{\text{base}}, \quad X_{\text{base}} \sim \mathcal{N}\left(0, \sigma^2_{\text{base}}\right) \tag{B.2}$$

where t_i denotes the i^th time step, $P_{\text{base}}(t_0)$ is a pre-defined value and X_{base} is sampled from a normal distribution with mean 0 and standard deviation σ^2_{base}, where σ_{base} again is a parameter which is pre-defined.

The daily variation of the power is defined as:

$$P_{\text{day,night}}(t) = X_{\text{day,night}} \cdot A_{\text{day,night}} \cdot \sin\left(\frac{2\pi}{24} \cdot t\right) \tag{B.3}$$

where A is the amplitude of this power fluctuation which can be defined by the user. $X_{\text{day,night}} \sim \mathcal{U}\left(1 - c_{\text{day,night}}, 1 + c_{\text{day,night}}\right)$ is resampled every 12 hours to add additional randomness to the system.

Finally $P_{\text{noise}}(t)$ is defined as Gaussian noise with a standard deviation σ^2_{noise}.

To summarize, the IT load values over the year can be tweaked by the input factors: $P_{\text{base}}(t_0)$, σ_{base}, $c_{\text{day,night}}$, $A_{\text{day,night}}$ and σ_{noise}.

When the the noise for each year has been generated, it is normalized between 0 and 1 to represent a fraction of the total IT load. This normalization is performed as follows:

$$P^{\text{ITE}}_{\text{normalized}}(t) = \frac{P^{\text{ITE}}(t) - \min(P^{\text{ITE}})}{\max(P^{\text{ITE}}) - \min(P^{\text{ITE}})} \tag{B.4}$$

The results of the generated IT loads for several years are shown in Figure B.1. It can be seen clearly here that the average IT load varies over the year, while never making too sudden, unpredictable jumps. A zoomed in plot spanning several days is shown in Figure B.2. In this plot, the daily variation and Gaussian noise are clearly visible. Also, it can be seen that the height of the peaks varies from day to day.

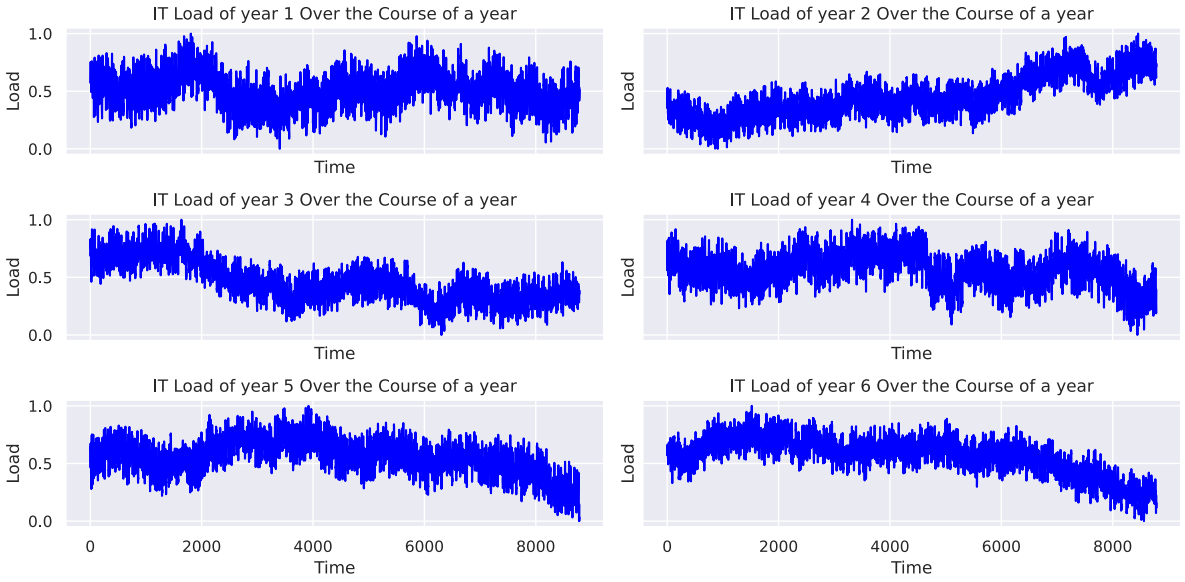


Figure B.1: The IT load over the course of a year

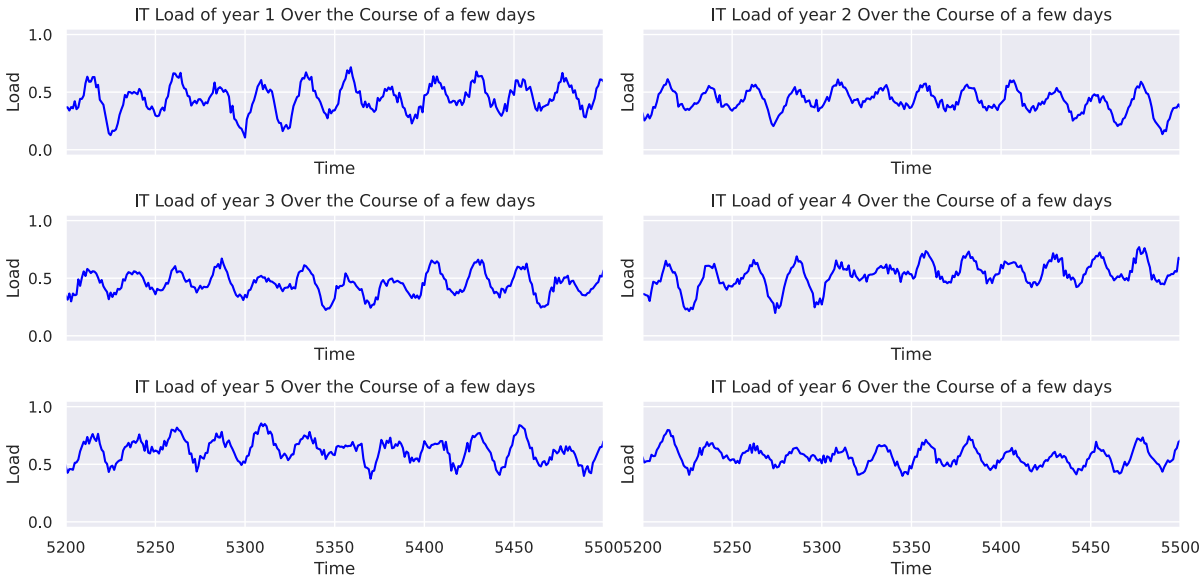


Figure B.2: The IT load over the course of several days

Appendix C

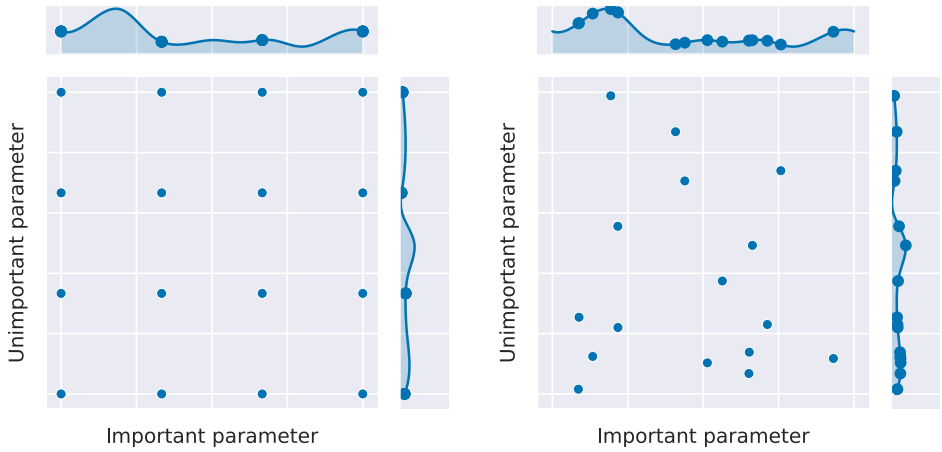
Hyperparameter samplers

This appendix contains a more in depth analysis of the most common hyperparameter sampling types. In Figure C.1, some common sampler types are visualized for sampling in 2D. In these examples, there is one hyperparameter which has a large influence on the performance of the model, while the other one has a small influence on the performance. The 4 sampler types are: grid sampling, random sampling, pseudo-random sampling and Bayesian sampling.

Grid search Figure C.1a shows a structured grid search. For such a grid search, the hyperparameters are divided into discrete intervals and all possible combinations in these intervals are sampled. Although the samples are quite well distributed over the search field, there are multiple large disadvantages to this type of search. Firstly, the amount of samples required for a good optimization increases exponentially with the amount of hyperparameters, making the method inefficient for higher dimensions. Secondly, the structure lacks randomness, as can be seen, large areas of the hyperparameter space are not explored using this kind of sampling. This becomes clear when one looks at the location of the sampling in the top distribution. Finally, a grid search does not adapt to intermediate results. For example, higher values for the important parameter lead to worse results, but after this becomes clear after some samples, the grid search will still keep sampling on this region.

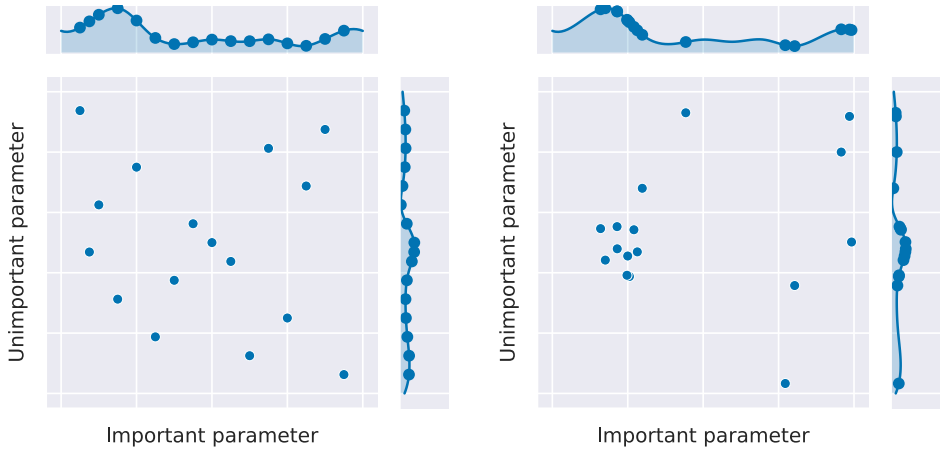
Random search Figure C.1b shows a random search. This already mitigates some of the problems related to a grid search. In this example, one hyperparameter has a lot more effect on the performance of the agent than the other. However, which parameter this is is not known in advance. Therefore, the random search effectively distributes its sampling better over the important parameter than a grid search could, while not affecting the performance by the unimportant parameter. There is a larger chance of finding a good performing configuration, as can be seen in the top distribution. A disadvantage of the random sampling is that there is still no adaptation to intermediate results. Another disadvantage is that there is the risk of clustering of samples. In Figure C.1b, this is seen at the left bottom, where there are a lot of samples close together, while there is just one sample in the top right corner.

Pseudo-random search To solve this problem of a possibly clustered or bad distribution by random search, pseudo-random samplers can be utilized. This class of samplers uses deterministic algorithms to sample sequences which seem to be random. This can be clearly seen in Figure C.1c, where a Sobol sampler has been used. When looking at the grid, these samples seem just as random as the samples from Figure C.1b. However, when looking at the distributions at the top and right, one can see that the samplers are distributed a lot better. Various



(a) A grid sampler with $n = 16$.

(b) A random sampler with $n = 16$.



(c) A Sobol sampler with $n = 16$.

(d) A Bayesian sampler with $n = 16$.

Figure C.1: Example of different samplers. The distributions on the top and right of the grids show the effect of changing this hyper parameter on the total performance of the model. In this case, one hyperparameter influences the performance of the model largely, while the other only has a small influence.

pseudo-random samplers, such as the Sobol sampler, Latin Hypercube, Halton sequences, and many others, are available for use. These samplers have the advantages of a random search with often a more structured distribution, which becomes increasingly important for larger dimensionality of the search space. However, the samplers still do not adapt to intermediate results.

Bayesian search What all the above samplers have in common is that they do not adapt to intermediate results. This leads to a decrease in sample efficiency, since a lot of samples are taken in unpromising regions of the search space. Especially in high dimensional search spaces, a lot of samples are required to cover the search space effectively. Bayesian samplers solve this problem by constructing a surrogate model, often using Gaussian processes. These models predict the performance of the hyperparameter configurations and give an estimate of the uncertainty

of the surrogate model. By focussing on promising areas, a lot more sampling can be done in these areas, as can be seen clearly in Figure C.1d, where most samples are taken in the most promising region. This is very beneficial for the sample efficiency, and thus the computational complexity. A disadvantage of Bayesian sampling is that it can focus on suboptimal regions when the initial surrogate model is very inaccurate. However, as the model updates with each new sample, this is often not a large problem in reality [68].

Appendix D

Tuned controller behaviour

This appendix contains figures with the behaviour of the tuned RL-based controller over the course of a year.

D.1 EOA

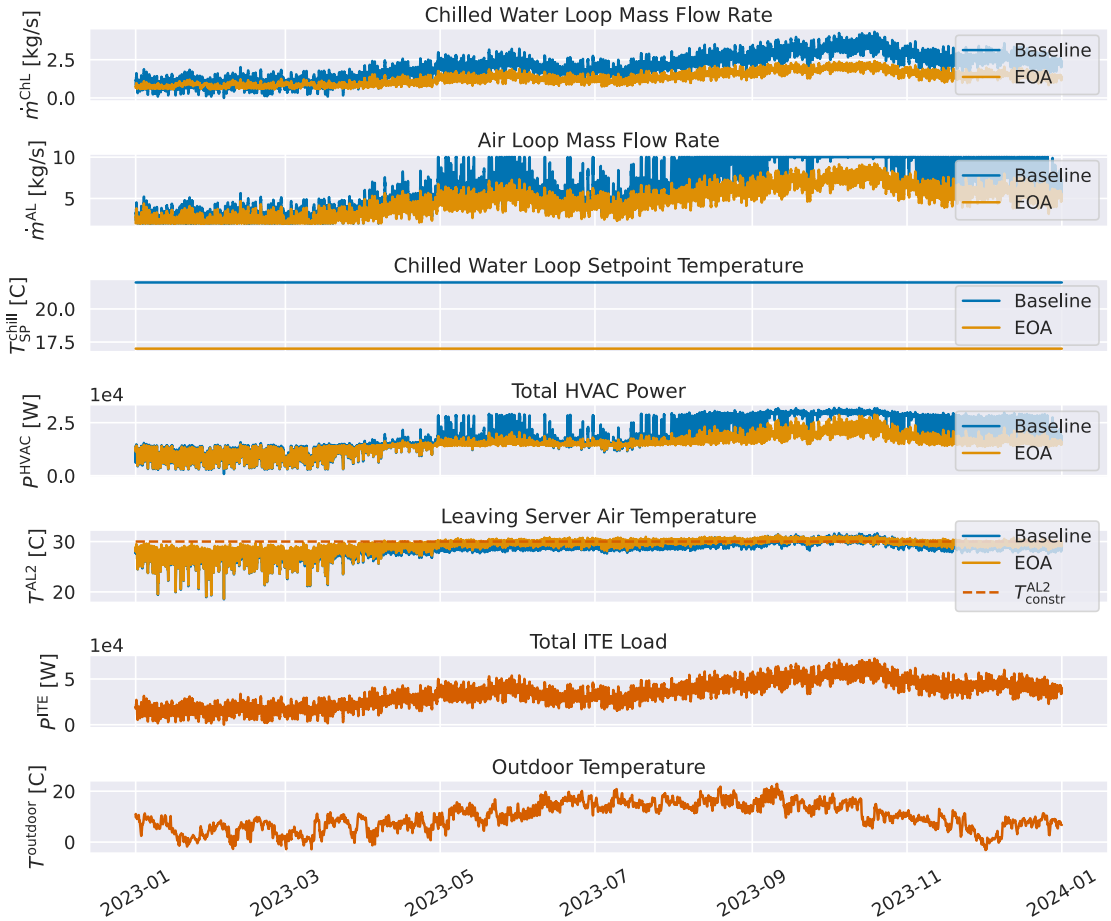


Figure D.1: The behaviour of the EOA over the course of a year, compared to the baseline.

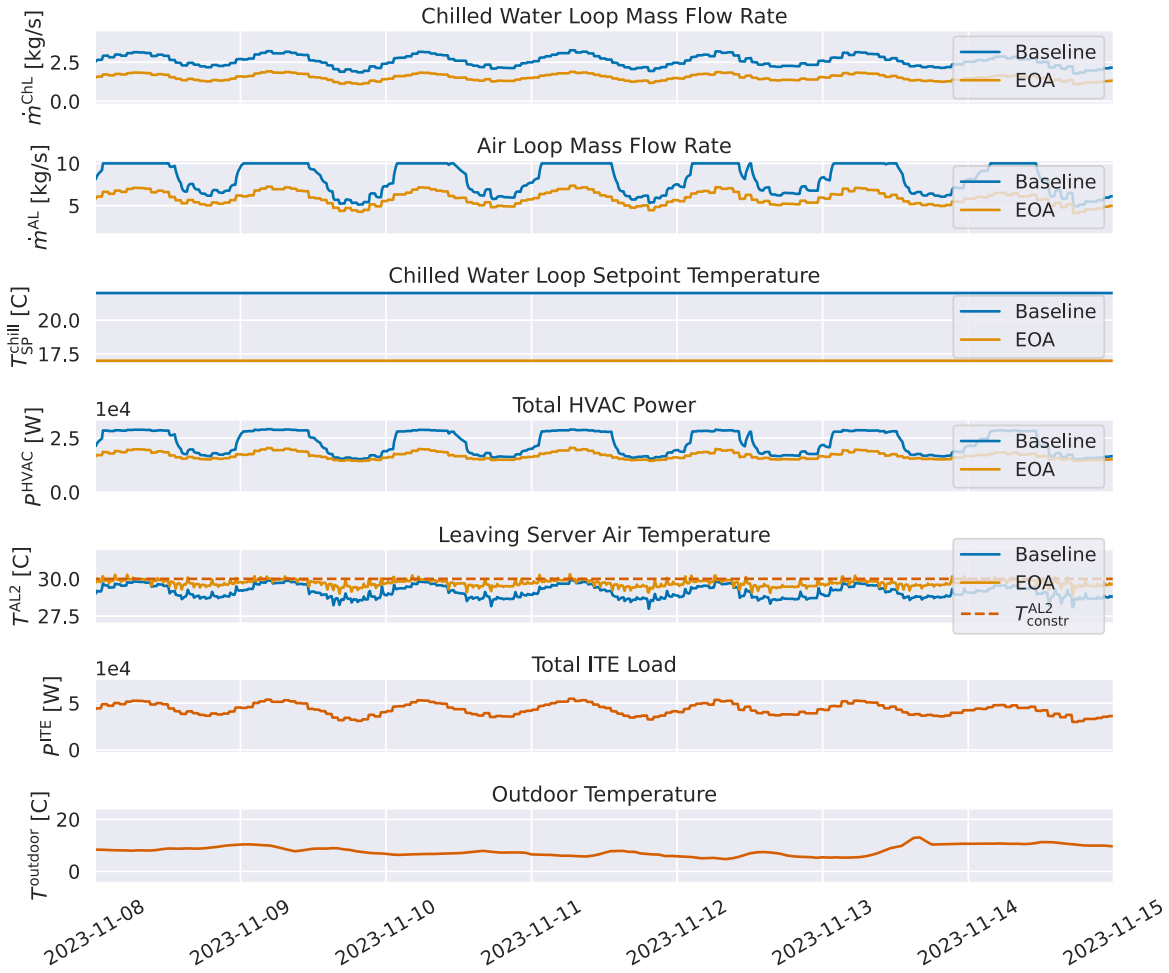


Figure D.2: The behaviour of the EOA over the course of a single week.

D.2 BA

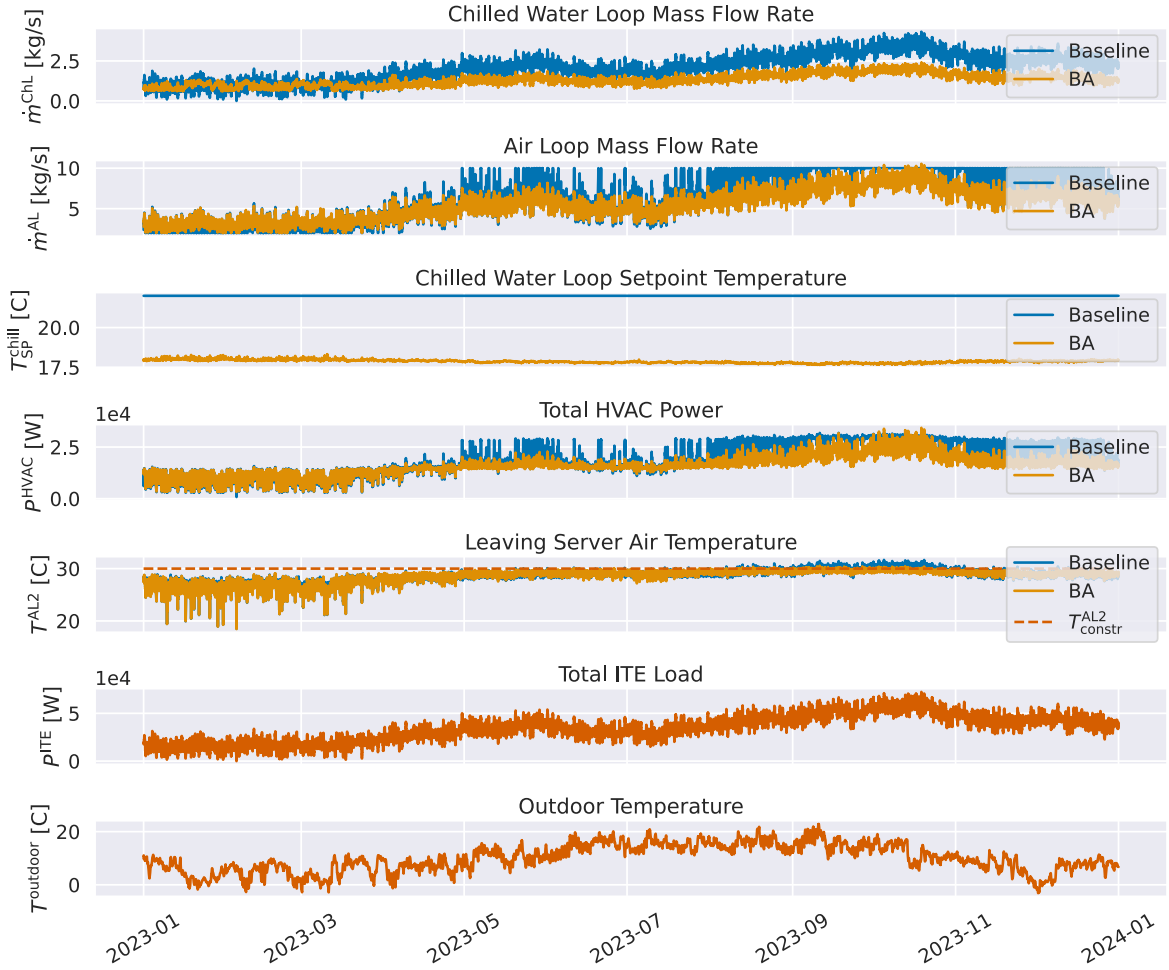


Figure D.3: The behaviour of the BA over the course of a year, compared to the baseline.

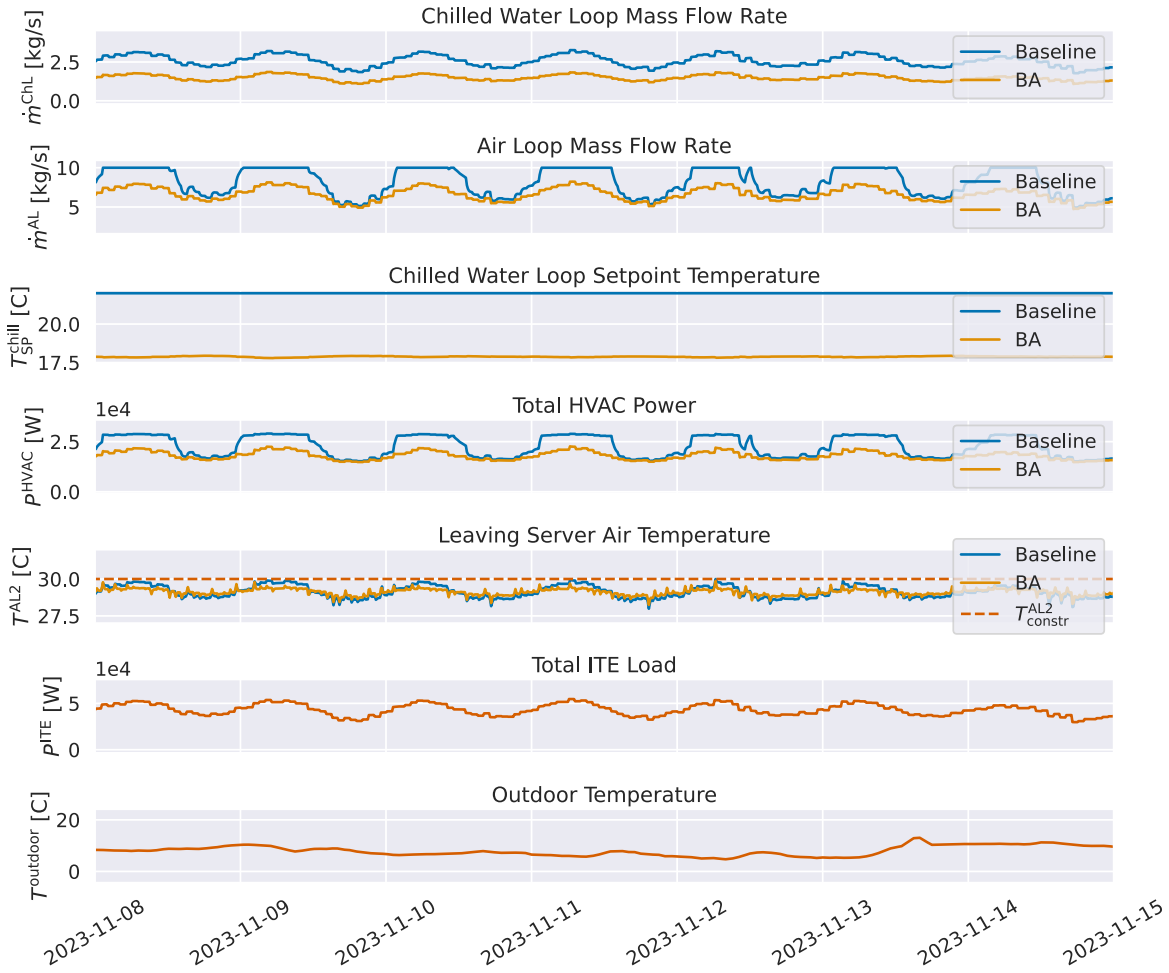


Figure D.4: The behaviour of the BA over the course of a single week.

D.3 COA

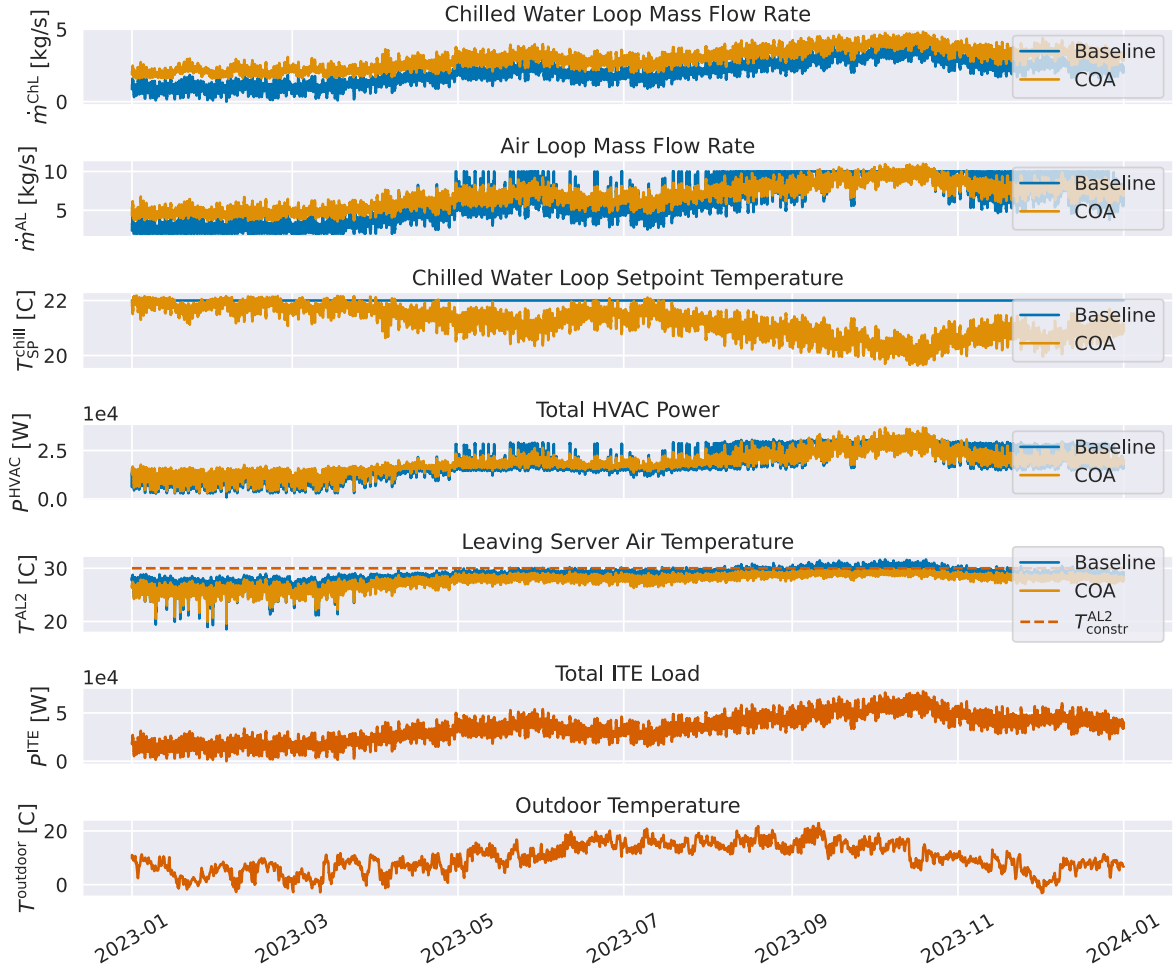


Figure D.5: The behaviour of the COA over the course of a year, compared to the baseline.

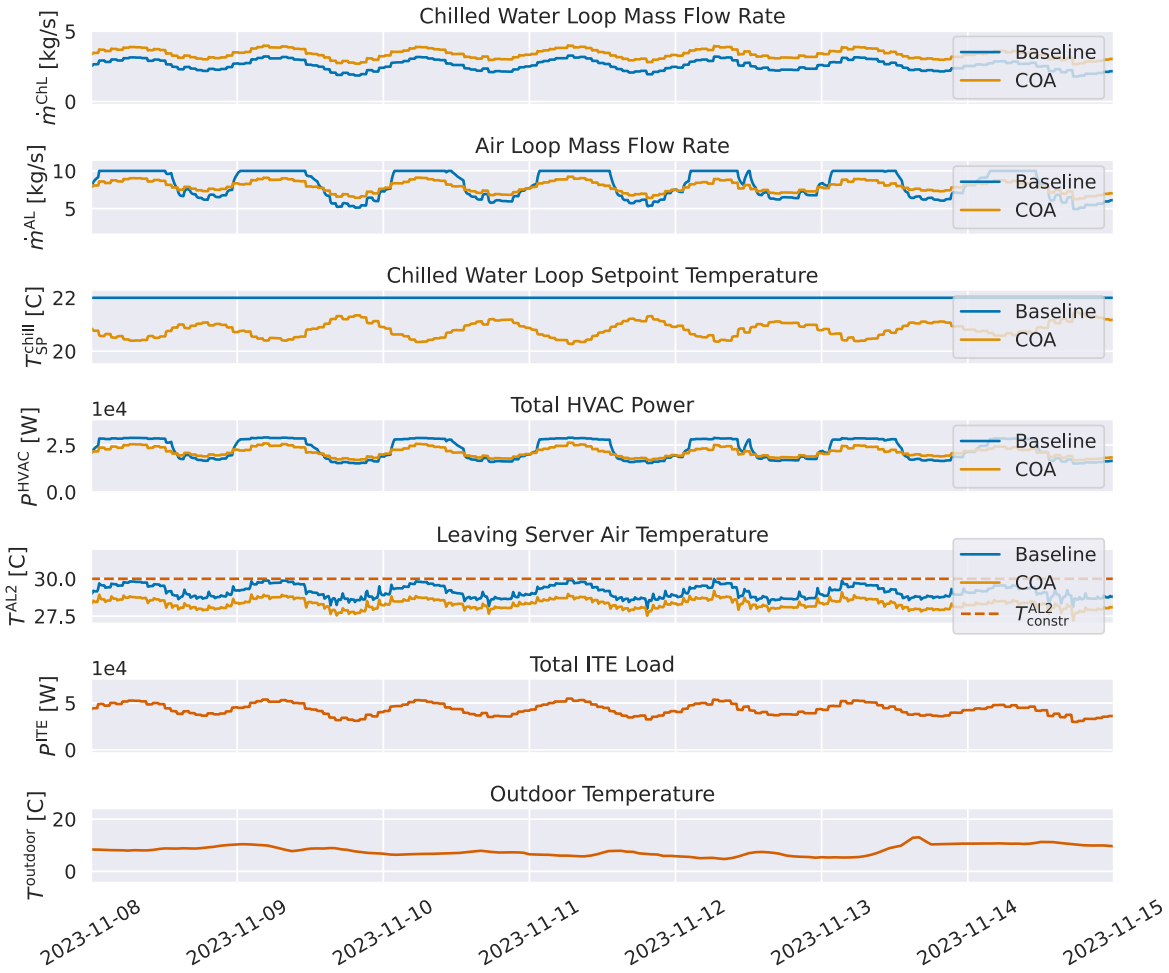


Figure D.6: The behaviour of the COA over the course of a single week.

Appendix E

Statistical analysis controllers

Table E.1: Shapiro-Wilk test results for normality across different agents and metrics.

Agent	Metric	Statistics	p-value	Normal distribution
COA	Mean HVAC Power	0.822	0.027	False
COA	L2 T_constr	0.923	0.385	True
COA	L2_T_SP	0.963	0.825	True
COA	L2_a	0.923	0.384	True
BA	Mean HVAC Power	0.940	0.552	True
BA	L2 T_constr	0.871	0.104	True
BA	L2_T_SP	0.828	0.032	False
BA	L2_a	0.877	0.121	True
EOA	Mean HVAC Power	0.856	0.068	True
EOA	L2 T_constr	0.717	0.001	False
EOA	L2_T_SP	0.633	0.000	False
EOA	L2_a	0.948	0.650	True
Baseline	Mean HVAC Power	1.000	1.000	True
Baseline	L2 T_constr	1.000	1.000	True
Baseline	L2_T_SP	1.000	1.000	True
Baseline	L2_a	1.000	1.000	True

Table E.2: Summary of the Mann-Withney U-test of the controllers on the test year

Metric	Comparison	Mann-Whitney U-value	p-value	Significant difference
$P_{\text{mean}}^{\text{HVAC}}$	COA vs. BA	0.940	0.552	No
	COA vs. EOA	0.856	0.068	No
	COA vs. Baseline	100.000	0.443	No
	BA vs. EOA	-	-	No
	BA vs. Baseline	0.000	0.000	Yes
	EOA vs. Baseline	0.000	0.000	Yes
$L_2\left(T_{\text{constr}}^{AL2}\right)$	COA vs. BA	0.871	0.104	No
	COA vs. EOA	0.717	0.001	Yes
	COA vs. Baseline	0.000	0.000	Yes
	BA vs. EOA	0.878	0.121	No
	BA vs. Baseline	0.000	0.000	Yes
	EOA vs. Baseline	0.000	0.000	No
$L_2\left(T_{\text{SP}}^{\text{AL1}}\right)$	COA vs. BA	0.828	0.032	Yes
	COA vs. EOA	0.633	0.000	Yes
	COA vs. Baseline	0.000	0.000	Yes
	BA vs. EOA	0.877	0.734	No
	BA vs. Baseline	0.000	0.000	Yes
	EOA vs. Baseline	0.000	0.017	Yes
$L2action$	COA vs. BA	0.877	0.121	No
	COA vs. EOA	0.948	0.650	No
	COA vs. Baseline	100.000	0.000	Yes
	BA vs. EOA	0.878	0.121	No
	BA vs. Baseline	90.000	0.001	Yes
	EOA vs. Baseline	90.000	0.001	Yes