# Physics-Informed Reinforcement Learning Feasibility Study for Building Energy Optimization

Antoni Carrillo Sala

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6246
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

# Abstract

Buildings worldwide account for 30% of energy consumption and Heating, Ventilation and Air Conditioning (HVAC) represent roughly 38% of a building's consumption. Therefore, energy savings are crucial for sustainability. The complexity of buildings, with diverse physical domains and large-scale components, presents challenges to achieving energy-efficient operation. Implementing high-performance controls is effective but takes time and requires qualified experts. Reinforcement learning (RL) offers adaptability but demands extensive data, making it difficult to scale to large systems. RL is extensively used in model-free environments, such as video games; however, when it comes to control the problem is a bit more challenging since it has to achieve stability and robustness of the system. This project explores Physics-Informed RL (PIRL) for building energy optimization, focusing on the supervisory control level. Information from physical models is selected to accelerate learning, and the impact of reinforcement learning on a building's cooling system is studied. Key questions include selecting appropriate information from physical models, determining data requirements, and exploiting the building system architecture for the scalability of PIRL. Dynamic models developed in the Modelica language with an open-source building library are used in the thesis. Numerical experiments are then performed to evaluate the scaling potential of PIRL. One goal is to understand and apply software in the loop methods using the PIRL methodology and Carrier automated logic building control software. It will be shown that physics information helps to reduce training time and that it is possible to save energy using PIRL, in comparison with the baseline controller.

# Acknowledgements

"Alone we can do so little; together we can do so much." – Helen Keller.

Firstly, I would like to thank my supervisors, **Bo Bernhardsson** and **Johan Åkesson**, for all their help and feedback during the development of the thesis.

I would also like to thank **Clas Jacobson** for sharing his knowledge of RL and optimization and for his valuable advice on giving oral presentations.

I would also like to thank **Magdalena Atlevi** and **Junyu Zhao** for introducing me to the simulation software and providing technical support about it.

Thanks to **Viktor Linders** for the great Fika conversations.

Finally, thanks to my colleagues, **Arshad Javeed** and **Ewoud Donck** for their collaboration to get through this together.

# Contents

# 1 Introduction

The global need to address climate change has propelled the development of sustainable practices across all sectors. It is estimated that buildings worldwide consume approximately 30% of the total energy generated [International Energy Agency, n.d.], this highlights the critical role of energy-efficient cooling systems in ensuring environmental sustainability. However, optimizing a cooling system to make it efficient is a challenge, characterized by many different interrelated parameters that can be tuned.

Traditional approaches to improving energy efficiency in buildings have often relied on high-performance control systems and architectures, which, while effective, demand significant time and expertise for implementation. Moreover, the dynamic nature of building systems necessitates continuous optimization to adapt to varying environmental conditions and cooling requirements. In recent years, reinforcement learning (RL) has emerged as a promising method for adaptive control, allowing autonomous optimization through continuous interaction with the environment.

RL has demonstrated remarkable success in some environments such as video games. However, its application to real-world control problems, particularly in complex systems like buildings, presents unique challenges. Ensuring stability and robustness in control strategies while effectively leveraging optimal control actions remains a remarkable task. This project seeks to bridge this gap by exploring the application of Physics-Informed Reinforcement Learning (PIRL) for building energy optimization.

The main focus of this thesis is to investigate how RL algorithms with knowledge about fundamental physical principles can be used in cooling systems control applications.

Key questions addressed in this research include the selection of pertinent information from physical models, determination of data requirements for effective learning, and exploitation of building systems architecture to facilitate the scalability of PIRL algorithms. For that, numerical experiments are performed in a simulated environment. The model used for simulation corresponds to a representative data center based in Florida, along with its cooling system.

The first chapter of this thesis aims to describe the cooling system object of this project and analyze its behavior through numerical experiments. After that, relevant assumptions for the rest of the project are presented in Chapter 3. Then the optimization problem that the RL algorithm will solve is formulated in Chapter 6. In Chapters 4 and 5, RL is introduced, briefly explaining how it works and its main concepts, including the environment definition, which essentially consists of the problem representation for RL and a simple experiment using the Q_learning algorithm is conducted. Then the definition and implementation of the different RL models is described, providing a summary table of the models that will be tested, in Chapter 7. Finally, the numerical experiments, including training and testing the RL models, and their results are addressed in Chapter 8.

# 2

# The Cooling System

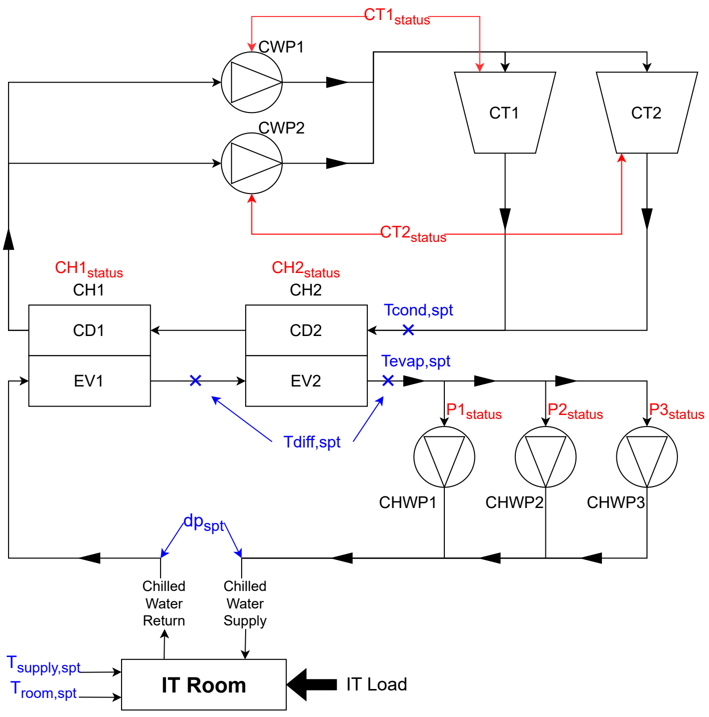
In this chapter, a brief description of the cooling system is given. Furthermore, some experiments are presented to study the influence of each controller parameter on the power consumption and efficiency of the components and the total plant.

The cooling plant consists of a chiller plant, where the chillers produce cold water, that will go to the IT room through pipes. The fans in the IT room are responsible for transferring the heat from the air to the cold water, which will go back to the chiller plant afterward. The cooling towers transfer the heat from the chiller's condensers to the environment. In both, the cooling water and chilled water circuits some pumps are responsible for moving the water through the circuit. This system is artificial which means that the component characteristics and system architecture are representative but do not correspond to any Carrier or other equipment or system.

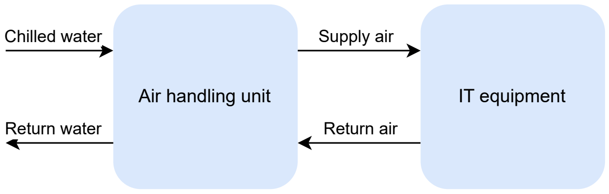
The cooling equipment, including the chillers, cooling towers, chilled water pumps, and cooling water pumps, is located in an isolated room, the chiller plant. Two pipes conduct the chilled water from the plant to the IT room and the return water from the IT room to the plant, respectively.

The cooling load that needs to be handled is produced inside the IT room by the IT equipment. Inside this room, air handling units transfer the heat from the air to the chilled water they receive from the chiller plant. The cooling load is assumed to be equal to the IT power consumption since the heat transfer between the room and the environment is negligible compared with the IT power.

A description of the system, IT room, and chiller plant is provided in Figure 2.1.



**(a)** Chiller plant layout. It includes the decision variables detailed in Table 2.1. **CT** = Cooling tower, **CD** = Condenser, **EV** = Evaporator, **CDWP** = Cooling Water Pump, **CHWP** = Chilled Water Pump, **CH** = Chiller.



**(b)** IT room layout

**Figure 2.1** System layout diagrams

The figure shows that the two chillers, CH1 and CH2, are connected in series instead of the standard parallel approach.

A building management system (BMS) controller controls the plant's dynamic inputs. This controller works in a regulatory level, below the RL controller, as shown in Figure 2.2. This controller aims to track the setpoints for the dynamic variables of the plant. However, those setpoints, which are inputs to the controller, can be optimized to increase the system's efficiency by relocating the power consumption between the different equipment units.

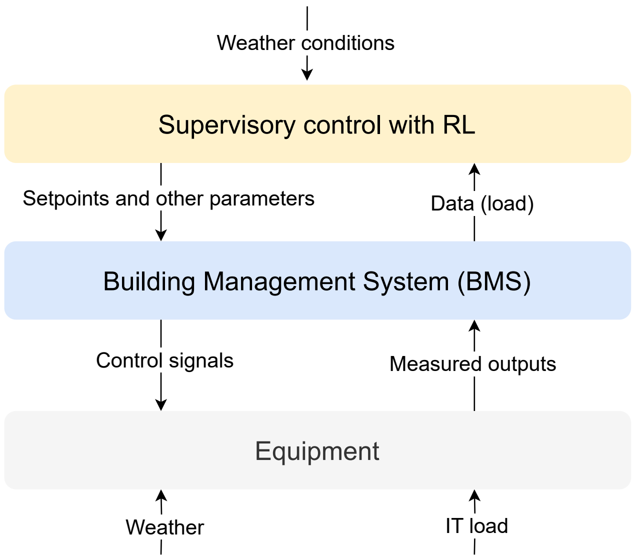

Weather conditions
Supervisory control with RL
Setpoints and other parameters
Data (load)
Building Management System (BMS)
Control signals
Measured outputs
Equipment
Weather
IT load

**Figure 2.2** Layered control architecture. The equipment is controlled by a regulatory control layer (BMS). The setpoints for the controllers on this layer and the equipment status variables are decided by the RL layer.

More concretely, the decision variables are shown in Table 2.1:

| **Variable** | **Description** |
|---|---|
| $CH1_{status}$ | Status (on/off) of chiller 1 |
| $CH2_{status}$ | Status (on/off) of chiller 2 |
| $T_{evap,spt}$ | Chilled water temperature set-point, after evaporator 2 |
| $T_{diff,spt}$ | Chilled water temperature set-point difference between chillers 1 and 2. The setpoint for the temperature of the water leaving the evaporator 1 is computed as $T_{evap,spt} + T_{diff,spt}$ |
| $CT1_{status}$ | Status (on/off) of cooling tower 1 |
| $CT2_{status}$ | Status (on/off) of cooling tower 2 |
| $T_{cond,spt}$ | Cooling tower leaving water (condenser 1 entrance) temperature setpoint value |
| $P1_{status}$ | Status (on/off) of chilled water pump 1 |
| $P2_{status}$ | Status (on/off) of chilled water pump 2 |
| $P3_{status}$ | Status (on/off) of chilled water pump 3 |
| $dp_{spt}$ | Differential pressure on the chilled water circuit, measured on the chiller plant side between the supply and return water |
| $T_{room,spt}$ | Setpoint for the room temperature |
| $T_{supply,spt}$ | Setpoint for the temperature of the supply air in the air handling units |

**Table 2.1** Decision Variables

The variables $T_{evap,spt}$, $T_{diff,spt}$ and $T_{cond,spt}$ are constrained to be in a certain range for model validity and safety reasons. These ranges are the following:

$$6.67°C < T_{evap,spt} < 15°C \tag{2.1}$$

$$0°C < T_{diff,spt} < 15°C - T_{evap,spt} \tag{2.2}$$

$$15°C < T_{cond,spt} < 29.45°C \tag{2.3}$$

Note that the $T_{diff,spt}$ range depends on the evaporator leaving water setpoint. By definition, the evaporator leaving water temperature setpoint of chiller 1 is $T_{evap,spt} + T_{diff,spt}$. Therefore, this constraint ensures that the evaporator leaving water temperature of the upstream chiller is between $6,67°C$ and $15°C$.

## 2.1 Experiments

One cannot be sure that the solution given by a trained RL model is optimal without looking at the energy behavior of the plant. With experiments, one can get an idea of what the optimal solution looks like and compare that with the solution that the RL controller achieves.

The experiments were performed in a simulated model of the real plant, as part of this master thesis work to study the effect of modifying each set point on equipment efficiency and power consumption. Some variables are modified in them, and others are set to their default values, which are the central values of the allowed variation range. All the experiments were performed with fixed outdoor temperature of 21 °C and relative humidity of 0.8, except the ones where the outdoor conditions were explicitly changed. In all the experiments it is checked that the cooling demand is tracked, this is done by checking the temperature in the room and ensuring that it has the steady state value meeting the setpoint (26 $^{\circ}C$). If the cooling demand was not covered in a particular simulation, which involves the temperature in the room being higher than its setpoint in steady-state, that result was disregarded, which means that the cooling load was always tracked in the plotted results.

## Chiller efficiency versus part load ratio and condenser and evaporator temperatures

This experiment aims to explore the chiller efficiency curves. For this experiment, just one chiller and one cooling tower were running. In the first stage of the experiment, the cooling load was set as a varying parameter to explore the effieciency for different Partial Load Ratio values (PLR). This load ratio measures the chiller capacity, being 0 when the chiller is off and 1 when the chiller is at its maximum capacity. The Coefficient of Performance (COP) of the chiller (not including the rest of the equipment) was computed for each value of PLR. The COP of a chiller corresponds to the coefficient between the cooling capacity supplied by it and its power consumption, the values plotted are steady-state values. Two cases were studied:

- Setting the temperature setpoint of the water leaving the evaporator at 10.8 $^{\circ}C$ and computing the COP for three different condenser entry water temperatures (A qualitative plot representing this function is displayed in Figure 2.3). The best efficiency is achieved with a part load ratio of around $50\%$. The efficiency is the same for condenser temperature setpoints of $15^{\circ}C$ and $22^{\circ}C$ with higher PLRs (which means higher loads) because the cooling tower fan runs at maximum speed. Therefore, the setpoints are not met, so the real condenser entry temperature is higher than them, however, this doesn't mean that the cooling demand is not supplied. This situation will change if the outdoor temperature varies, with the saturated condenser temperature decreasing as the outdoor temperature decreases.

- Setting the temperature setpoint of the water entering the condenser at 24 $^{\circ}C$ and computing the COP for three different evaporator-leaving water temperatures (A qualitative plot representing this function is displayed in Figure 2.3).

Again, It can be seen that the best efficiency is achieved with a part load ratio of around 50%.

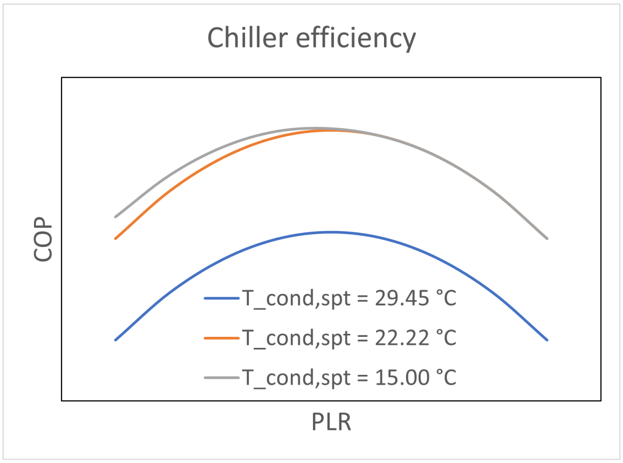


**Figure 2.3** Qualitative plot of the COP from one chiller versus partial load ratio and condenser entry water temperature setpoint. It is seen that the optimal value of PLR is around 50%. The efficiency increases with decreasing condenser temperature. For higher condenser temperature setpoints, the setpoint is not met, so there reducing it more doesn't make a difference.

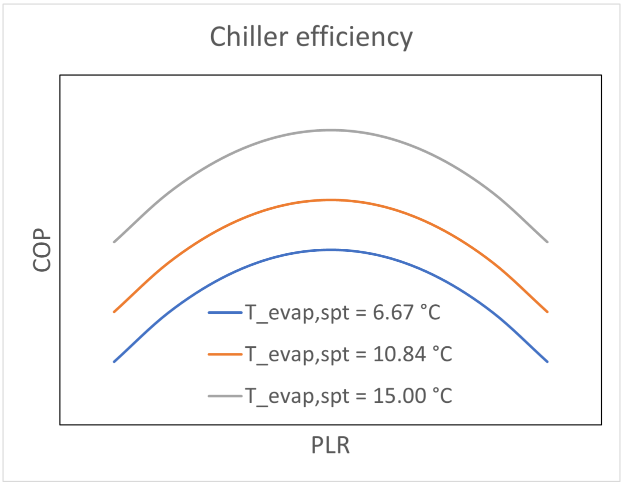


**Figure 2.4** Qualitative plot of the COP from one chiller versus partial load ratio and evaporator leaving water temperature. An optimal value of PLR around 50%. The efficiency increases with increasing evaporator temperature.

In the second stage of the experiment, the load was set constant to 500kW, which corresponds to a PLR of roughly 45%. The effect on the efficiency of $T_{evap,spt}$ and $T_{cond,spt}$ was studied. A qualitative plot representing this function is displayed in Figure 2.5.

The chiller efficiency always increases when the condenser entry water temperature decreases, up to a certain point, at which the cooling tower is saturated, so the real condenser entry temperature becomes no longer controllable and cannot be decreased more. Moreover, the efficiency increases as the evaporator leaving water temperature increases.

Different weather conditions and cooling load would lead to different saturation values for the condenser entry water temperature.

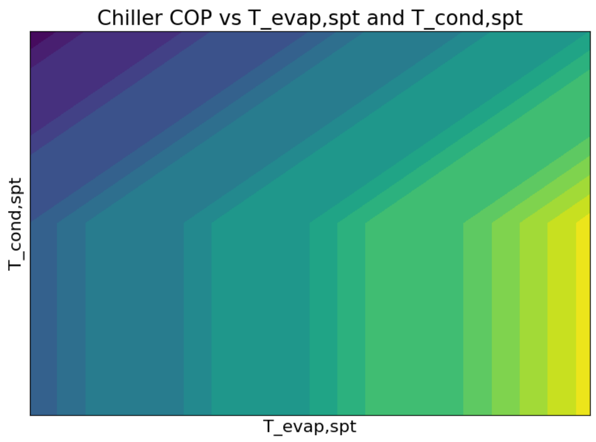


**Figure 2.5** Qualitative plot of the COP from one chiller versus condenser and evaporator temperature setpoints. The results are accordant with the previous experiment: Higher evaporator temperature and lower condenser temperatures contribute to increasing the efficiency. There is saturation in the condenser temperature for the experiment conditions (the setpoint is not met for lower values)

## Power consumption versus chiller staging

In this experiment, the staging of the chillers, turning one of the chillers on and off while keeping the other one on, was explored. The chiller power consumption was studied for different cooling load values. Figure 2.6 shows each case's power consumption on the different equipment elements.

The tested loads are 25 %, 35%, 50%, 75 % and 100 % of the maximum capacity (2000 kW). The power consumption plotted is the steady state power consumption for the experiment conditions. Note that, for obvious reasons, the total power consumption increases as the load increases.

In some cases, the power consumption of the Air Handling Units (AHUs) and chilled water pumps is so small that it is not noticeable on the graph. The experiment conditions are: $T_{evap,spt} = 7.5°C$, $T_{diff,spt} = 2.2°C$, $T_{cond,spt} = 23°C$, outdoor temperature: $T_{outdoor} = 21°C$, Relative Humidity: $RH = 0.8$

It is concluded that with lower loads, it is more efficient to have one chiller on, while with a higher load, turning both of them on is better. Besides, for the two higher cooling loads, the system cannot provide the needed cooling capacity with only one chiller. It requires both of them to run. For the experiment conditions, the threshold to turn the second chiller on seems to be between 700 and 1000 kW. However, this will differ for other set points' values, loads, and weather conditions.

Furthermore, the staging of the chillers seems only to affect the power consumption on the chillers, not really on the other elements, one can appreciate that by looking at the different color sections in the bar plot.

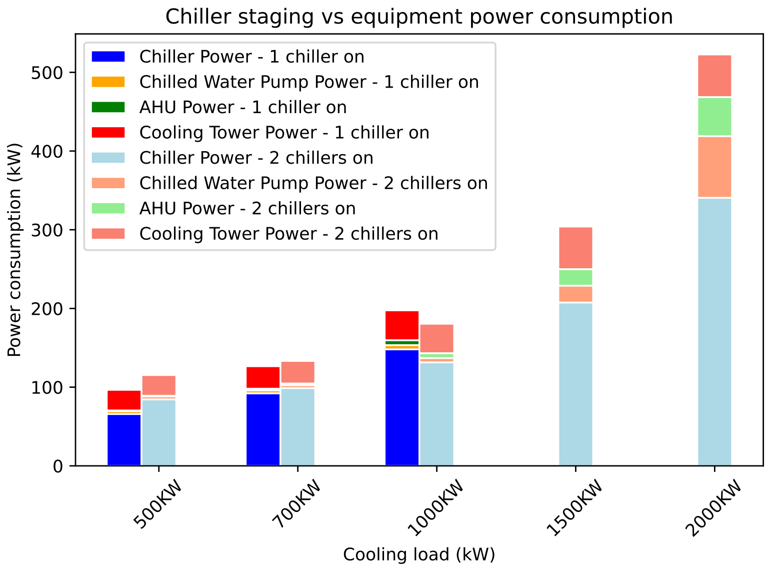


**Figure 2.6** Equipment power consumption versus chiller staging. 1 chiller is better for lower loads. For very high loads, 2 chillers are needed to supply the cooling demand, for the experiment conditions. When 2 chillers are running, the chiller power consumption is the sum of the power consumption at each chiller.

## Equipment power consumption versus cooling tower staging

In this experiment, the staging of the cooling towers, turning one of the cooling towers on and off while keeping the other one on was explored. The status (on/off) of each cooling water pump is the same as its associated cooling tower. Therefore, with this experiment, the pumps are also turned on and off.

The equipment power consumption was studied for different cooling load values. Figure 2.7 shows each case's power consumption on the different equipment units. The experiment conditions are: $T_{evap,spt} = 7.5°C$, $T_{diff,spt} = 2.2°C$, $T_{cond,spt} = 23°C$, $T_{outdoor} = 21°C$, $RH = 0.8$, all chillers were running

It is concluded that with lower loads, it is more efficient to only have one cooling tower and pump on, while with a higher load, turning both of them on is better. The cooling tower staging affects mostly the power consumption on the chillers and cooling towers. In the cases where it is better to have two cooling towers (and their associated pumps) running, turning on the second cooling tower reduces significantly the chiller power consumption. The reason for this is that it decreases the temperature of the water entering the condenser of chiller two, which is probably saturated in that situations and not at its setpoint.

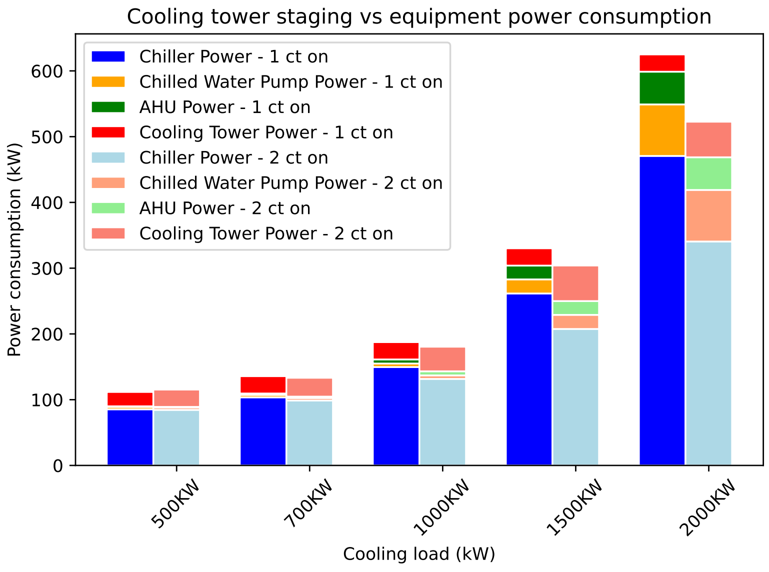


**Figure 2.7** Equipment power consumption versus cooling tower staging. The threshold for turning on the second cooling tower is between 500 and 700 kW for the experiment conditions. For the cases with two cooling towers, the power consumption in the cooling towers is the sum of the power from each tower.

## Power consumption versus equipment staging

This experiment explores the simultaneous staging of the chillers (CH) and cooling towers (CT) for different cooling load values and outdoor temperature values. It can be seen that the best case for lower loads is to have one chiller and one cooling tower running, whereas, for bigger loads, it is better to have two chillers and one cooling tower on. The experiment conditions are: $T_{evap} = 7.5°C$, $T_{diff} = 2.2°C$, $T_{cond} = 23°C$, $T_{outdoor} = 21°C$, $RH = 0.8$

The results are consistent with the previous experiments about staging. Figures 2.8, 2.9, 2.10 show the experiment for different outdoor temperatures. It can be appreciated that the outdoor temperature modifies the optimal load threshold for turning on the second cooling tower. Besides, the optimal threshold to turn on the chiller remains between 700 and 1000 kW.

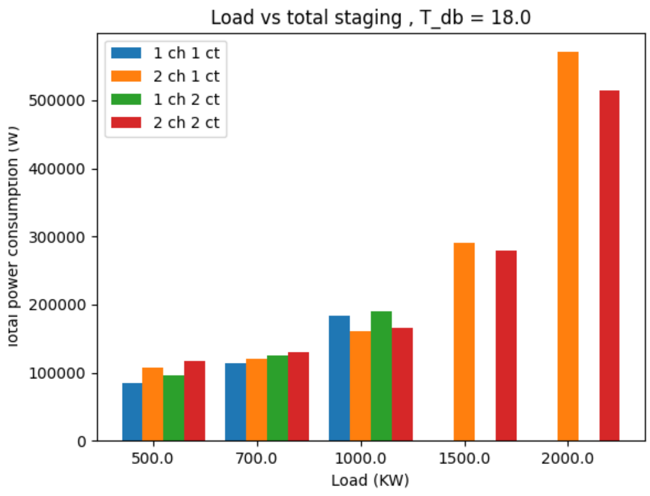


**Figure 2.8** Total power consumption versus equipment staging, $T_{outdoor} = 18°C$. It can be seen that it's better to have 1 chiller for the 2 lowest load values and 2 for the other cases. Turning on the second cooling tower becomes better for the 2 higher loads.

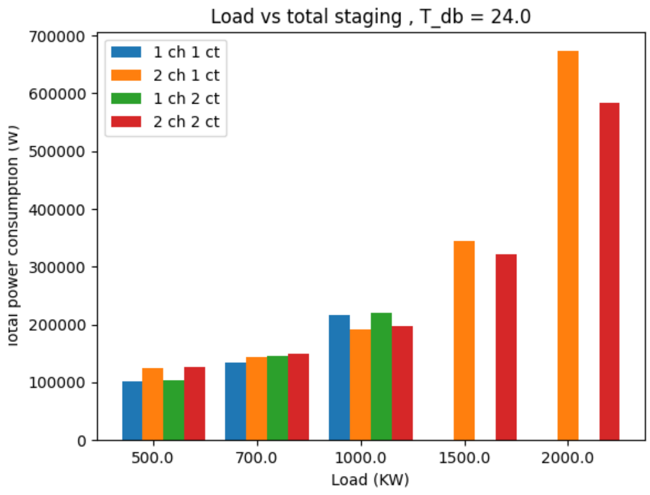


**Figure 2.9** Total power consumption versus equipment staging, $T_{outdoor} = 24°C$. Again, it can be seen that it's better to have 1 chiller for the 2 lowest load values and 2 for the other cases. Turning on the second cooling tower becomes better for the 2 higher loads.

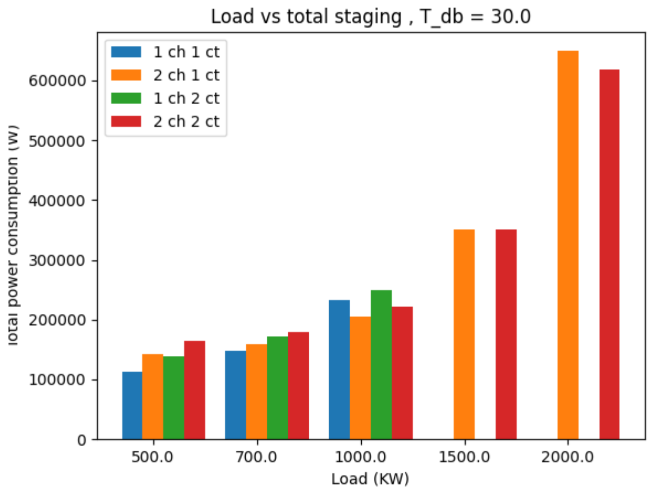


**Figure 2.10** Total power consumption versus equipment staging, $T_{outdoor} = 30°C$. It can be seen that it's better to have 1 chiller for the 2 lowest load values and 2 for the other cases. Turning on the second cooling tower becomes better only for the highest loads.

## Power consumption versus chilled water temperature setpoint

In this experiment, the power consumption of the different elements is explored when varying the leaving chilled water set point ($T_{evap}$). The experiment conditions are: $T_{diff} = 2.2°C$, $T_{cond} = 23°C$, $T_{outdoor} = 21°C$, $RH = 0.8$

The results are shown in Figure 2.11. It is appreciated that, as one increases $T_{evap}$, the power consumption on the chillers mainly decreases, while the power consumption on the AHUs and chilled water pump increases. It is seen that the power of the pumps increases following an approximately quadratic relation with $T_{evap,spt}$, which results in a high total power consumption for high values of $T_{evap,spt}$. Moreover, the chilled water flow saturates at some point (for setpoint values over $9°C$ for the experiment conditions), which involves that the chilled water temperature setpoint $T_{evap,spt}$ is not being met, the real value of the evaporator leaving water temperature is kept lower to meet the cooling demand.

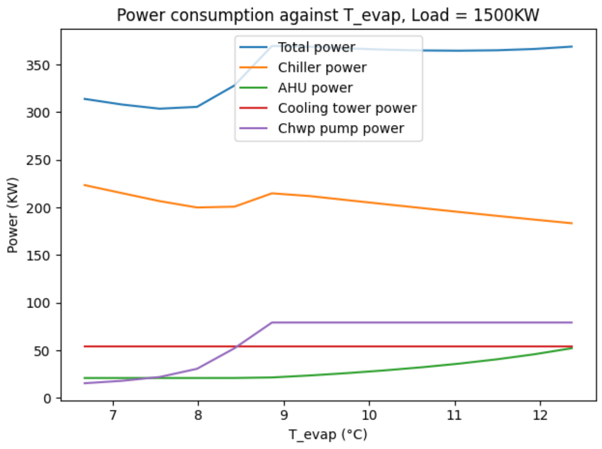
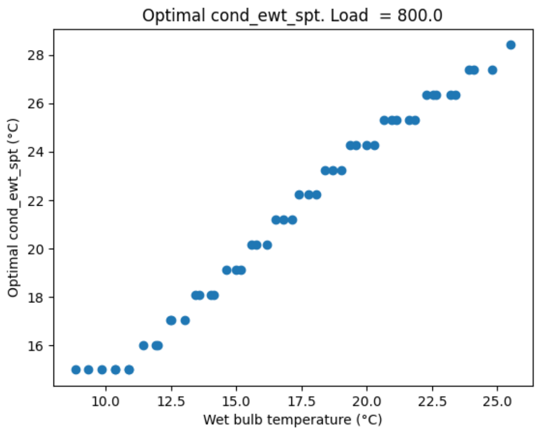


**Figure 2.11** Equipment power consumption versus chilled water temperature setpoint. It can be observed an increasing influence of the chilled water pump power for higher setpoint values. The chilled water flow saturates at some point, involving the real chilled water temperature being colder than its setpoint.

## Best condenser entry temperature

This experiment aims to determine the best condenser entry temperature ($T_{cond,spt}$) for different weather and load conditions. Results are shown in Figure 2.12. The experiment conditions are: $T_{evap} = 7.5°C$, $T_{diff} = 3.2°C$, all the equipment was kept on. The best temperatures are found by grid search, selecting the one that achieves the minimal power consumption from a list.



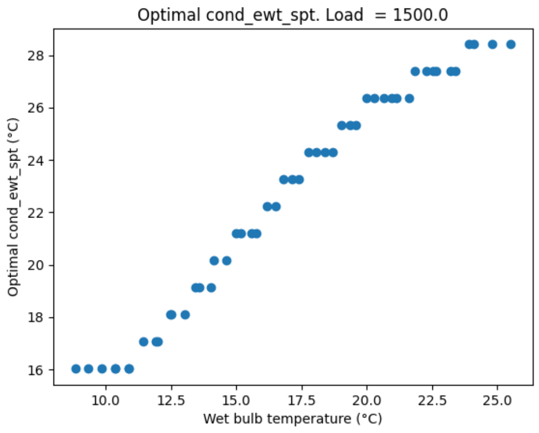


**Figure 2.12** Best $T_{cond}$. For each value of $T_w$, the $T_{cond,spt}$ that gives the smaller power consumption is found through a grid search and plotted. The experiment is repeated for several values of cooling load. It is observed that the best setpoint value follows an approximately piecewise linear relation with $T_w$, slightly influenced by the cooling load.

It can be seen that the relationship can be approximated, with not much error, with a linear policy that computes the optimal $T_{cond,spt}$ based on the wet-bulb temperature and then constrains it to the allowed range. However, the cooling load also has a bit of influence.

## Power consumption versus both chiller evaporator set-points

In this experiment, the differential and evaporator chilled water temperature set-points are varied for different values of the cooling load, and the plant efficiency (calculated as the cooling load over the total power consumption) is represented on a level plot. Figure 2.13 shows the results. A red cross is displayed at the point with the minimum power consumption, this was computed by simply selecting the coordinates of the point with the best efficiency, from the discrete set of set-point values. The experiment conditions were: $T_{cond,spt}$ = 26 $^\circ C$, all the equipment running.

Note that the points where $T_{evap} + T_{diff} > 15$ have been omitted since this would violate Constraint 2.3. The limit region where the constraint is valid appears to not be completely linear in the plot. This is due to the discretization of the setpoints that were tested. Moreover, the simulation fails when the condenser entry water temperature and evaporator leaving water temperature are too close (Small outdoor temperature and large $T_{evap,spt}$), and the load is high. The reason for this is that the cooling capacity is lowered for those conditions to the point of not being capable of handling the cooling load.

It can be observed that the optimal values for this pair of variables in this situation are not very dependent on the weather. Instead, they are highly dependent on the cooling load.

There is a region, approximately for $T_{evap,spt}$ higher than 10 $^\circ C$, but dependent on the conditions, where the efficiency drops drastically. This is related to an increase in the chilled water pump power, which, as seen in Figure 2.11 increases quadratically with $T_{evap,spt}$ (which is directly related to the flow rate).

Moreover, there is mostly one unique global maximum, except for the case of higher loads, when a local maximum appears, and the outdoor temperature begins having some influence. This could be problematic if a gradient method were used to optimize those set points.

Note that the plots are only valid under the experiment conditions, for example regarding staging or $T_{cond,spt}$, if those conditions were fixed, the equipment could just be operated in the maximum efficiency point from these plots. However, this is not the case, so some sort of optimization, including all the parameters that can influence the power consumption and external conditions is needed to find the best operating points.

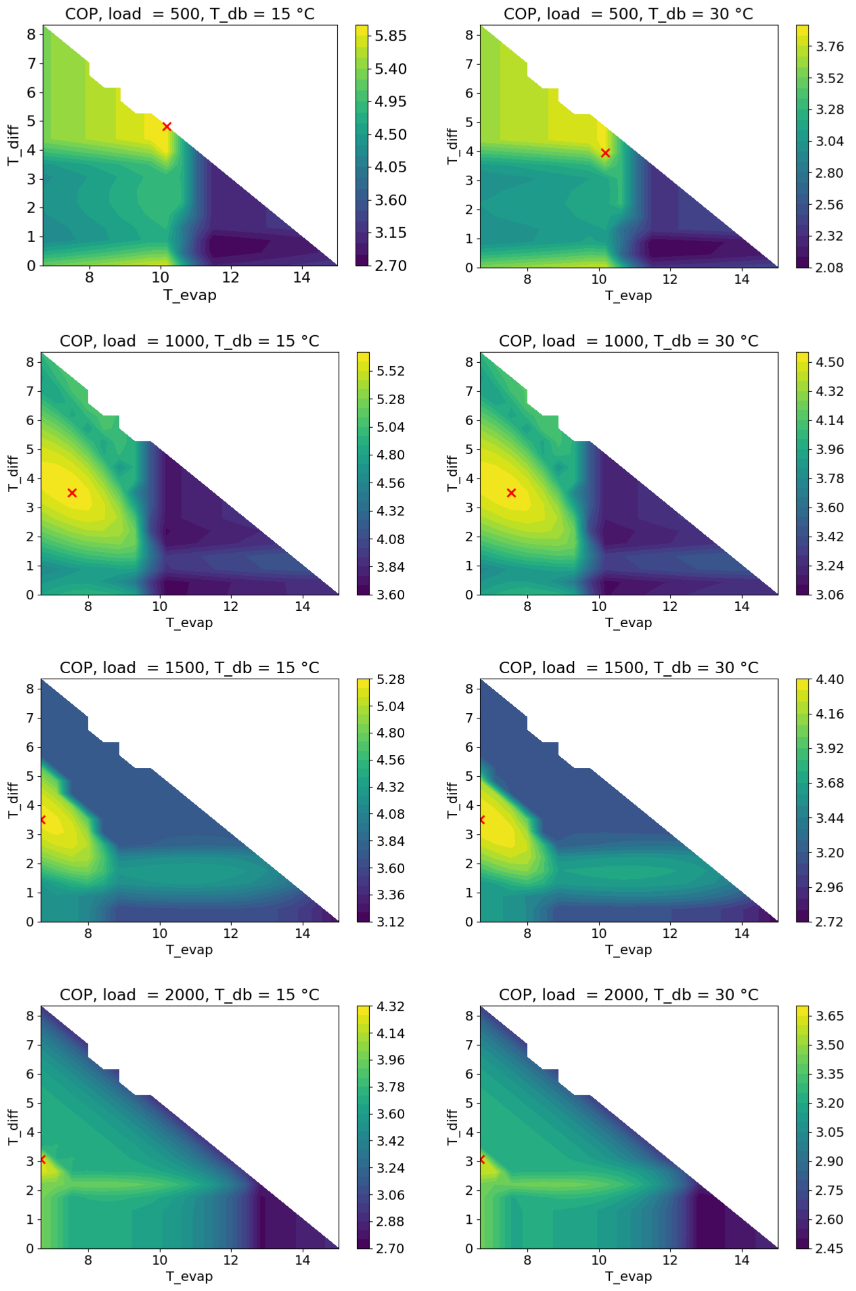


**Figure 2.13** Power consumption versus chilled water temperature setpoint and differential temperature setpoint for different load values. Points where $T_{evap} + T_{diff} > 15$ have been omitted since this would violate Constraint 2.3

## Optimal chiller set-points

This experiment aims to determine the optimal $T_{evap}$ and $T_{diff}$ for different weather and load conditions. Results are shown in Figure 2.14. The upper figure corresponds to fixed weather conditions and variable load, while the lower corresponds to fixed load and variable weather. The best setpoints are found through a grid search by iterating over a list of values for both. The experiment conditions were: $T_{cond,spt}$ = 26 $°C$, all the equipment running. The $T_{diff,law}$ is the temperature that distributes the cooling load evenly between both chillers. It is computed as:

$$T_{diff,law} = \frac{T_{return} - T_{evap}}{2} \tag{2.4}$$

Where $T_{return}$ is the temperature of the return water to the chiller plant, both chillers and cooling towers were kept on during the experiment.

It can be seen that the best $T_{evap,spt}$ follows a piecewise linear relationship with the cooling load. Both best setpoints are affected mainly by the cooling load rather than the weather conditions. The best $T_{diff}$ is close to the one computed using the above formula for loads higher than 800 kW. For lower loads, having a large $T_{diff}$ means turning off chiller 1 since it would have a set point equal to or higher than the return water temperature. This can be observed in the figures for load = 500 kW on experiment 2.1 where increasing $T_{diff}$ over a specific limit does not affect the power consumption. The simulation considers that chiller one consumes no power under that situation, even if it is kept on. However, in the real plant, it would be better to turn the chiller off. In the range where two chillers are better than one, following the law for evenly distributed load gives roughly an optimal setpoint for $T_{diff}$.

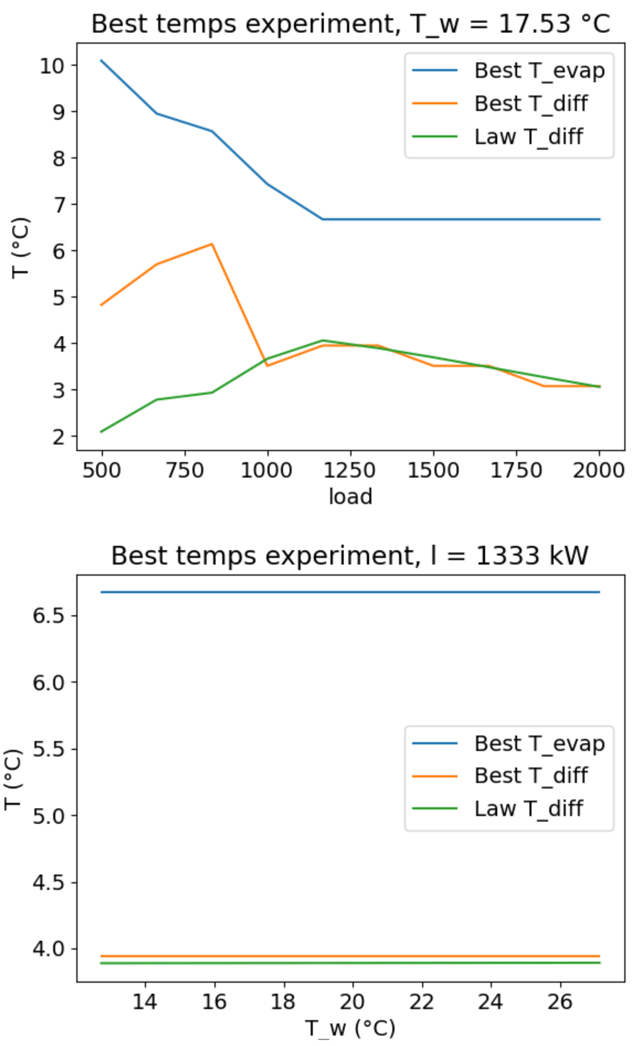


**Figure 2.14** Optimal set-points for varying load (upper figure) and outdoor wet-bulb temperature (lower figure). It can be noted that the best $T_{evap,spt}$ follows a piecewise linear relationship with the cooling load. the best setpoints are affected mainly by the cooling load. The law for $T_{diff,spt}$ seems to give a good approximation of the best setpoint value for loads over 800 kW.

## Efficiency versus evaporator and condenser temperatures

In this experiment, the efficiency of the plant was studied for different values of $T_{evap,spt}$ and $T_{cond,spt}$, creating efficiency level curves for various loads and weather conditions. Figure 2.15 illustrates some of these experiments. The experiment conditions were: RH = 0.8, all equipment running.

The results of this experiment coincide with the previous one in showing that the optimal $T_{evap,spt}$ is not very dependent on the weather but more on the cooling load. However, the optimal $T_{cond}$ depends mostly on the weather, as Figure 2.12 shows.

For lower $T_{cond}$ than a specific value, lowering it more does not affect the efficiency. This is because the cooling towers are saturated, and the set point is not being met, so the real condenser entry water temperature is higher than that.

As before, for high loads it is not possible to supply the demand with low condenser temperatures, this is only appreciable with low outdoor temperatures, since only in this situation the condenser entry temperature is equal to its setpoint (otherwise it could be higher than the setpoint and therefore not cause any problem). However, in this situation, the optimal condenser temperature is very close to the limit value where the cooling demand cannot be met. This last result makes it particularly difficult for an RL controller to learn the optimal law for $T_{cond,spt}$ in a safe way since exploring the area close to the optimal setpoints involves not meeting the cooling demand under certain conditions.

Moreover, there is a local maximum apart from the global, this could be problematic if a gradient method were used to optimize those setpoints.

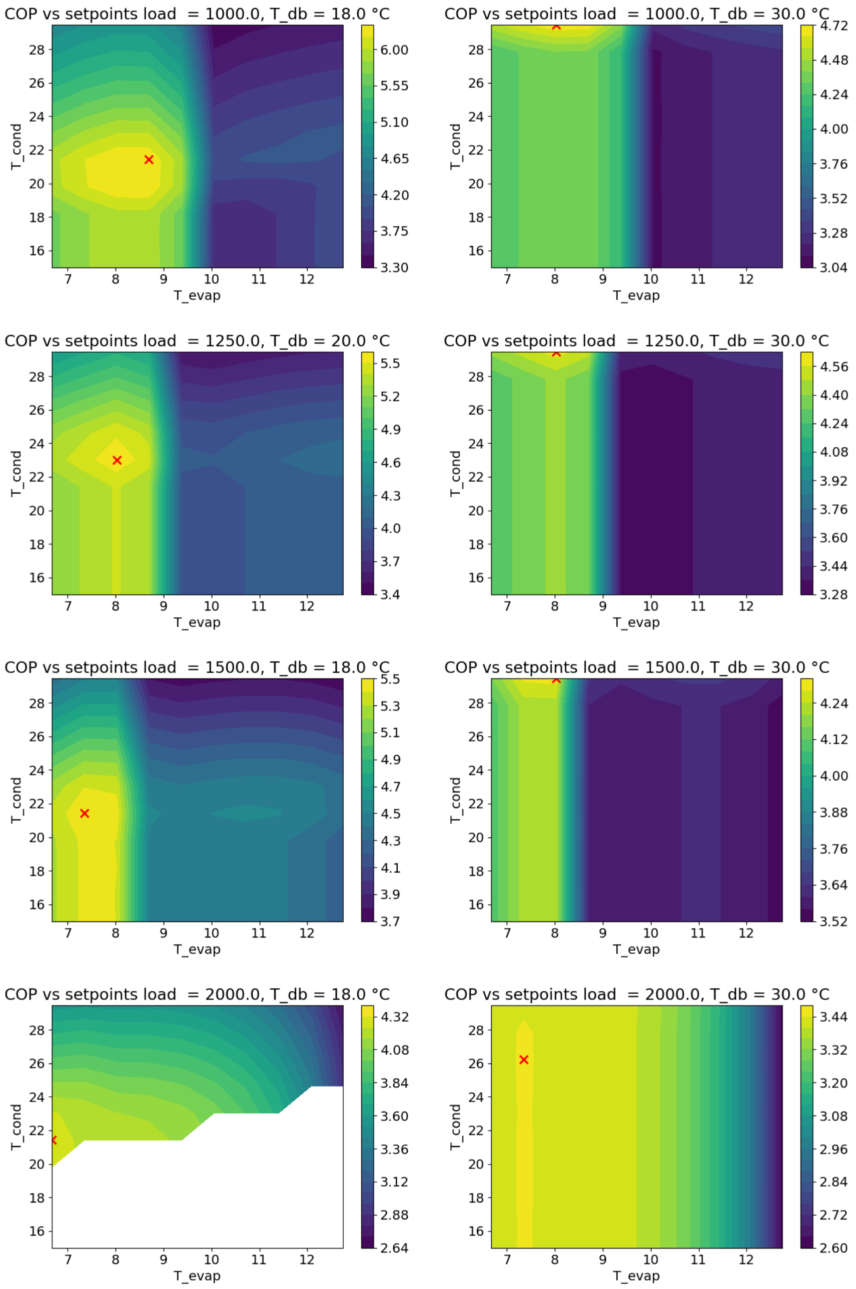


**Figure 2.15** Efficiency versus $T_{evap,spt}$ and $T_{cond,spt}$ for different weather and load conditions. The best $T_{evap,spt}$ is mostly influenced by the cooling load, while the best $T_{cond,spt}$ is mostly influenced by the weather. Experiment conditions: RH = 0.8, all equipment running.

### Power consumption versus differential pressure on the chilled water circuit

In this experiment, only the differential pressure of the chilled water is varied. The results are shown in Figure 2.16. It can be seen that increasing the differential pressure setpoint increases the power consumption in the chilled water pumps but decreases the power consumed in the AHUs slightly. The total power is increased with this setpoint, and it seems optimal to keep it low.

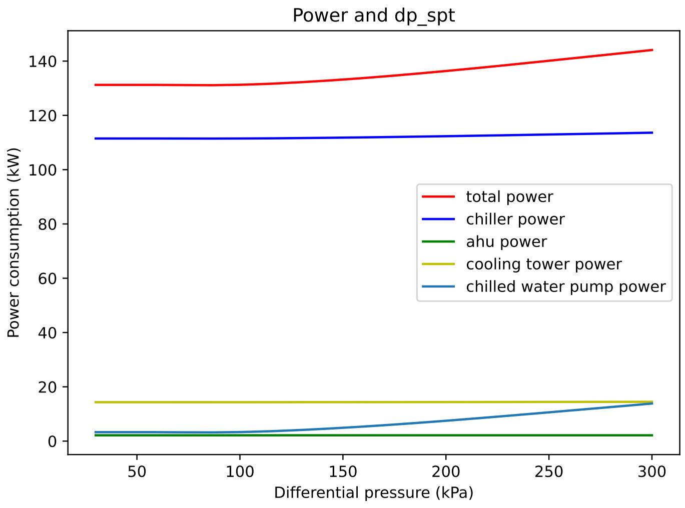


**Figure 2.16** Power consumption versus differential pressure on the chilled water circuit. It is always better to keep this parameter low.

### Power consumption versus chilled water pump staging

This experiment analyses the chilled water pump staging while keeping on the two chillers and the two cooling towers. The experiment conditions are: $T_{evap} = 6.7°C$, $T_{cond} = 24°C$, $T_{diff} = 2°C$, $T_{outdoor} = 30°C$, $RH = 0.8$.

Figure 2.17 shows the total power consumption for different numbers of chilled water pumps running. The experiment shows that having the three pumps running is always better. This seems intuitive considering that the power consumption of each pump depends quadratically on its flow rate, the total power consumption will be lower if the total flow is split along the three pumps.

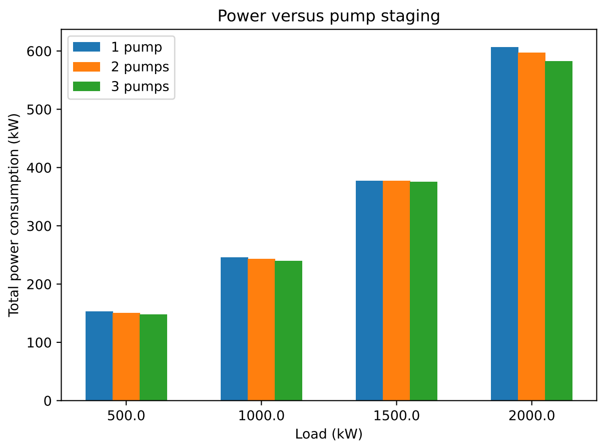


**Figure 2.17** Power consumption versus chilled water pump staging. It is always better to have all of the pumps running for the experiment conditions.

## 2.2 Insights from the experiments

From the previous experiments and physical intuition, the following conclusions can be made:

- The set-point variables that affect the power consumption at the chillers are $T_{evap,spt}$, $T_{diff,spt}$, and $T_{cond,spt}$.
- Increasing the chilled water setpoint ($T_{evap,spt}$) or decreasing the temperature setpoint at condenser two ($T_{cond,spt}$) reduces the chiller power consumption, although it can have the opposite effect on the total power consumption. This can be concluded from Figure REF.
- Increasing $T_{evap,spt}$ increases the power consumption in the AHUs, and chilled water pumps. This is due to the fans in the AHUs needing more power to extract the same amount of heat from the air if the cooling water has a higher temperature. There is also a need for a higher chilled water flow rate to extract the same amount of heat if the setpoint is increased, which increases the power consumption in the chilled water pumps. This can be concluded from figure 2.11
- Decreasing $T_{cond,spt}$ increases the power consumption in the cooling towers. This is because more water cooling in the cooling tower implies a higher consumption of the cooling tower fan. Therefore, it is necessary to find an optimal value where the sum of the power consumption of each component is minimized.

- The chilled water differential pressure balances the power consumption in the chilled water pumps and the AHU fan power. The intuition behind this is that more differential pressure in the system is related to a higher hydraulic pressure drop in the circuit, which relates to higher water flow. If the water flow increases, less fan power is needed to transfer the heat from the air to the water in the AHUs. The experiment shown in Figure 2.16 shows this behavior, showing that the optimal is to keep the differential pressure low. However, if it is too low, there is a risk of not having enough pressure to overcome the losses on the circuit. Therefore, a minimum value of 118 kPa will be considered for this variable.

- The optimal condenser temperature depends piecewise linearly on the wet bulb's outdoor temperature and it depends weakly on the cooling load as well, as seen in Figure 2.12.

- This is a multi-dimensional problem. The figures only explore one setpoint or a pair of setpoints at a time, but the shape of the plots can be highly dependent on the rest of the setpoints and conditions. Therefore, automated optimization methods or data-driven approaches are needed to find a global solution to the problem.

# 3

# Assumptions

This chapter summarizes the assumptions made for the rest of this work.

## 3.1 Steady state assumption

The goal is to minimize the power consumption of the chiller plant. When a setpoint is modified or the external conditions (load and weather) change, the plant goes through a transient and then reaches an equilibrium point with a new steady-state value of power consumption.

The set points need to be modified when the external conditions change. Therefore, if the transient is short in comparison to the rate at which these conditions change, it would be much more beneficial to reach an optimal value of steady-state power consumption than to improve the transitory behavior.

In this work, it will be assumed that the weather and cooling load conditions change at a much slower rate than the settling time of the system.

This assumption is reasonable in the case of the weather since it changes slowly. However, it is not very reasonable to assume this for the cooling load, since it can change quickly in punctual moments. However, the assumption is needed for training the algorithms, then, they can be deployed to the real plant and be used to predict the best action based on the external conditions, even if they change before the system has reached an equilibrium.

## 3.2 Stateless assumption

In this section, the problem is classified according to Powell's Article [Powell, 2019] as a state-independent terminal reward problem.

From the problem's point of view, every simulation until steady state corresponds to one time-step in the sequential decision problem, as long as one considers each transition of the Markov Decision Process (MDP) to correspond to a simulation until steady state.

In other words, for a set of decisions and external conditions (modeled as exogenous information), the plant behavior is simulated until it reaches steady state. Then, the power consumption is read from the simulation. Let's define input as the set of decision variables and external conditions. Similarly, outputs can be defined as the set of power consumption in the different elements and state variables. The work is based on the assumption that for each set of inputs, there is only one possible output under steady-state conditions.

Taking this into consideration, it is easy to see that if the system is on a state $S_t$, the next state $S_{t+1}$ will not depend on $S_t$, it will only depend on the inputs to the simulation. Therefore, it is assumed that the problem is state-independent.

This reduces the problem to minimizing the power consumption for given a set of external conditions at each time-step. Therefore, relating this to Powell's framework, the focus is set on the terminal reward of each time step.

From all of this it is concluded that the problem corresponds to class 1 in Powell's classification, it is a classic stochastic search problem. The problem consists of finding the best policy that inputs the optimal decisions to the system.

All in all, the system can be seen as a static mapping from weather, load conditions, and decision variables to the Steady State power consumption, as seen in Figure 3.1

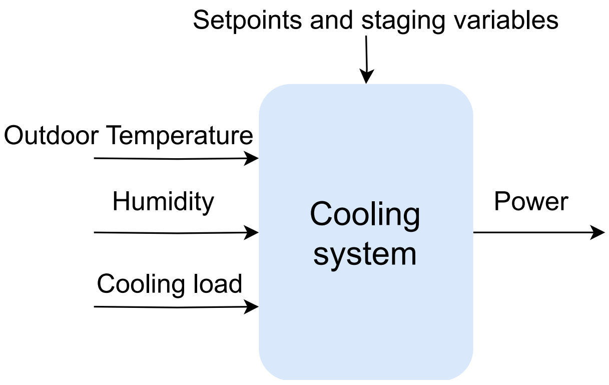


**Figure 3.1** Cooling system. Under the stateless and steady-state assumptions it can be seen as a static mapping from weather and load conditions, setpoints and staging variables to power consumption.

## 3.3 Plant Variation.

The simulation model is not perfect and the real plant changes over time, due to equipment aging. Although the goal of having RL in the controller is to adapt to those variations, some conditions need to be fulfilled for the controller to work as intended.

1. The real plant changes at a slower rate than the RL algorithm is capable of learning. So the algorithm converges quicker than significant plant variations occur.

2. The two chillers deteriorate at the same rate. So both chillers need to have sufficiently similar efficiency curves, even if it's not their original curve. This can be achieved by periodically alternating which is the chiller that can be off during certain periods. This assumption is only needed for models that use a physical law to compute $T_{diff}$, as described in Section 7.5. Whether this is true on the real plant and what the error associated with this assumption is has not been further explored in this work.

# 4 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning method where an agent learns to make sequential decisions by interacting with an environment to maximize cumulative rewards. Unlike traditional supervised learning, where explicit input-output pairs are provided, or unsupervised learning, where patterns are inferred from unlabeled data, RL operates through interaction with the environment. At each step, the agent observes the current state of the environment, selects an action, receives feedback in the form of a reward, and updates its policy to improve future decision-making. The working principle is summarized in Figure 4.1. The goal is to maximize the cumulative reward over several time steps, so the method is particularly suitable for sequential decision-making problems such as the one targeted in this work.

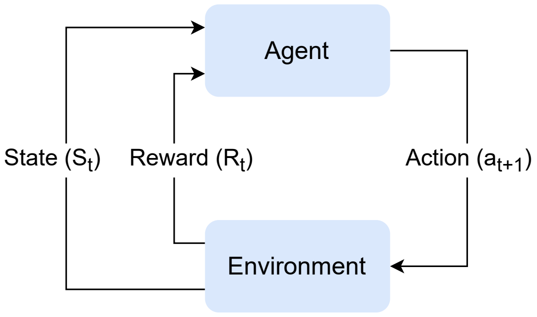


**Figure 4.1** Reinforcement learning concept. The agent (controller) learns by interacting with the environment and observing the results of each action, in terms of next state and reward.

The main characteristic of reinforcement learning is that the agent does not need any internal model of the environment, it just needs to interact with it. For this reason, RL has been extensively used to play video games, achieving good results, an example of this is the Alphazero project [Zhang and Yu, 2020], which developed a model capable of playing chess.

However, in this work, a control problem is attained, rather than a video game. This implies numerous challenges for the reinforcement learning methodology. Concretely, the following points need to be taken into consideration:

- The controlled system must be stable and robust.
- The control signal and the output must satisfy some constraints.

Another common issue in RL is the trade-off between exploration and exploitation, which is crucial for effective learning. Exploitation involves the agent using its current knowledge to maximize immediate rewards, while exploration means trying new actions to gather more information about the environment. Excessive exploitation can lead to suboptimal performance if the agent's knowledge is incomplete, whereas too much exploration can waste resources and delay learning.

There are many different algorithms for reinforcement learning. The Reference [AlMahamid and Grolinger, 2021] suggests a classification based on the state and action spaces topology. Unlimited states refer to those problems where the state space is continuous and therefore the variables can have many values, while limited states refer to problems with a discrete state space. Figure 4.2 shows a classification of the different algorithms.

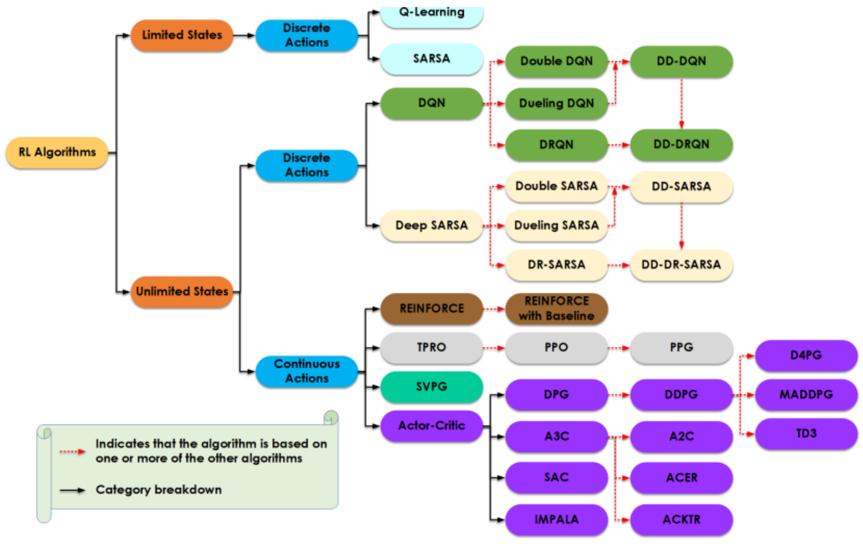


**Figure 4.2** RL algorithms. Source [AlMahamid and Grolinger, 2021]

## 4.1 Fundamentals

This section aims to describe some of the mathematical fundamentals of RL.

## Markov Decision Processes (MDPs)

At the core of RL lies the Markov Decision Process (MDP) framework [Metropolis et al., n.d.] An MDP is a tuple $(S, A, P, R, \gamma)$, where:

- $S$ is the state space.
- $A$ is the action space.
- $P$ is the transition probability function.
- $R$ is the reward function.
- $\gamma$ (gamma) is the discount factor.

An MDP can be represented with a graph where the nodes represent the states and the arrows represent the possible transitions. For RL, each transition or subset of transitions (for stochastic processes) is typically associated with an action. Each transition has a certain reward value. A graphical representation of a generic MDP is shown in Figure 4.3.

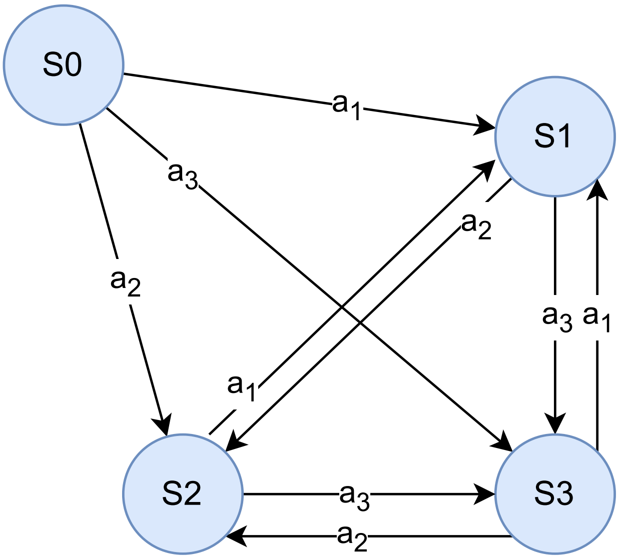


**Figure 4.3** Generic deterministic MDP, with 4 states and 3 possible actions. In an RL problem, each state transition (each of the arrows) would have an associated reward.

## Value Functions

Value functions estimate the goodness of being in a particular state and taking a particular action in a state. There are two fundamental value functions:

- **Action Value Function (Q(s, a))**: It represents the expected return starting from a particular state (s), taking a particular action (a).

$$Q(s,a) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a \right]$$

- **State Value Function (V(s))**: It represents the expected return starting from a particular state under a given policy.

$$V(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right]$$

Let's assume the policy is to choose the action that maximizes Q:

$$\pi(s) = \arg\max_a Q(s,a) \tag{4.1}$$

Then V would be the value of Q for the action chosen that maximizes Q:

$$V(s) = \max_a Q(s,a) \tag{4.2}$$

This is the approach used in algorithms like DQN and Q-learning.

## Bellman Equation [Bellman, 1966]

The Bellman equations are recursive relationships that express the value of a state or action in terms of the values of subsequent states or actions. They are fundamental to understanding the dynamics of RL algorithms. They are based on the principle of optimality, which states that an optimal policy has the property that, whatever the initial state and decisions are, the remaining decisions must constitute an optimal policy, regarding the current state before each decision. In other words, the Bellman equations are built upon the assumption that future actions will continue to be optimal.

The action-value function $Q(s,a)$ represents an estimate of the future discounted reward expected from being in state $s$ and taking action $a$. It can be expressed as

$$Q(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s') Q(s',a')] \tag{4.3}$$

Where $r$ is the scalar reward from each simulation. $\pi(a|s)$ represents the probability of taking a certain action ($a$) when being on a certain state ($s$) and following the policy $\pi(s)$. $p(s',r|s,a)$ is the transition function, defined in Section 6.6.

The state value function $V(s)$ represents the expected return from being in state $s$. It can be expressed by the Bellman state value equation:

$$V(s) = \sum_{a \in A} \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')] \tag{4.4}$$

RL algorithms try to learn a policy that maximizes $V(s)$, so under optimality conditions the following equality would be true:

$$V(s) = \max_{a \in A} Q(s,a) \tag{4.5}$$

## 4.2 Model free RL

### Q learning

The core idea behind Q-learning is to learn a Q-value function that represents the expected value of taking a particular action in a particular state. The Q-value function can be represented as a look-up table, where the entries are updated iteratively based on the agent's experiences.

The Q-value update rule is based on the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left( r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

Where:

- $Q(s,a)$ is the Q-value of taking action $a$ in state $s$.
- $r$ is the observed reward.
- $\alpha$ is the learning rate.
- $\gamma$ is the discount factor.

The rationale of Q-learning is to minimize the error of the Bellman equation using fixed point iteration. There are two hyper-parameters to tune. The intuition behind it is as follows:

- $\alpha$ represents how much new experiences are trusted. A high value of $\alpha$ means that a high weight is given to the observed reward, while a low value means that more importance is given to the current Q-value.
- $\gamma$ represents the importance of future rewards with respect to current rewards, as in other RL algorithms. A value of gamma close to 1 means that the future rewards are almost as important as the immediate reward. While a value of gamma close to 0 means that the immediate reward is prioritized.

The policy with this algorithm would be to look at the Q table and select the action with the higher value for the current state.

$$\pi(s_t) = \arg\max_{a} Q(s_t,a)$$

However, an $\varepsilon$ greedy strategy is typically used to favor exploration. This essentially consists of randomly picking some of the actions, instead of using the previous expression. The parameter $\varepsilon$ defines what portion of the actions are randomly selected and it is typically decreasing as training advances.

## Policy Gradient

Policy Gradient methods [Sutton and Barto, 2018] are a class of reinforcement learning algorithms that directly learn the policy function, which maps states to actions, to maximize the expected cumulative reward. Unlike value-based methods that aim to estimate the value function and then derive a policy from it, policy gradient methods directly optimize the policy parameters using gradient ascent.

The basic idea behind policy gradient methods is to parameterize the policy with some function approximator (such as a neural network) and then update the parameters in the direction that increases the expected cumulative reward. This is typically done by computing the gradient of the expected reward with respect to the policy parameters and then performing gradient ascent.

Mathematically, the objective of policy gradient methods is to maximize the expected return $J(\theta)$, where $\theta$ represents the parameters of the policy:

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t R_t]$$

where $\gamma$ is the discount factor, $T$ is the time horizon, and $R_t$ is the reward at time step $t$.

Note that in this expression, the sub-index $t$ refers to a discrete time step of the sequential decision problem. Which, in the context of this work, consists of one simulation until steady state or a simulation during a fixed time step, as will be described in Section 7.3.

## Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO) [Schulman et al., 2015] is a policy gradient algorithm that aims to improve stability and sample efficiency compared to basic policy gradient methods. TRPO constrains the policy update step to ensure that the new policy remains close to the old policy, thus avoiding large policy updates that could destabilize learning.

The key idea behind TRPO is to maximize the following surrogate objective function, subject to a constraint on the KL-divergence [Pollard, 2000] between the old and new policies, which is a measure of how much the policy varies between iterations:

$$\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\text{old}}} \left[ \frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)} A^{\pi_{\text{old}}}(s,a) \right],$$

$$\text{subject to} \quad \mathbb{E}_{s\sim\pi_{\text{old}}}[D_{\text{KL}}[\pi_{\text{old}}(\cdot|s)||\pi_\theta(\cdot|s)]] \leq \delta,$$

where $A^{\pi_{\text{old}}}(s,a)$ is the advantage function under the old policy, and $D_{\text{KL}}$ is the KL-divergence between the old and new policies.

The advantage function, denoted as $A^\pi(s_t,a_t)$, represents the advantage of taking action $a_t$ in state $s_t$ compared to the expected value of being in state $s_t$ under the current policy $\pi$. It's calculated as the difference between the Q-value and the state value:

$$A^\pi(s_t,a_t) = Q^\pi(s_t,a_t) - V^\pi(s_t),$$

where:

- $Q^\pi(s_t,a_t)$ is the action value function, representing the expected cumulative discounted return of taking action $a_t$ in state $s_t$ and then following policy $\pi$ thereafter.
- $V^\pi(s_t)$ is the state value function, representing the expected cumulative discounted return starting from state $s_t$ and then following policy $\pi$ thereafter.

Again, in these expressions, the sub-index $t$ refers to a discrete time step of the sequential decision problem.

TRPO uses conjugate gradient optimization or other constrained optimization techniques to solve the above optimization problem efficiently.

## Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) [Schulman et al., 2017] is a further improvement over TRPO, designed to address some of its limitations and make it more practical and efficient. The main outcome of PPO, in contrast with TRPO, is that there are no hard constraints in the loss functions. PPO incorporates a soft constraint directly in the definition of policy loss by using a clipped surrogate objective function. This clipping has a similar effect as the hard constraint in the TRPO loss function and eliminates the need for complex constraint optimization.

In PPO, the objective function is modified to prevent aggressive policy updates that could lead to instability. Instead of directly maximizing the ratio of new policy probabilities to old policy probabilities, PPO clips this ratio to keep it close to 1. This is done using a hyperparameter $\varepsilon$:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_{\hat{t}}\left[\min\left(r_t(\theta)A_{\hat{t}}, \text{clip}\left(r_t(\theta), 1-\varepsilon, 1+\varepsilon\right)A_{\hat{t}}\right)\right],$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$, and $A_{\hat{t}}$ is the advantage function:

$$A_t^\lambda = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1},$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$.

Again, in these expressions, the sub-index $t$ refers to a discrete time step of the sequential decision problem.

This loss function essentially incorporates a soft constraint for avoiding aggressive updates of the policy paramters. Since there are no hard constraints, this makes the optimization problem more computationally efficient than in TRPO.

PPO also incorporates a value function loss term $L^{\text{VF}}$ and an entropy term $S[\pi_\theta]$ in the final loss function to encourage exploration and improve policy robustness. The overall objective function in PPO is given by:

$$L^{\text{CLIP+VF+S}}(\theta) = \mathbb{E}\left[L^{\text{CLIP}} - c_1 L^{\text{VF}} + c_2 S[\pi_\theta](s_t)\right],$$

where $c_1$ and $c_2$ are coefficients that balance the importance of the value function loss and the entropy term, respectively.

PPO uses batches of data to update its policy, it runs the policy for a while in the environment and collects data, which is used to compute the expectations in the loss function. Then it performs a number of gradient optimization iterations on the network parameters using the total loss function. A pseudocode of the algorithm is shown in Figure 4.4

```
Algorithm 1 PPO, Actor-Critic Style
  for iteration=1, 2, ... do
    for actor=1, 2, ..., $N$ do
      Run policy $\pi_{\theta_{\text{old}}}$ in environment for $T$ timesteps
      Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$
    $\theta_{\text{old}} \leftarrow \theta$
  end for
```

**Figure 4.4** Pseudocode for PPO. Source: [Schulman et al., 2017]. The actors are threads that run a given policy to collect data. N actors run in parallel to collect data. By default N = 1.

## Twin Delayed Deep Deterministic Policy Gradient (TD3)

Twin Delayed Deep Deterministic Policy Gradient (TD3) [Fujimoto et al., 2018] is an extension of the Deep Deterministic Policy Gradient (DDPG) [Silver et al., 2014] algorithm, designed to improve its stability and performance.

In TD3, two separate neural networks (NN), referred to as the "twin" critics, are employed to estimate the action-value function. This helps mitigate overestimation bias in the Q-value estimates.

Moreover, TD3 introduces a policy smoothing noise during action selection to reduce the variance of the Q-value estimates, which further enhances stability. Additionally, TD3 uses target policy smoothing, which adds noise to the target policy during the critic update step to prevent overestimation.

The TD3 algorithm has shown improved performance and stability compared to DDPG, especially in environments with high-dimensional state spaces.

## 4.3 Model based RL

Model-based RL is a method that uses some knowledge about a physical model to select the right actions.

It involves learning or knowing a model of the environment to improve decision-making. These methods can include planning algorithms to simulate future trajectories and learn from them, to decide which is the right action to take.

Model-based RL is known for being more sample-efficient and stable than model-free RL, which makes it an interesting alternative for control-related problems, however, it is more difficult to implement.

### Planning and learning

"Planning and learning may actually be combined, in a field which is known as model-based reinforcement learning" [Moerland et al., 2023]

Figure 4.5 illustrates the planning and learning framework introduced in [Moerland et al., 2023].

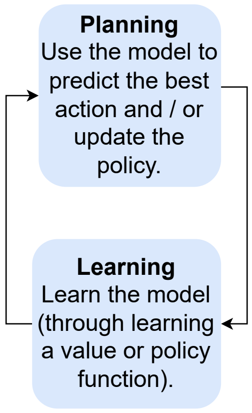

**Planning**
Use the model to predict the best action and / or update the policy.
**Learning**
Learn the model (through learning a value or policy function).

**Figure 4.5** Planning and learning. Essentially this consists of iterations of system identification and controller optimization.

This consists essentially of running several iterations of system identification (learning) and policy improvement (planning).

## 4.4 RL Environment

In the context of Reinforcement Learning (RL), an environment serves as the simulation or representation of the real-world system with which an agent (controller) interacts. Technically, it consists of a Python class where one can define the problem, including a representation of the dynamics of the system. The interaction between the agent and the environment occurs over discrete time steps. At each time step, the agent observes the current state of the environment, selects an action from its action space, and receives feedback in the form of rewards and potentially a new state. This process continues iteratively during the training, with the agent aiming to learn an optimal policy that maximizes its cumulative reward over time. For this work, the package gym, from OpenAI has been used to generate the environment.

## 4.5 Elements of the environment

### State Space

The state space (or observation space) represents the set of all possible configurations or states that the system can be in. Those states encapsulate all relevant information necessary for decision-making within the environment. One can arbitrarily decide what variables or information are included in the state space of the RL controller. A detailed definition of the state definition for the problem attained in this work is given in Section 7.2.

### Action Space

The action space represents the set of all possible actions that the agent can take within the environment. It is where one can define the number and topology of actions.

In the context of the cooling problem, there are 2 binary actions and 3 continuous actions. Different RL models have been built, considering all or a subset of the actions on each model. More details about the action space definition for each RL model are given in the next chapter.

The agent selects actions based on its current observation of the environment.

### Reward

At every time step, the environment should return a reward, based on how good was the action applied for the particular state. The reward function is a critical concept in the definition of an RL problem since it plays a crucial role in defining the objective of the algorithm. For this work, the reward function must guarantee that the power consumption is minimized and the cooling demand constraint is met. The next chapter introduces different definitions of reward functions for each of the models that will be proposed.

### Episode definition

When training an RL agent, the simulations are often organized in episodes. In a video game context, one episode would typically consist of one level, which ends when either the player completes the level or dies. The environment is restarted at the end of each episode. In the context of this work, one episode consists of several simulations with different weather and load conditions. For example, one episode could correspond to the weather and load data from one year discretized using a certain time-step length or to a number of synthetic combinations of weather and load.

## 4.6 Main methods of the environment

OpenAI's Gym package [Brockman et al., 2016] was used for this work. It provides a standardized interface for creating RL environments, which are compatible with most of the RL packages and easy to use for testing own implementations of algorithms. An environment is represented by a class in Python, that contains at least the following main methods:

- **Init():** This is the method that initializes the environment, here one can define the topology of the state and action spaces. In other words, specify the type (discrete, continuous), number, and range of each state and action variable.
- **Step(action):** The step method takes an action as input, applies it to the environment, and returns four values: the next observation (state), the reward obtained from taking the action, a boolean indicating whether the episode has terminated, and additional information useful for debugging or analysis.
- **reset():** This method initializes the environment to its initial state and returns the initial observation. It is typically called at the beginning of an episode or whenever a new episode starts.

# 5

# RL Basic Experiment

In this chapter, a basic experiment using Q-learning is performed. The motivation is to better understand the reinforcement learning fundamentals and the limitations of Q-learning before introducing more complex algorithms.

## 5.1 Reduced problem formulation

In this problem formulation, weather conditions and IT load are constant. The states are given by:

- Chilled leaving water temperature difference between chiller 1 and 2, $T_{diff}$.
- The cooling water leaving water temperature, $T_{cond}$.

The actions are:

- The set-point of the chilled leaving water temperature difference between chiller 1 and 2, $T_{diff,spt}$.
- The set-point of the cooling water leaving water temperature, $T_{cond,spt}$.

Summarizing, the state (S) and action (A) spaces are:

- S = [$T_{diff}$, $T_{cond}$]
- A = [$T_{diff,spt}$, $T_{cond,spt}$]

The admissible values for these variables are given by their respective validity ranges of the system.

Let's denote the state transition function:

$$s_{t+1} = f(s_t, a_t)$$

The function $f$ is evaluated by simulating the dynamic model for the thermal cooling system for a given time step $\Delta t$.

The reward function is given by:

$$R = \begin{cases} -P & \text{if } (T_{\text{IT}} - T_{\text{sp IT}}) < 0,1°C \\ -\infty & \text{else} \end{cases}$$

, where P is the steady-state power consumption. This is to reflect that the main constraint is to maintain the IT space temperature set-point, and the main control objective is to minimize the power required to operate the cooling equipment.

## 5.2 Experiment

Given this simplified problem formulation, an RL controller was trained using Q-learning. The simulation time-step was set sufficiently long to achieve steady state, more concretely one hour.

The outdoor conditions and cooling load were fixed:

- Outdoor temperature: 20 $°C$
- Relative humidity: 0,8
- Cooling load: 1250 kW

The evaporator leaving water temperature was fixed to 6,67 $°C$.
The hyper-parameters were set in the following way:

- $\alpha$ was set to 1, to make the learning converge faster.
- $\gamma$ was set to 0 and the episode length was set to one since each time step is independent of the other.
- The initial $\varepsilon$ was set to 0.9, with a linear decay rate of 0.0001

The set-points and state values were discretized into 5 values for $T_{diff}$ and 10 for $T_{cond}$, as shown in Table 5.1. To plot the Q table, the arrays were flattened as well.

| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $T_{diff}(°C)$ | 0 | 2,08 | 4,17 | 6,25 | 8,33. | | | | | |
| $T_{cond}(°C)$ | 15 | 16,60 | 18,21 | 19,82 | 21,42 | 23,02 | 24,63 | 26,23 | 27,84 | 29,44 |

**Table 5.1** Discretization

Since the second state has 10 possible values, then the states and actions can be easily expressed in flattened index or array formulations. For example, the flattened state 34 would correspond to the array of discrete indices [3,4] which corresponds to $T_{diff}$ = 6,25 $(°C)$ and $T_{cond}$ = 21,42 $(°C)$ according to Table 5.1. The same logic applies to the actions.

The algorithm converged in around 10000 time steps. Looking at the Q-table, shown in Figure 5.1 one can realize two things:

1. The state doesn't affect which action is the best. In other words, each action has a different Q-value that is not dependent on the current state of the problem. This is accordant with the intuition behind Assumption 3.2

2. Not all states are explored since not all states are reachable. For example, a too-low condenser temperature cannot be reached, since the cooling towers saturate. More concretely, it is observed that condenser temperatures lower than the fourth discrete value and the last discrete differential temperature value are never reached. This is not a problem, as long as the cooling demand is supplied, which is the case for this experiment.

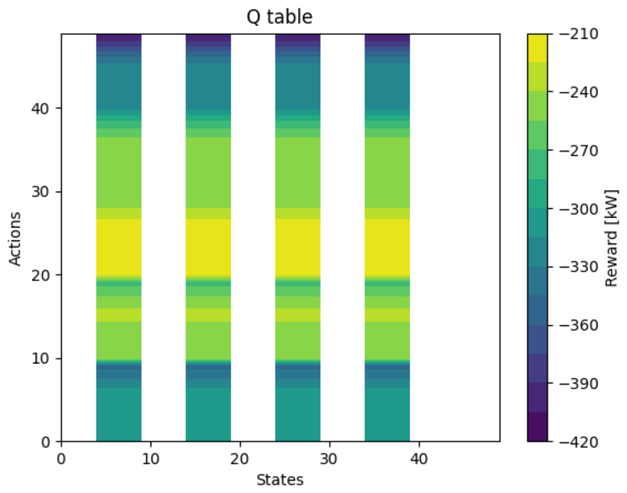


**Figure 5.1** Static experiment Q-table color map. The best action does not depend on the physical state of the system.

The best action selected by the algorithm was action 25, which corresponds to the discrete action pair [2, 5] which corresponds to $T_{diff,spt}$ = 4,165 $°C$ and $T_{cond,spt}$ = 23,02 $°C$. This is always the best action without depending on what state the plant is in, as shown in Figure 5.3.

The value function $V(s)$ represents the outcome of being in a particular state $(s)$. In the case of Q_learning it is defined as:

$$V(s) = \min_a Q(s,a)$$

The function is represented in figure 5.2, where the unexplored states are not plotted.

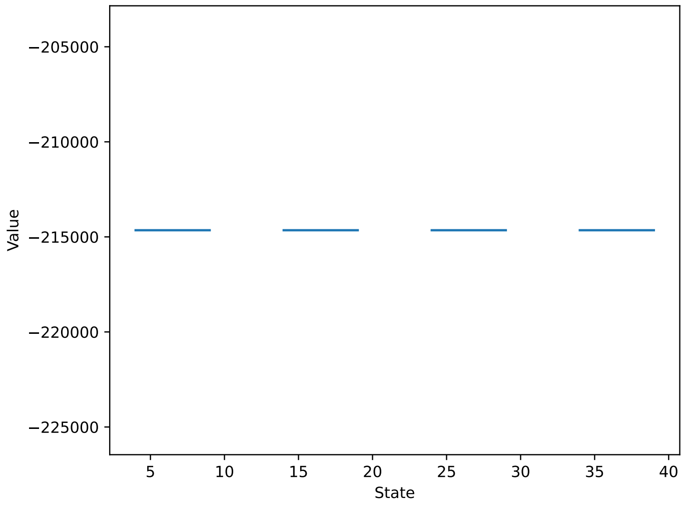


**Figure 5.2** Value function. The value of each of the explored states corresponds to the maximum Q-value for that state. It is seen that all explored states have the same value.

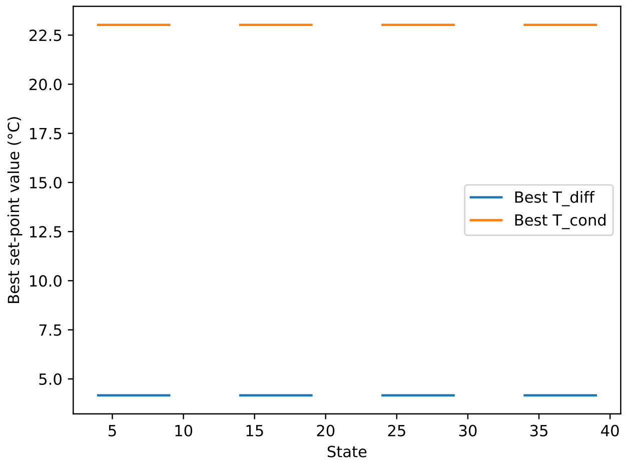


**Figure 5.3** Best $T_{diff}$ and $T_{cond}$ for each state. Note that the best action is the same one, independently of the current physical state.

## 5.3 Conclusions

There are 2 main conclusions from this experiment.

- Q-learning seems to be working for this simple problem. The results are accordant to the exploratory data analysis performed in Chapter 2. More concretely, the weather of this experiment corresponds to a wet-bulb temperature of 17,52 $°C$. Taking a look at Figure 2.14, it can be seen that the optimal $T_{diff,spt}$ found is close to this curve for the load value 1250 kW and temperature 20 $°C$, being the chosen value the discrete value of the list closer to the optimal. Similarly, taking a look at Figure 2.15, specifically to the second row left one (corresponding to the experiment conditions), the optimal $T_{cond}$ is close to the maximum efficiency point shown in the figure.

- The main disadvantage of tabular Q-learning is that the size of the Q-table grows exponentially with the number of actions and states. That makes this algorithm unsuitable for complex problems. Note that the problem solved in this section is not the same as the one described in the original problem formulation (Chapter 6). The actual problem has more actions and states, besides, to have an efficient controller, the actions should be either continuous or with more discrete values than the ones considered in this section. This makes Q-learning a non-suitable algorithm for the problem attained in this work, instead more complex algorithms should be studied, such as DQN, PPO or TD3.

# 6

# Problem Formulation

Here, the problem is formulated in a Powell fashion, according to the Book [Powell et al., 2022]. Only set points for the regulatory control layer and staging of components will be optimized. In further attempts, optimizing the controller parameters can also be explored.

This problem can be modeled as a Markov Decision Process (MDP), which is done at the end of this chapter.

## 6.1 The narrative

Our problem has a sequential decision nature since the supervisory control has to apply an action at each time step based on the status of the plant and external information such as the outdoor temperature or the IT load. The goal is to reduce the total energy cost of the data center, focusing on the energy used by the cooling system, as well as to keep the servers safe.

As mentioned in Chapter 2, The system consists of a chiller plant where chilled water is produced and an IT room. The chilled water is transported to the air handling units in the IT room (AHUS) that supply cold air to the IT equipment, according to figure 2.1.

The cooling load can take different values between 500 kW and 2000 kW.

It could be considered that electricity has different prices at different times, but to be able to minimize energy costs in that manner, energy storage would be needed. Therefore the goal will be to minimize the power consumption regardless of the different prices at different times.

There are low-level control architectures in the building that track certain set points for the relevant variables, located at the BMS control layer (see Figure 2.2). The goal is to develop a high-level controller that achieves the objective of minimizing power consumption by adjusting these set points and the staging of the equipment.

It is important to optimize these set points, otherwise the system can use much more energy than it needs to cool down the IT equipment.

## 6.2 Optimization possibilities

The complete list of variables to optimize can be found in Table 2.1. However, some variables cannot be optimized which must have a certain fixed value for reasons of security, comfort, or achieving the cooling objectives (temperature of the IT room, and supply air temperature).

Moreover, it has been decided not to modify the staging of the chilled water pumps. One reason is that the energy consumption on them is not very big compared to the rest of the elements and the aim is to keep the problem simple. Moreover, optimizing them doesn't have a big impact on power consumption and it is always better to have all the pumps running, as shown in Figure 2.17.

This leaves us with the set of variables to optimize shown in Table 6.1

| **Variable** | **Description** |
|---|---|
| $CH1_{status}$ | Status (on/off) of chiller 1 |
| $CH2_{status}$ | Status (on/off) of chiller 2 |
| $T_{evap,spt}$ | Chilled water temperature set-point |
| $T_{diff,spt}$ | Chilled water temperature difference set-point between chillers 1 and 2 |
| $CT1_{status}$ | Status (on/off) of the cooling tower 1 |
| $CT2_{status}$ | Status (on/off) of the cooling tower 2 |
| $T_{cond,spt}$ | Cooling tower leaving water (condenser entrance) temperature set-point value |

**Table 6.1** Optimization Variables

According to the variation ranges (2.1) - (2.3), the following constraints must be considered:

$$6.67°C < T_{evap,spt} < 15°C \tag{6.1}$$

$$0°C < T_{diff,spt} < 15°C - T_{evap,spt} \tag{6.2}$$

$$15°C < T_{cond,spt} < 29.45°C \tag{6.3}$$

Moreover, under certain conditions, the degrees of freedom of the optimization problem are reduced. For example, for high loads (roughly over 1300 kW) it is necessary to have the two chillers on to be able to meet the demand (so the temperature in the IT room remains at its set point in steady-state).

Furthermore, the rate at which the equipment is turned on and off must be limited, since too many changes on the staging are harmful to the equipment and imply a higher transient power consumption.

There are two ways to implement these constraints:

1. **Soft constraints:** Adding a penalty on the objective function for not meeting the constraint. For example, a fixed penalty when a staging variable is modified within a fixed time interval since it was modified the last time. With these constraints, it is not guaranteed that the condition will always be met. However, it is possible to define (by tuning the penalty terms) how harmful it is to not meet the condition and allow the system to disregard it if it's really beneficial according to the optimization criteria.

2. **Hard constraints:** Specifying a constraint that must always be met. This needs to be done in the policy definition. With this approach, the constraints will always be met.

In this work, hard constraints will be used, since equipment safety is of critical importance.

To reduce the dimension of the combinatorial problem, some constraints need to be imposed on the boolean decision variables. The idea is that when there is only one chiller working, that should be chiller 1. In the same way, when there is only one cooling tower working, that should be cooling tower 1.

In this sense, the reinforcement learning model will not need to differentiate between the case where only chiller 1 is running and the case where only chiller 2 is running. Since they are essentially the same case, the second option is just not allowed to happen, the same applies to the staging of the cooling towers.

To implement this, the RL model will have 2 boolean actions. One corresponds to the chillers and one corresponds to the cooling towers. When the chiller action takes the value 0, it means that only chiller 1 is on and when it takes the value 1 it means that both chillers are on. The same applies to the cooling towers with their associated action. This approach imposes a hard constraint on the equipment staging.

## 6.3 State variables

For the general problem (not taking into account the assumption in section 3.2), the state of the system consists of variables that reflect the internal conditions of the plant.

The state variables considered for this problem are shown in Table 6.2:

| Variable | Description |
|---|---|
| $T_{evap}$ | Evaporator leaving water temperature. |
| $T_{cond}$ | Condenser entry water temperature. |
| $T_{diff}$ | Evaporator leaving water temperature difference between chiller 1 and 2 |
| $CH1_{status}$ | Status (on/off) of chiller 1 |
| $CH2_{status}$ | Status (on/off) of chiller 2 |
| $CT1_{status}$ | Status (on/off) of cooling tower 1 |
| $CT2_{status}$ | Status (on/off) of cooling tower 2 |

**Table 6.2** State Variables

Note that the state variables are not the same as the decision variables. The first ones refer to the real value of each particular physical variable, while the later ones refer to the set point for that particular variable. The state variables might not have the same value as their setpoint, even in steady state. This is due to equipment saturation (the equipment is not capable of reaching one particular set-point, but still meets the cooling requirements).

The importance of this variables in decision making will be explored later in this work. A different definition of state variables for the RL controller will be given, which will include only the variables containing relevant information for the control problem.

## 6.4 Performance metrics

The performance metrics evaluate the power consumption of each element and the total system. The variables shown in Table 6.3 can be used as those.

| Variable | Description |
|---|---|
| $P_{ct}$ | Power consumption at time t in the chillers. |
| $P_{tt}$ | Power consumption at time t in the cooling towers. |
| $P_{at}$ | Power consumption at time t in the AHUs. |
| $P_{pt}$ | Power consumption at time t in the chilled water pumps. |
| $P_t$ | Total power consumption at time t. |
| $COP$ | $cooling\_load/P_t$ |

**Table 6.3** Performance metrics

## 6.5 Exogenous information

The exogenous information in our problem is the weather conditions and the cooling load. The variables are summarized in Table 6.4

| Case | Decision variables |
|---|---|
| $T\_db_{t+1}$ | Dry-bulb temperature of the outdoor air. |
| $RH_{t+1}$ | Relative humidity of the outdoor air. |
| $load_{t+1}$ | Cooling load. |

**Table 6.4** Exogenous information

Note the index t+1 in the variable. This denotes that future weather conditions are exogenous information. The past and current weather conditions are known and the system must use them to estimate the optimal parameters.

## 6.6 Transition Function

The transition function describes how the state of the system evolves over time in response to the decisions made by the controller and the exogenous information (such as weather conditions and cooling load). In this context, it represents the dynamics of the cooling system and how it reacts to changes in the control inputs.

The transition function, denoted as $p(s_{t+1}, s_t, a_t)$, calculates the probability of transitioning from state $s_t$ to state $s_{t+1}$ given the action $a$. It encapsulates the dynamics of the cooling system and how it responds to changes in control inputs and exogenous information.

Using mathematical notation, the transition function can be expressed as:

$$p(s_{t+1}, s_t, a_t) = P(s_{t+1} \mid s_t, a_t) \tag{6.4}$$

This equation represents the probability of transitioning to state $s_{t+1}$ at time step $t+1$ given that the system is in state $s_t$ at time step $t$ and action $a$ is taken.

For this problem, the transition function will be given by performing a simulation starting from state $s_t$, taking action $a_t$ for given exogenous information $w_t$ and then reading the value of the state variables:

$$s_{t+1} = simulate(s_t, a_t, w_t) \tag{6.5}$$

## 6.7 Objective function

The goal is to minimize energy consumption under the constraint of meeting the cooling demand.

$$\begin{aligned} &\min_a \quad \int_{\text{year}} P(t)\,dt \\ &\text{subject to:} \\ &\quad T_{\text{IT}} < T_{\text{sp,IT}} + \beta \end{aligned}$$

In the equation, $T_{\mathrm{IT}}$ and $T_{\mathrm{sp,IT}}$ are the IT room temperature and its setpoint, respectively, $\beta$ is a small number, $P$ is the power consumption of the cooling plant.

The constraint ensures that the cooling demand is met, by ensuring that the room temperature is never higher than its setpoint with a small admissible error range.

## 6.8 Designing policies

The policy will be to use a physics-informed Reinforcement learning model to make the decisions. The RL algorithm will modify the decision variables, based on a reward proportional to the COP at each time step. The shape of the policy will be dependent on the algorithm used. Several experiments will be performed with different algorithms, therefore, different types of policies will be used.

For example, some algorithms are based on policy optimization, Proximal Policy Optimization (PPO) being an example of them, for those, the policy is represented by a Neural Network (NN) that predicts the best setpoints for each state observation. Other algorithms, such as Q_learning, store a table with the expected cost of selecting each action from a possible set of actions for each state observation from a set of observations. These algorithms simply select the best possible action according to the information stored in that table.

# 7

# RL Models and Algorithms

In the definition of the most suitable RL model, various factors must be considered and some parameters must be tuned. In this work, different experiments have been performed. This chapter provides a general description of the RL models that have been set up and tested. The last section summarizes the models by specifying which methods have been applied in terms of algorithm selection, action space definition, reward function, episode configuration, and physics information techniques.

## 7.1 Training process

The training process can be performed in two different ways:

- Offline, using a simulated environment of the real plant.
- Online, directly using the real plant.

The first method involves needing an accurate simulation environment that represents well the real plant. The approach would be to use that simulation model to train the RL controller and then deploy it to the real plant.

The following issues need to be considered when deciding what approach to follow:

- Flexibility of designing training scenarios. In simulation, there is more flexibility for this, for example, the weather and load can be varied as one desires and simulations can be performed until steady state. In reality, the controller should be able to train with realistic weather and load patterns, as a consequence, the algorithm may never learn what operating point is optimal for unexplored external conditions.
- Training speed. While training the model on the real plant would take years, training it in a simulated environment should only take a few hours.

- Safe training. Some actions can break the IT or cooling equipment under certain circumstances if performed on the real plant, however, in the simulation, this is not an issue. The agent can receive a big negative reward when such an action is taken, learning then that that action is not adequate for that particular state.

- Model accuracy. When training on a simulation, the model needs to be sufficiently accurate, the efficiency curves should be the same as for the real plant. This problem only applies to the simulation training approach.

In this work, the model is trained on a simulated model. However, some of the experiments study a particular scenario that should be able to resemble training in the real plant, as described later in this section.

## 7.2 State Variables for RL

RL algorithms converge faster if the dimension of the state is reduced, so the state variables should ideally include all the relevant information for decision-making, but not unnecessary information.

Since it is assumed that the problem is stateless (see Section 3.2), which was proved experimentally in Chapter 5, it has been decided to define the state variables for RL as:

$$S = [l, T_w] \tag{7.1}$$

Where $T_w$ is the wet bulb outdoor temperature and $l$ is the cooling load, both of them are normalized so their range is from -1 to 1. This is the recommended range for state variables when using RL packages such as stable baselines.

In this case, the only relevant information for deciding which action is more suitable at each time step is the load and weather conditions, which are modeled as exogenous information in Section 6.3. A more detailed motivation on why to use the wet-bulb temperature in the state space can be found in Section 7.5, as well as the details of how it is computed and used.

Provided this definition of state space, the simplified MDP in Figure 7.1 can be defined. This MDP also assumes deterministic sampling of conditions from a given list and discrete actions. Since the external conditions are the only information in the state space, the state transitions will always go from the current sample on the list to the next one. However, the same transition could be done with different actions (represented by different arrows) with different associated reward values. The behavior is similar to an MDP with an uncontrollable state. The RL algorithm aims to find the best action to be applied at each state.

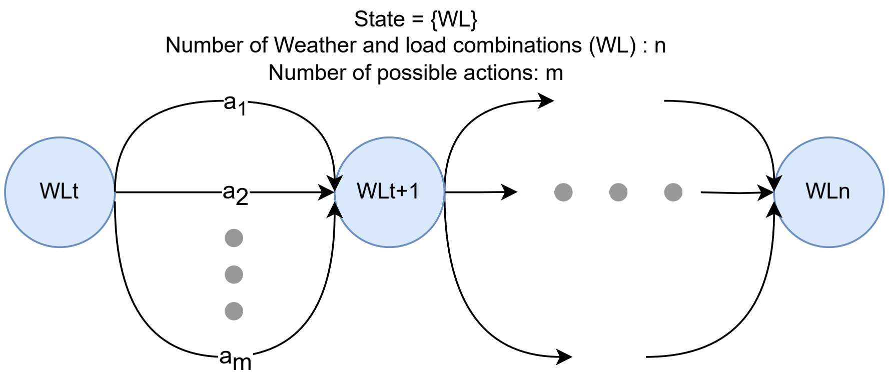

State = {WL}
Number of Weather and load combinations (WL) : n
Number of possible actions: m
$a_1$
$a_2$
$a_m$
WLt
WLt+1
WLn

**Figure 7.1** Simplified MDP for RL. One always moves one step forward in the weather and load list, but one can do that with many different actions. Each action has an associated reward depending on the current weather, load, and the action itself (one reward for each arrow in the diagram).

The state variables are usually modified at each time step according to the controller's action and the system dynamics. However, in this case, the state is modified by resampling the weather and load conditions after the simulation. In this way, the new normalized conditions are returned to the RL controller by the environment after each time step to compute the action for the next step.

## 7.3 Scenarios

The word scenario, in this context, refers to how the action space, reward function, and episode configuration are defined.

One common problem is that the environment has both continuous and discrete actions. The majority of algorithms don't support this. Q-learning and DQN need discrete action spaces, while PPO can work with continuous action spaces, but not with mixed (continuous and discrete) actions. One way to solve this is to discretize the continuous actions. Another solution might be to disregard the discrete actions and optimize only the continuous ones. In this work, the following scenarios have been defined:

***Discrete.*** In this scenario, the actions are all discrete and correspond to setpoints and staging. The action space is formed by the following variables:

- $CH1_{status}$: Boolean
- $CT1_{status}$: Boolean
- $T_{evap,spt}$: Discrete, 15 levels.

- $T_{cond,spt}$: Discrete, 15 levels.
- $T_{diff,spt}$: Discrete, 15 levels.

$$A = [CH1_{status}, CT1_{status}, T_{evap,spt}, T_{cond,spt}, T_{diff,spt}]$$

When training in this scenario, the simulations are performed until steady-state and the reward is defined in the same way as described in the problem formulation:

$$r_t = \begin{cases} \frac{L_t}{P_t} * \frac{1}{6} & \text{if } (T_{\text{IT}} - T_{\text{sp,IT}}) < \beta \\ -0.7 & \text{else} \end{cases} \tag{7.2}$$

Where t is the current time step (current simulation until steady state) and $L_t$ and $P_t$ are the values at the end of the simulation (steady state values). Note that this reward is always positive except when the cooling demand is not met. The average efficiency is around 6, so it is normalized by dividing it by this number since smaller rewards tend to improve training time.

One episode consists of all the simulations with all possible outdoor conditions and cooling load. The training is performed with loads between 500 and 2000 kW sampled deterministically from a linearly spaced list. The same kind of sampling for the $T_{db}$ and $RH$, that range between 12 and 30 ° C and 0.8 and 0.55 respectively. 20 levels are used for the load and 10 for the weather conditions, so the episode length is 200.

***Continuous.*** In this scenario, the actions are all continuous and correspond to setpoint tunning. The staging is fixed, and all the equipment is running. The action space is composed of the following variables:

- $T_{evap,spt}$: Continuous.
- $T_{cond,spt}$: Continuous.
- $T_{diff,spt}$: Continuous.

$$A = [T_{evap,spt}, T_{cond,spt}, T_{diff,spt}]$$

The reward function and episodes are defined in the same way as for the discrete scenario.

This scenario simplifies the policy network since the problem consists of a regression, with the observation as an input and action as an output, rather than a classification, which is the case for the discrete scenario.

However, the equipment staging optimization was not considered in this scenario. If this controller was implemented in the real plant, the rule from the baseline controller could be applied for cooling tower staging. However, this was not tested due to the time constraints of the project.

The chiller staging can be controlled with a single rule, turn off the second chiller when its set-point (set by the RL controller) is larger than the evaporator entering water $T_{return}$, which can be measured.

$$CH1_{status} = \begin{cases} 1 & \text{if } (T_{evap} + T_{diff}) < T_{return} \\ 0 & \text{else} \end{cases}$$

This is not necessary to implement when working on the simulated environment, since the power consumption of chiller 1 is considered zero automatically when the mentioned condition is met.

***Realistic.*** In this scenario, the actions are all continuous and correspond to set-point tunning. The staging is fixed, and all the equipment is running, as in the previous scenario. The action space is composed of the following variables:

- $T_{evap,spt}$: Continuous.
- $T_{diff,spt}$: Continuous.

$$A = [T_{evap,spt}, T_{diff,spt}]$$

$T_{cond,spt}$ is set according to the baseline controller, which controls it safely. Too low condenser temperatures reduce drastically the cooling capacity. Therefore, meeting the cooling demand depends highly on this parameter.

The purpose of this scenario is to study the feasibility of training on the real plant. With this action space, if the ranges are properly limited, it can be achieved that the simulation never fails for any action applied, which would mean that the cooling demand is met and the equipment safety is guaranteed, so the agent would be able to train on the real plant.

When training in this scenario, the simulations are performed for intervals of 20 minutes. The reward is defined as:

$$r_t = \begin{cases} \frac{L_t}{P_t} * \frac{1}{6} & \text{if } (T_{\text{IT}} - T_{\text{sp,IT}}) < \beta \\ -0.7 & \text{else} \end{cases} \tag{7.3}$$

Where t is the current time step (current simulation index). The Load ($L_t$) and Power ($P_t$) values are measured at the end of each simulation.

One episode consists of one one-year simulation, with weather data from Miami and the variable load pattern, described in Section 8.2

Similar policies as the ones described in the continuous scenario could be deployed for chiller and cooling tower staging when implementing the controller on the real plant.

## 7.4 Algorithms

As mentioned previously, the size of the action and state spaces makes tabular Q-learning not suitable for the problem. 3 different algorithms are therefore studied. Not all of them work with all the scenarios, due to their different action space nature.

- PPO: Works with continuous or discrete actions.
- TD3: Works only with continuous actions.
- DQN: Works only with discrete actions.

However, DQN was quickly discarded since it was observed that PPO converges much faster to a higher reward value, as seen in Figure 7.2. So the final results are only shown for PPO and TD3.

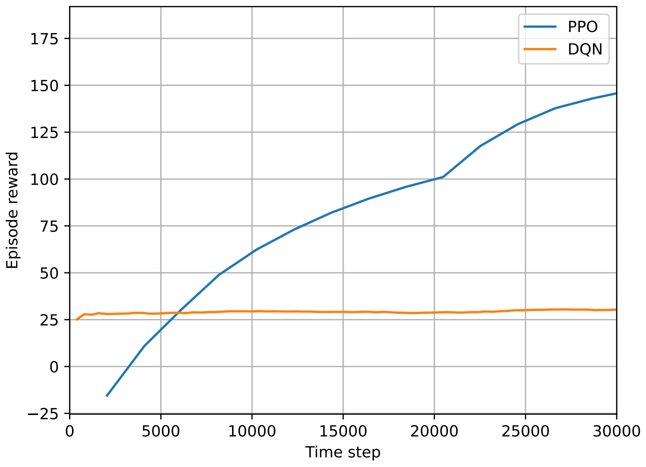


**Figure 7.2** Comparison training DQN and PPO, for the discrete scenario (described later in this section). PPO reaches higher reward values quicker than DQN.

### Python Library

It has been decided to use the python package stable baselines [Raffin et al., 2021], which provides implementations of these algorithms. This package is intuitive and easy to use, since it is compatible with gym environments and the algorithms come with a default set of hyperparameters suitable for different types of problems.

## Parameter Considerations

Several parameters affect the convergence speed of the algorithm when training. These parameters include:

- **Policy and Value Network Architectures**: This is the neural network architecture that represents the policy and value functions. In PPO, both typically consist of 2 hidden layers with 64 neurons with hyperbolic tangent activation functions, as well as the input and output layers. In this project, it has been decided to keep the main structure but change the hidden layer size to 32 and use Rectified Linear Unit (ReLu) activation functions when working with PPO. These parameters were obtained by trial and error. Two PPO models were trained, one used the described architecture and the other used the default architecture. It turned out that the model with the proposed architecture performed better on the test benchmark, described in Section 8.2.

- **Discount Factor ($\gamma$)**: This parameter determines how much future rewards are considered in the agent's decision-making process. Higher $\gamma$ values prioritize long-term rewards, while lower values focus more on short-term gains. Due to the assumptions in Sections 3.2 and 3.1, the intuition suggests setting $\gamma$ to 0. Since each time step is a simulation until a steady state, the outcome of future simulations should not influence how the actions at the current time step are selected.

- **Episode Length**: The episode length defines the duration of each episode in the RL training process. In the project's case, one episode consists of a set of situations with all the possible combinations of cooling loads and weather conditions. More concretely, 20 samples have been used for the cooling load and 10 for the weather, so the episode length is 200.

- **Entropy Coefficient**: This parameter can be used to boost exploration by adding entropy to the loss calculation. The parameter was kept to its default value of 0, to not use entropy. Note that this does not mean that there is no exploration at all. The policy used is stochastic, concretely, the policy network outputs the standard deviation ($\sigma$) and mean ($\mu$) of a normal distribution for each action. Then the actions are sampled from their respective distributions, where the predicted variance $\sigma$ is larger at the beginning of the training, due to random initialization of the neural network weights. Then $\sigma$ decreases as training advances, since the agent finds that some particular actions work better than randomly taken actions, favoring exploration in the initial steps and exploitation at the later steps.

## 7.5 Physics information

Incorporating domain-specific knowledge, such as physics information, into the RL framework can enhance learning efficiency and performance. The goal of leveraging physics information is to provide the agent with prior knowledge about the environment, enabling it to converge faster and make informed decisions.

[Banerjee et al., 2023] introduces a taxonomy of physics information techniques. In this work, two physics information techniques are introduced, the first one (optimal law for $T_{diff,spt}$) corresponds to the "action regulation" method, while the second one ($T_w$ computation) corresponds to the "state design" method in that taxonomy.

***Optimal law for*** $T_{diff,spt}$**.** As mentioned in the action space description, and relating to Section 2.1. The setpoint for $T_{diff}$ can be set according to the law:

$$T_{diff,spt} = \frac{T_{return} - T_{evap}}{2} \tag{7.4}$$

This ensures that both chillers have the same temperature difference (provided that they are not saturated, so the set points are met). This seems to be optimal according to Figure 2.14.

This policy reduces the dimension of the action space since it has one less variable to optimize, which should imply less time to train the RL controller.

***Wet-bulb temperature computation.*** Recalling Section 4.5, the state space is defined as the array:

$$S = [l, T_w] \tag{7.5}$$

where $l$ is the cooling load and $T_w$ is the wet bulb temperature. The wet-bulb temperature is used to reduce the dimension of the steady state. In the simulation environment, the weather is defined by the outdoor dry-bulb temperature ($T_{db}$) and the outdoor relative humidity ($RH$). One could include this weather information directly in the state space, but that would involve adding two dimensions representing the weather. Instead of doing that, the two variables can be used to compute the wet-bulb temperature and include only that variable in the state space, which is the approach taken in this work. The wet-bulb temperature is calculated using the following empirical equation, described in [Chen and Chen, 2022], where $T$ must be expressed in $^\circ C$ and $RH$ must be expressed as a percentage:

$$\begin{aligned} T_w = {} & T * atan(0.152 * \sqrt{RH + 8.314}) + 0.00392 * \sqrt{RH^3} \\ & * atan(0.0231 * RH) - atan(RH - 1.676) + atan(T + RH) - 4.686 \end{aligned} \tag{7.6}$$

Figure 7.3 illustrates this function. Note that the wet-bulb temperature is always equal to or lower than the dry-bulb temperature, they are only equal when the relative humidity is 100 %

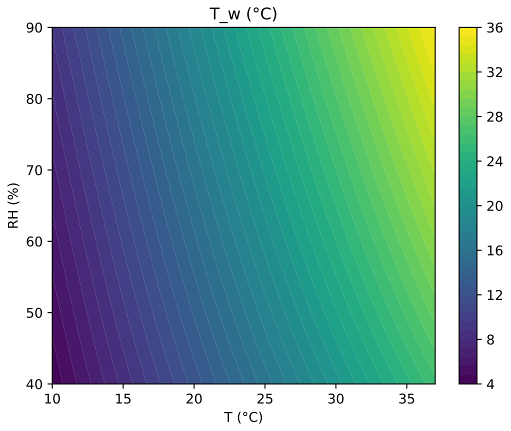


**Figure 7.3** $T_w$ function. It consists of a mapping from dry-bulb temperature and relative humidity to wet-bulb temperature.

For this to be a good approximation, $T_w$ must contain relevant information for the system. In other words, the power consumption must depend on $T_w$, rather than on $T_{db}$ and $RH$ individually, so all combinations of $T_{db}$ and $RH$ that correspond to the same value of $T_w$ should give the same value of power consumption. This is the case for the cooling system, the reason is that the weather affects mainly the efficiency of the cooling towers. The intuition behind this is that objects can be cooled down to the wet-bulb temperature, which is usually lower than the dry bulb temperature, without using additional energy. The other equipment units where the weather is relevant are the different pipes since the thermal loss of the pipes is computed based on the outdoor temperature, but the contribution of this to the total power is deprecable in relation to the cooling tower power consumption.

The cooling tower in the simulated environment has been modeled using Merkel's calculation method. A detailed explanation of the calculation is given in [Wetter et al., 2024]. The most significant aspects are described in this section. The heat exchange can be computed as:

$$Q = UA \times \frac{h_s - h_a}{c_p} \tag{7.7}$$

Where U is the cooling tower's overall heat transfer coefficient and A is the heat transfer surface area. The coefficient UA is computed as:

$$UA = UA_0 \times f_{UA,\text{wetbulb}} \times f_{UA,\text{airflow}} \times f_{UA,\text{waterflow}} \tag{7.8}$$

Where $f_{UA,\text{wetbulb}}$, $f_{UA,\text{airflow}}$ and $f_{UA,\text{waterflow}}$ are factors that depend on the wet-bulb temperature, airflow, and water flow respectively. So, the efficiency of the cooling towers, in the way that they have been modeled, is not dependant on the dry-bulb temperature and relative humidity individually, but rather on the wet-bulb temperature.

This is the case for the system, the weather affects mostly the efficiency of the cooling towers, which is computed as a function of $T_w$. The weather also affects the heat transfer in the different pipes, which is calculated as a function of $T_{db}$, but the effect of this on the total power consumption is neglectable since it is minimal, see Table 7.1

| $T_{\text{dry-bulb}}(°C)$ | RH | $T_w(°C)$ | Power Consumption (steady state) (kW) |
|---|---|---|---|
| 21 | 0.8 | 18.49 | 226.95 |
| 22.37 | 0.7 | 18.49 | 226.60 |
| 19.72 | 0.9 | 18.49 | 226.55 |
| 22 | 0.8 | 19.45 | 237.01 |
| 23.4 | 0.7 | 19.45 | 236.81 |
| 20.69 | 0.9 | 19.45 | 236.80 |

**Table 7.1** Experiments to test the effect of the dry-bulb temperature and RH on the power consumption individually. The table shows that the effect of these variables individually on the power consumption is deprecable, in relation to the effect of $T_w$. Experiment conditions: $T_{evap,spt}$ = 7.5 $°C$, $T_{cond,spt}$ = 23 $°C$, $T_{evap,spt}$ = 2.2 $°C$, load = 1200 kW, all the equipment running.

## 7.6 Experiments

Different RL models have been set, in the next chapter, the training and testing of each model is explored. The models, with their characteristics are summarized in Table 7.2.

| Model | Scenario | Algorithm | Physics information Techniques |
|---|---|---|---|
| 1 | Discrete | PPO | $T_w$ computation |
| 2 | Continuous | PPO | $T_w$ computation |
| 3 | Continuous | TD3 | $T_w$ computation |
| 4 | Continuous | PPO | $T_w$ computation, Law for $T_{diff}$ |
| 5 | Realistic | PPO | $T_w$ computation |
| 6 | Realistic | TD3 | $T_w$ computation |

**Table 7.2** Model setup

Model 1 uses the full action space but requires discretizing the continuous actions. Models 2 and 3 ignore equipment staging, and only optimize the continuous setpoints, the only difference between them is the algorithm used. Model 4 is like Model 2 but fixing $T_{diff,spt}$ to follow a physical law, this is expected to reduce the data requirements for training. Models 5 and 6 use a reduced action space to train safely in the real plant, and more realistic training conditions, their purpose is to study the feasibility of training on the real plant.

# 8

# Numerical Experiments

## 8.1 Training

### Discrete Scenario

Figure 8.1 shows the evaluation reward from the training process of Model 1 in Table 7.2, which uses PPO, and the discrete scenario, this is the reward obtained from running one episode with the current model at each evaluation step. This reward is evaluated every 2048 training steps, the same frequency at which the policy is updated. Note that one time step corresponds, for this model, to one simulation until steady state. It can be observed that the algorithm converges after approximately 160000 time steps.

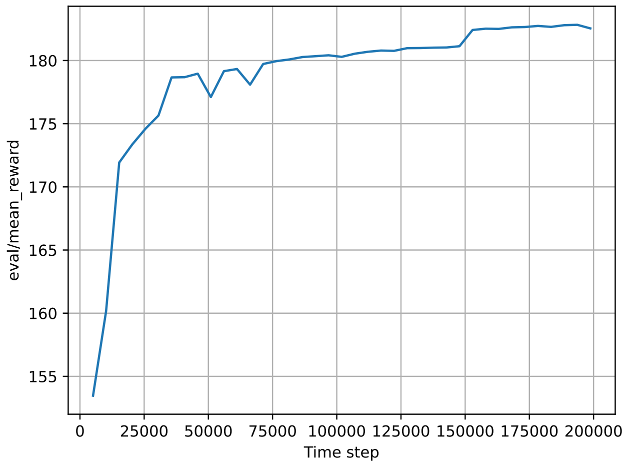


**Figure 8.1** Discrete scenario training. The algorithm converges after around 160000 time steps.

### Continuous Scenario

Figure 8.2 shows the evaluation reward from the training process of the models with the continuous scenario: 2, 3 and 4 in Table 7.2. These models use the algorithms PPO, TD3 and PPO respectively. The last one differs from the first in that it doesn't learn a policy for $T_{diff,spt}$, but instead it applies the physical law described in Section 7.5.

The reward is again evaluated every 2048 training steps, corresponding to the policy update frequency. As before, one time step corresponds, for all these models, to one simulation until steady state. The following can be concluded from the experiment:

- TD3 converges faster than PPO. It takes around 50000 time steps to converge, while PPO needs around 100000.
- TD3 also gives significantly better results at the beginning of the training process, which would be beneficial if training on the real plant.
- Using a physical law for $T_{diff,spt}$, which reduces the action space, improves the results at the beginning of training. This is logical since, if the assumption in Section 3.3 is fulfilled, this setpoint is assumed to always follow an optimal law, even at the beginning of training.

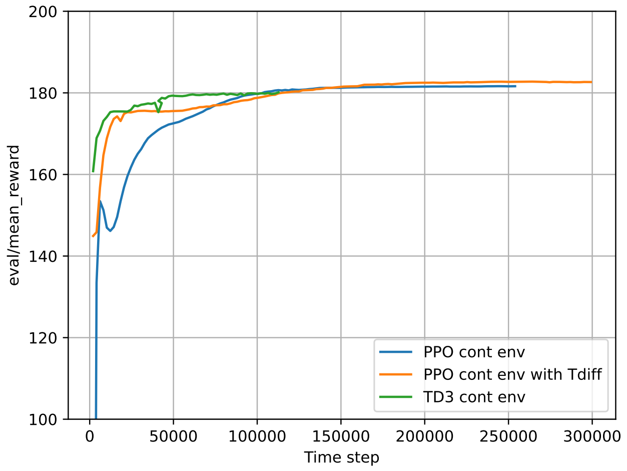


**Figure 8.2** Continuous scenario training. TD3 converges quicker than PPO, TD3 and the PPO model with the physical law for $T_{diff,spt}$ give better results at the beginning of training.

## Realistic Scenario

As mentioned before, the realistic scenario aims to resemble real training conditions, to evaluate the data requirements for training on the real plant and the performance of the algorithms trained in this way.

Figure 8.3 shows the rollout reward of the models that train in this scenario: 5 and 6 in Table 7.2.

To understand the concept of rollout reward, the PPO working principle needs to be taking into account. A fixed policy is executed on the environment to collect T (hyperparameter with default value 2048) data samples, that will be used to compute the advantage expectations for the next policy update, as shown in Figure 4.4.

The rollout reward is the average episodic reward gotten from the samples that are used on every policy update. However, in this case, since the episode length is higher than 2048, the rollout reward is computed at the end of each training episode. The main difference with the evaluation reward is that this is the reward observed on the training data, rather than the reward from a separate evaluation experiment. The reasons to use this metric are the following:

- The rollout reward is more realistic when the model trains on the real plant. Since this is the reward directly gotten from the samples used for training, which correspond to stochastic actions, they correspond to the real reward that one could expect when training online. The stochasticity in the actions is needed to favor model exploration and be able to properly train the model.

- The episode length for this scenario is very long in comparison to the number of training time steps, therefore, it would take very long to simulate one full episode for evaluation. That would correspond to evaluating the model over one year of data after it has been trained on one year of data, so it would double the training time.

It can be observed that with both algorithms, the reward increases over time and it seems that it will eventually converge. Note that, for this model, one episode consists of 26278 time steps, since it consists of a one-year simulation (corresponding to 2023) discretized in intervals of 20 minutes, rather than the previous 200 time steps. Therefore, it is expected that it will take longer to converge than the previous 50 or 150 thousand time steps since it takes a higher number of time steps to complete each episode.

In this case, PPO apparently outperforms TD3. The models have been trained for around 150 thousand time steps, corresponding to 5 complete years of training (5 complete episodes).

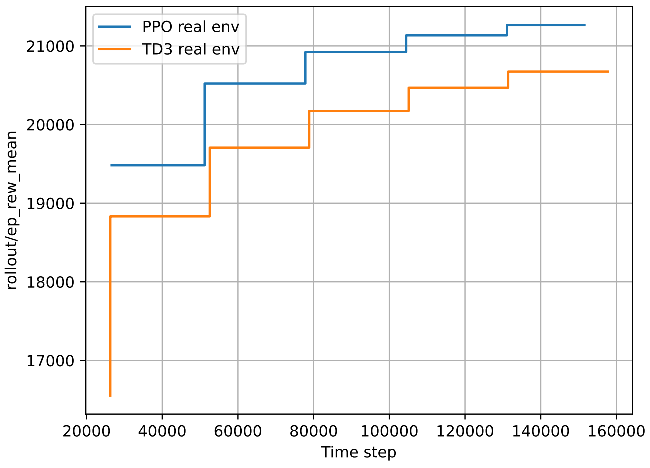


**Figure 8.3** Realistic scenario training. The reward increases over time with both algorithms, but PPO achieves higher reward values with the same training time.

## 8.2 Algorithm Evaluation

### Weather data

The weather data used is from Miami. It is sampled every hour and then interpolated to have data every 20 minutes since that is the control interval used when testing. As mentioned before, this might imply that the steady-state assumption (mentioned in Section 3.1) is not met. However, the aim of this decision is to develop a realistic testing scenario.

The outdoor temperature and humidity are shown in Figure 8.4.

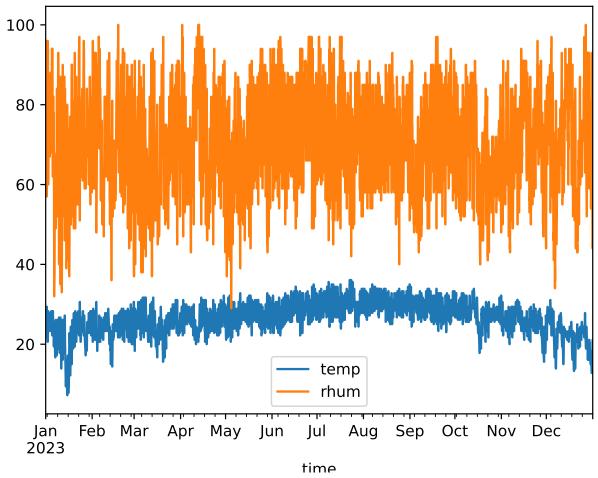


**Figure 8.4** Weather data from Miami in 2023. Temperature in $^\circ C$ and relative humidity as a percentage.

## Load Patterns

3 load patterns are defined:

***Constant.*** Constant loads at different levels: 500, 1000, 1500, and 1800 kW

***Sinusoidal.*** Offset loads at different levels: 500, 1000, 1500, and 1800 kW with an added sinusoidal component with amplitude 10% of the offset load and period 24 hours.

***Variable.*** Consists of 3 components, the notation: $N(\mu, \sigma)$ corresponds to a normal distribution with mean $\mu$ and standard deviation $\sigma$:

- Base load following a distribution $N(1200, 30)$ kW
- Added load on peak hours (19-23) following a distribution $N(400, 30)$ kW
- Added load on weekends following a distribution $N(150, 30)$ kW

So, this pattern has essentially 3 different offset levels for the load and an added noise component:

- 1200 kW on weekdays outside peak hours.
- 1600 kW on weekdays during peak hours.
- 1350 kW on weekends outside peak hours.

- 1750 kW on weekends during peak hours.

The load is clipped to a maximum value of 2000 kW. This aims to be a realistic representation of the real load pattern. A plot of the load can be found in Figure 8.5

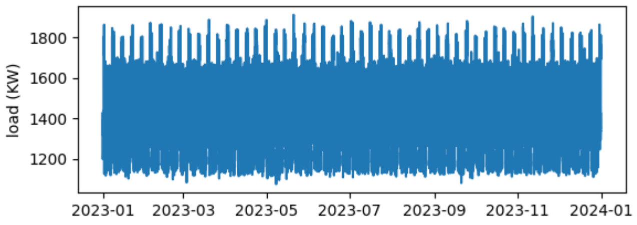


**Figure 8.5** Variable load pattern.

## Benchmark

The results from the RL controllers are compared with the baseline controller, in terms of total energy consumed, energy savings and set-point adjusting.

## Results

Tables 8.1 and 8.2 contain the yearly energy consumption from each model for different load patterns. Tables 8.3 and 8.4 contain the overall energy savings, with respect to the baseline controller.

| Algorithm | Scenario | Variable load | Constant load (kW) | | | |
|---|---|---|---|---|---|---|
| | | | 500 | 1000 | 1500 | 1800 |
| PPO | discrete | 2327 | 833 | 1652 | 2727 | NM |
| PPO | continuous | 2301 | 1040 | 1620 | 2709 | 3625 |
| TD3 | continuous | 2337 | 951 | 1701 | 2723 | 3625 |
| PPO with $T_{diff,spt}$ law | continuous | 2301 | 1053 | 1625 | 2702 | 3627 |
| PPO | realistic | 2371 | 958 | 1709 | 2768 | 3704 |
| TD3 | realistic | 2376 | 958 | 1709 | 2779 | 3708 |
| Baseline | | 2379 | 910 | 1707 | 2769 | 3747 |

**Table 8.1** Yearly energy consumption in MWh for the different RL models and the baseline controller. The data is shown for the variable and constant load patterns. NM means that the cooling demand was not met for that particular case, so the experiment should not be considered valid.

| Algorithm | Scenario | Sinusoidal load (offset) (kW) | | | |
|---|---|---|---|---|---|
| | | 500 | 1000 | 1500 | 1800 |
| PPO | discrete | 834 | 1659 | 2727 | NM |
| PPO | continuous | 1042 | 1624 | 2719 | NM |
| TD3 | continuous | 951 | 1695 | 2733 | NM |
| PPO with $T_{diff,spt}$ law | continuous | 1056 | 1627 | 2718 | NM |
| PPO | realistic | 982 | 1717 | 2779 | 3765 |
| TD3 | realistic | 973 | 1719 | 2790 | 3744 |
| Baseline | | 911 | 1714 | 2784 | 3782 |

**Table 8.2** Yearly energy consumption in MWh for the different RL models and the baseline controller. The data is shown for the sinusoidal load patterns. NM means that the cooling demand was not met for that particular case, so the experiment should not be considered valid.

| Algorithm | Scenario | Variable load | Constant load (kW) | | | |
|---|---|---|---|---|---|---|
| | | | 500 | 1000 | 1500 | 1800 |
| PPO | discrete | 2.18 | 8.46 | 3.24 | 1.90 | NM |
| PPO | continuous | 3.26 | -14.35 | 5.10 | 2.17 | 3.28 |
| TD3 | continuous | 1.75 | -4.43 | 0.36 | 1.68 | 3.26 |
| PPO with $T_{diff}$ law | continuous | 3.28 | -15.69 | 4.81 | 2.39 | 3.19 |
| PPO | realistic | 0.35 | -5.30 | -0.10 | 0.03 | 1.17 |
| TD3 | realistic | 0.11 | -5.30 | -0.13 | -0.36 | 1.03 |

**Table 8.3** Energy savings in % for variable and constant load patterns with respect to the baseline controller. NM means that the cooling demand was not met for that particular case, so the experiment should not be considered valid.

| Algorithm | Scenario | Sinusoidal load (offset) (kW) | | | |
|---|---|---|---|---|---|
| | | 500 | 1000 | 1500 | 1800 |
| PPO | discrete | 8.43 | 3.23 | 2.04 | NM |
| PPO | continuous | -14.42 | 5.23 | 2.31 | NM |
| TD3 | continuous | -4.34 | 1.07 | 1.82 | NM |
| PPO with $T_{diff}$ law | continuous | -15.90 | 5.05 | 2.36 | NM |
| PPO | realistic | -7.78 | -0.17 | 0.17 | 0.43 |
| TD3 | realistic | -6.78 | -0.31 | -0.23 | 0.99 |

**Table 8.4** Energy savings in % for sinusoidal load patterns with respect to the baseline controller. NM means that the cooling demand was not met for that particular case, so the experiment should not be considered valid.

Let's assume that the variable load pattern is the more realistic one, then it seems that the Continuous models with PPO, with and without the physics law for $T_{diff}$ seem to be the best one. Their efficiency is similar, varying slightly for each load pattern, so the main benefit of having a fixed control law for $T_{diff}$ is reflected when training the model, as discussed previously, rather than in the controller performance. However, this law will not work if the chillers are different (the assumption in Section 3.3 would not be true). Therefore, if training the model offline, it would be better to include $T_{diff,spt}$ in the action space, since training time doesn't matter that much.

The discrete model with PPO performs a bit worse than the continuous, with the variable pattern and it doesn't meet the cooling requirements for high loads, but it performs significantly better with very low loads. The intuition behind this is that the discrete model is allowed to optimize as well the staging of the equipment, while the continuous model is forced to have all the equipment running. The staging of chillers, however, can be controlled through $T_{diff,spt}$, if the chiller 1 evaporator leaving water setpoint ($T_{evap,spt} + T_{diff,spt}$) is larger or equal than the return water temperature, the chiller 1 doesn't need to be on. Similarly, if the value for $T_{diff,spt}$ is 0, the chiller 2 doesn't need to be running. This is already accounted for in the simulation, which in those cases shows the same power consumption value as if the corresponding chiller was off, in reality, a rule-based controller for the chiller status that checks the previously mentioned conditions could be implemented. This is the reason why in Figure 2.14 one observes an increase and deviation from the physical law in the optimal $T_{diff,spt}$ for lower loads, since for those loads it would be better to only use one chiller, which would imply to set $T_{diff,spt}$ to either a high value or 0. However, it seems like the continuous model is not capable of learning this behavior, Figure 8.6 shows the action values for the continuous model with PPO, with constant load of 500 kW, it can be appreciated that $T_{diff,spt}$ is very different than the ideal values shown in Figure 2.14 for lower loads. The setpoint $T_{diff,spt}$ decided by the RL controller is instead close to the value from the physical law, so it could be the case that the controller learned that law from the larger load cases and generalized it to lower loads.

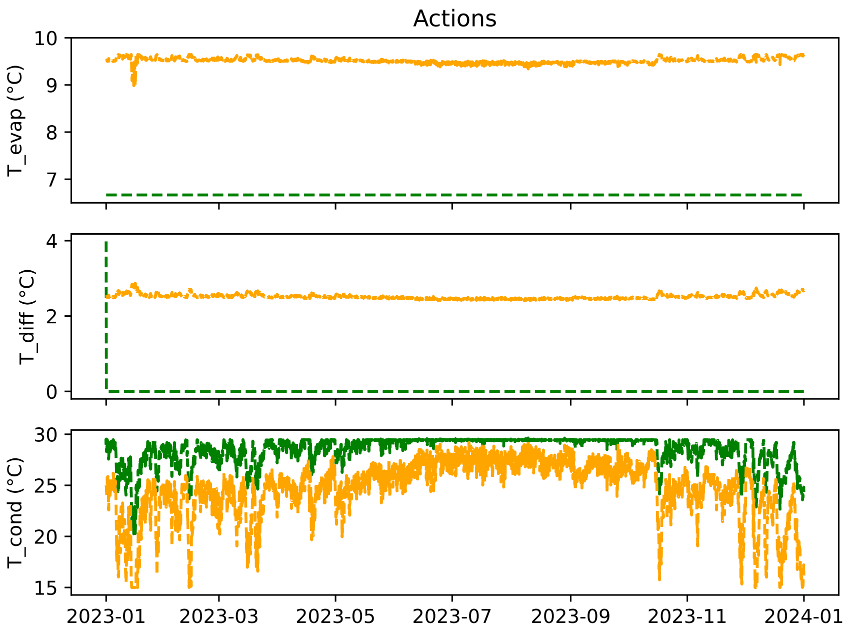


**Figure 8.6** Actions for the continuous model with PPO for a constant load of 500 kW. The green and yellow lines represent the actions from the baseline controller and RL controller respectively.

The staging of cooling towers is also not contemplated in the continuous models. However, both, cooling towers and chillers staging could be implemented on top of the RL controller, using the rules from the baseline controller. This would significantly increase the efficiency of the RL controller with lower loads, and hopefully maintain its current efficiency for higher loads. This would potentially result in a better controller than the one with discrete actions, especially taking into account that the savings with higher load values are more significant in terms of absolute energy, which is directly related to the energy cost.

Regarding the continuous models, it can be seen that PPO outperforms TD3 in testing, however, it is important to note that these experiments have been performed under certain conditions and hyperparameters and the results might be different with other settings. Something that could be explored is reducing the network sizes for TD3, as it was done for PPO, however, this was not explored due to the time constraint of the project.

Regarding the realistic models, it seems that again PPO outperforms TD3, this time in both, training rollout reward, as seen in Figure 8.3, and efficiency with variable load, when testing, as seen in Table 8.3. However, it is to note, that in both cases, the savings are pretty low when testing with the same data that the algorithm was trained (the variable load pattern). Moreover, it can be seen that the algorithm struggles to generalize to other loads, contrary to the other models, the savings are highest for the constant and sinusoidal pattern with an offset of 1800 kW. These load patterns are very close to one of the offset levels from the variable load pattern 1750 kW, so the algorithm has trained on the load region around them. The other load patterns result in even higher energy consumption than the baseline controller.

To analyze the outputs from the controllers, two weeks, corresponding to the winter and summer seasons, will be analyzed. Figure 8.7 shows the wet-bulb temperature and load conditions during the winter week, while Figure 8.8 shows the same data for the summer week. The outputs from the realistic and continuous models using PPO will be analyzed, for both weeks, the variable load pattern is used.

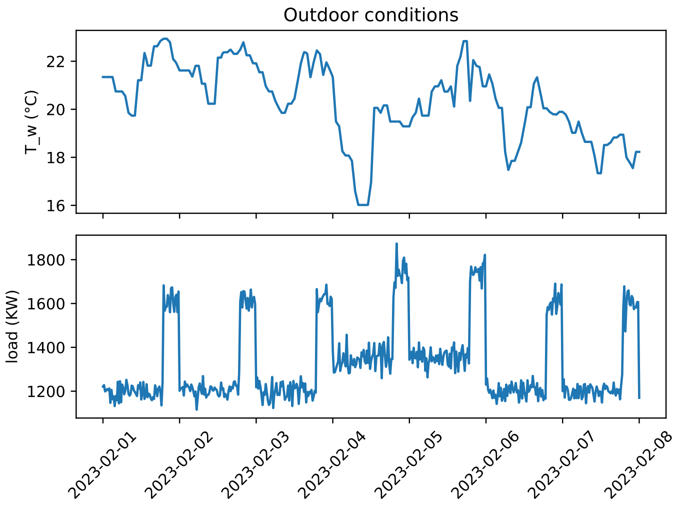


**Figure 8.7** Outdoor wet-bulb temperature and load, from the variable pattern, for a week in winter.

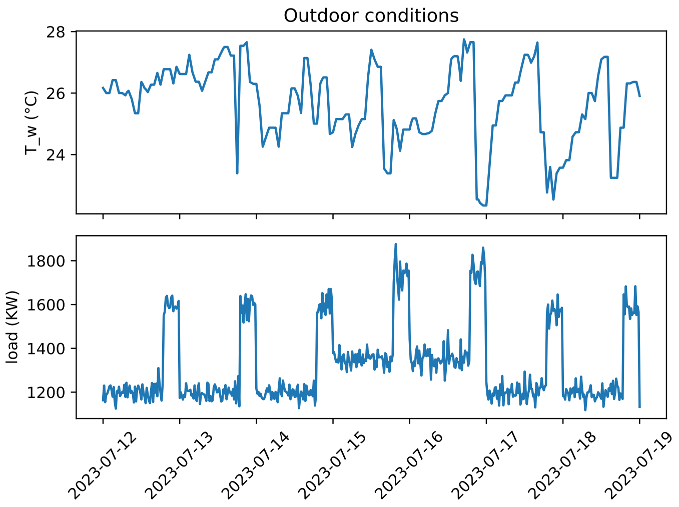


**Figure 8.8** Outdoor wet-bulb temperature and load, from the variable pattern, for a week in summer.

***Continuous model with PPO.*** This section explores the decisions taken by the RL controller with the Continuous scenario and PPO (model 2 in Table 7.2) when evaluating the performance during the proposed weeks.

Figures 8.9 and 8.10 show the RL actions for the winter and summer weeks respectively.

It seems that the RL controller learned to keep $T_{evap,spt}$ low, to prevent the chilled water pumps from consuming too much power when the load is relatively high (for this load pattern it is always higher than 50 %). So it fixes $T_{evap,spt}$ to the minimum and then adjusts $T_{diff,spt}$, to balance the load between the chillers. $T_{diff,spt}$ is highly dependent on the load, and it decreases when the load increases, as in Figure 2.14 for this load range.

The condenser temperature ($T_{cond,spdt}$) shows some influence from the cooling load, rather than depending purely on the weather, as the baseline controller law does. This load influence is mostly appreciated in the summer plot, where $T_{cond,spdt}$ is usually at its maximum value except when the load increases. The winter plot shows that this setpoint is mostly affected by the weather (it resembles to some extent the law from the baseline controller), with some influence from the load. Note that in both cases, the closed loop controller is very conservative with this setpoint, keeping it always higher (except when it is at its maximum value) than the RL controller. This means that lowering a bit $T_{cond,spt}$ favors saving energy, although, it decreases the cooling capacity and in some situations, it can lead to not meeting the cooling requirements.

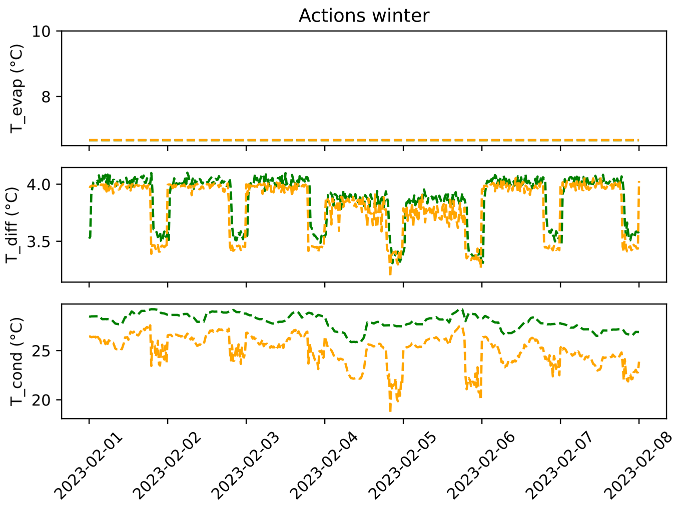


**Figure 8.9** Actions for the continuous model with PPO and variable load pattern, during a winter week. The green and yellow lines represent the actions from the baseline controller and RL controller respectively. It can be observed that $T_{cond,spt}$ is always lower than with the baseline controller, and there is some dependency with the cooling load.

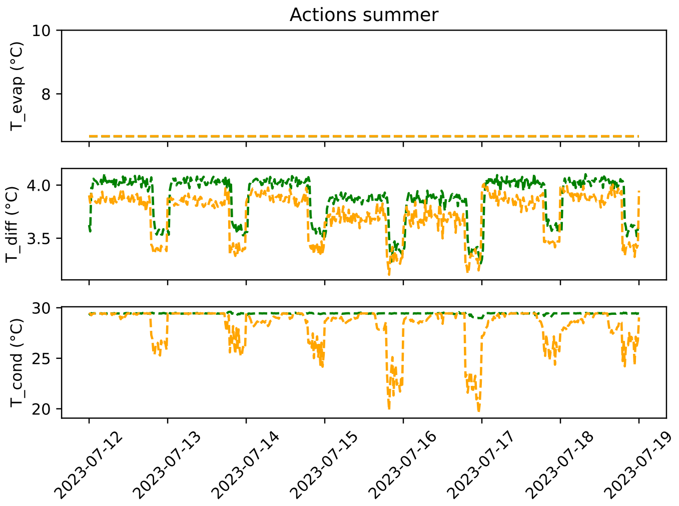


**Figure 8.10** Actions for the continuous model with PPO and variable load pattern, during a summer week. The green and yellow lines represent the actions from the baseline controller and RL controller respectively. It can be observed that $T_{cond,spt}$ is dependent cooling load.

***Realistic model with PPO.*** This section explores the decisions taken by the RL controller with the Realistic scenario and PPO (model 5 in Table 7.2) when evaluating the performance during the proposed weeks.

Figures 8.11 and 8.12 show the RL actions for the winter and summer weeks respectively.

A similar behavior to the continuous model is observed concerning $T_{evap,spt}$ and $T_{diff,spt}$.

The condenser temperature ($T_{cond,spt}$) is set according to the law from the baseline controller. Therefore they have the same value.

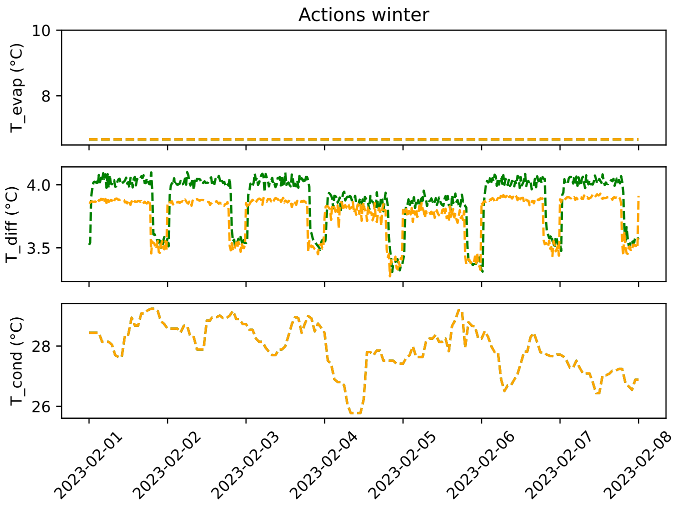


**Figure 8.11** Actions for the realistic model with PPO and variable load pattern, during a winter week. The green and yellow lines represent the actions from the baseline controller and RL controller respectively.

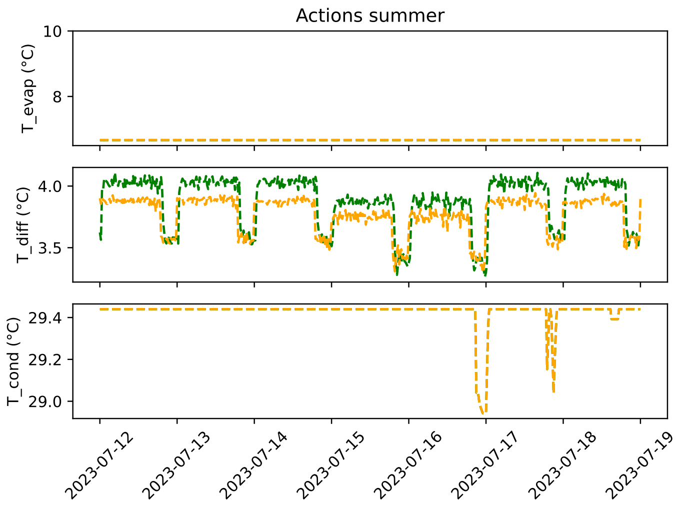


**Figure 8.12** Actions for the realistic model with PPO and variable load pattern, during a summer week. The green and yellow lines represent the actions from the baseline controller and RL controller respectively.

## Analisis of results

The realistic models, which could be trained online safely, show very low power savings and generalize poorly to other conditions. These models take 5 years to train from scratch. However, this is a maximum limit on the actual training time, since the controller can be pre-trained using simulation and then continue training on the real plant. In this way, it could take much less than 5 years of training on the real plant until it reaches a sufficiently good average reward value, depending on how affine the simulation is to the real plant.

Despite this, due to the reduced energy savings observed, it seems much more interesting to implement and test the approaches described in Section 8.1 as future work rather than implementing this controller. Both of those approaches involve training the controller offline. So it could be possible to achieve similar efficiencies to the continuous model with Proximal Policy Optimization (PPO), with around 3% energy savings in comparison to the baseline controller.

The extensive evaluation and analysis conducted revealed several key insights into the energy-saving potential of different models under varying load conditions. Around 3.2 % of energy could be saved on average, and up to 8.4% for lower load conditions.

The main factors that influence energy savings are the load and weather. Under low load conditions, the system is not operating near its maximum capacity. As a result, there is more flexibility for the controller to make energy-efficient decisions without compromising performance. On the other hand, when the load is high, the system operates closer to its maximum capacity, which means that there are less degrees of freedom, since certain control actions are compatible with providing the necessary cooling capacity. The priority under these conditions shifts to maintaining performance and stability.

Similarly, higher outdoor temperatures reduce the optimization possibilities. In this case because the condenser two entry temperature is saturated, so lowering $T_{cond,spt}$, more than the saturation value will not have any effect on the power consumption. Moreover, the efficiency of the chillers decrease as the condenser entry temperatures increases, so the total power consumption is higher under this situation.

The controller trained online demonstrated poor generalization to varying conditions, which means its performance was not consistent across different load scenarios. This inconsistency can lead to poor energy savings, under fluctuating loads. Offline-trained controllers, as suggested for future work, could potentially address this by being trained on a broader range of scenarios, leading to better generalization and consistent performance.

In conclusion, it seems that it is possible to save energy using RL. However, it doesn't seem possible to directly train the controller on the real plant in the same way as it would be done in a simulated environment.

# 9 Conclusions and Future Work

## 9.1 Conclusions

This thesis was focused on the application of Physics-Informed Reinforcement Learning (PIRL) to optimize energy efficiency in cooling systems, specifically within data center environments. It was shown that it is possible to save energy using RL although it is hard to design a controller that learns directly on the real plant, without a model.

As a first step, some literature about stochastic optimization and problem modeling was reviewed, more concretely [Powell, 2019] and [Powell et al., 2022]. This provided a foundation for crafting the problem formulation and getting some inspiration about how to tackle the problem. It was a useful step, that provided insights on how to model problems before thinking of how the solution looks like.

The next step was to carry out experiments to observe the behavior of cooling systems under various operational conditions. These experiments revealed how different parameters, such as setpoints and staging variables affect the power consumption of the plant and the different equipment units. They were especially useful in identifying relevant physical behaviors such as equipment saturation. Furthermore, an idea of how the solution should look like was build up, and compared with the solution that the different RL controllers found.

Several RL models were developed and trained using a simulation model from a data center cooling system in Florida. These models incorporated physics-based information to enhance learning. A test case was defined for evaluating the controllers.

The RL models that were trained offline in a synthetic scenario, where the weather and cooling load conditions can be decided arbitrarily, showed energy savings of around 3.2% in comparison with the baseline controller. However, using a more realistic training scenario and constraining the action space so the actions are safe showed barely around 0.3 % savings.

Finally, the literature was reviewed, seeking approaches to train the controller without a simulation. Two relevant papers were found, described in Section 9.2.

## 9.2 Future Work

This section briefly introduces some fields of study that could be applied to the current problem to train the RL controller effectively using data from the real plant, rather than a simulation.

The approaches described in this section suggest offline training methods based on data from the real plant. These methods aim to achieve better energy savings by addressing the limitations of online training, such as poor generalization and long training times. Implementing these approaches could lead to higher energy savings, potentially matching or exceeding the 3.2% savings observed with the continuous model using PPO.

[Naug et al., 2020] propose an approach where the plant is modeled using Long Short-Term Memory Neural Networks (LSTMNN), instead of a modelica simulation. The aim of this is to be able to train the neural networks automatically with data collected from the real plant. Then the RL controller is trained offline as it was done in this project, with the only difference of using the (LSTMNN) model for getting data instead of a Modelica model.

[Levine et al., 2020] suggests using batch reinforcement learning, to train the controller offline. Off policy algorithms, such as TD3 or DQN support this approach, which consists on learning from a dataset instead of learning by interaction with the real plant. In this way, data, including the power consumption and relevant state variables, could be collected from the real plant, controlled by an arbitrary policy, and then used to train an RL controller.

# Bibliography

AlMahamid, F. and K. Grolinger (2021). “Reinforcement learning algorithms: an overview and classification”. In: *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, pp. 1–7.

Banerjee, C., K. Nguyen, C. Fookes, and M. Raissi (2023). *A survey on physics informed reinforcement learning: review and open problems*. arXiv: `2309.01909 [cs.LG]`.

Bellman, R. (1966). “Dynamic programming”. *science* **153**:3731, pp. 34–37.

Brockman, G., V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba (2016). *Openai gym*. `https://github.com/openai/gym`.

Chen, H.-Y. and C.-C. Chen (2022). “An empirical equation for wet-bulb temperature using air temperature and relative humidity”. *Atmosphere* **13**:11, p. 1765.

Fujimoto, S., H. van Hoof, and D. Meger (2018). *Addressing function approximation error in actor-critic methods*. arXiv: `1802.09477 [cs.AI]`.

International Energy Agency (n.d.). *Buildings*. `https://www.iea.org/energy-system/buildings`. Accessed: 2024-05-29.

Levine, S., A. Kumar, G. Tucker, and J. Fu (2020). *Offline reinforcement learning: tutorial, review, and perspectives on open problems*. arXiv: `2005.01643 [cs.LG]`.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, and A. Teller (n.d.). “Markov decision processes” ().

Moerland, T. M., J. Broekens, A. Plaat, C. M. Jonker, et al. (2023). “Model-based reinforcement learning: a survey”. *Foundations and Trends® in Machine Learning* **16**:1, pp. 1–118.

Naug, A., M. Quinones-Grueiro, and G. Biswas (2020). “A relearning approach to reinforcement learning for control of smart buildings”. *arXiv preprint arXiv:2008.01879*.

Pollard, D. (2000). “Asymptopia: an exposition of statistical asymptotic theory”. *URL http://www. stat. yale. edu/pollard/Books/Asymptopia*.

Powell, W. B. (2019). “A unified framework for stochastic optimization”. *European Journal of Operational Research* **275**:3, pp. 795–821.

Powell, W. B. et al. (2022). *Sequential Decision Analytics and Modeling: Modeling with Python*. now publishers.

Raffin, A., A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann (2021). “Stable-baselines3: reliable reinforcement learning implementations”. *Journal of Machine Learning Research* **22**:268, pp. 1–8. URL: `http://jmlr.org/papers/v22/20-1364.html`.

Schulman, J., S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel (2015). “Trust region policy optimization”. *CoRR* **abs/1502.05477**. arXiv: `1502.05477`. URL: `http://arxiv.org/abs/1502.05477`.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov (2017). “Proximal policy optimization algorithms”. *arXiv preprint arXiv:1707.06347*.

Silver, D., G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller (2014). “Deterministic policy gradient algorithms”. In: Xing, E. P. et al. (Eds.). *Proceedings of the 31st International Conference on Machine Learning*. Vol. 32. Proceedings of Machine Learning Research 1. PMLR, Bejing, China, pp. 387–395. URL: `https://proceedings.mlr.press/v32/silver14.html`.

Sutton, R. S. and A. G. Barto (2018). *Reinforcement Learning: An Introduction*. MIT Press.

Wetter, M., P. Sahlin, and C. van Treeck (2024). *Merkel cooling tower model from the modelica buildings library*. Accessed: 2024-05-30. URL: `https://simulationresearch.lbl.gov/modelica/releases/latest/help/Buildings_Fluid_HeatExchangers_CoolingTowers.html#Buildings.Fluid.HeatExchangers.CoolingTowers.Merkel`.

Zhang, H. and T. Yu (2020). “Alphazero”. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 391–415.

| | |
|---|---|
| **Lund University**<br>**Department of Automatic Control**<br>**Box 118**<br>**SE-221 00 Lund Sweden** | *Document name*<br>MASTER'S THESIS<br>*Date of issue*<br>June 2024<br>*Document Number*<br>TFRT-6246 |
| *Author(s)*<br>Antoni Carrillo Sala | *Supervisor*<br>Johan Åkesson, Carrier Refrigeration Sweden AB<br>Bo Bernhardsson, Dept. of Automatic Control, Lund University, Sweden<br>Anders Rantzer, Dept. of Automatic Control, Lund University, Sweden (examiner) |

*Title and subtitle*

# Physics-Informed Reinforcement Learning Feasibility Study for Building Energy Optimization

*Abstract*

Buildings worldwide account for 30% of energy consumption and Heating, Ventilation and Air Conditioning (HVAC) represent roughly 38% of a building's consumption. Therefore, energy savings are crucial for sustainability. The complexity of buildings, with diverse physical domains and large-scale components, presents challenges to achieving energy-efficient operation. Implementing high-performance controls is effective but takes time and requires qualified experts. Reinforcement learning (RL) offers adaptability but demands extensive data, making it difficult to scale to large systems. RL is extensively used in model-free environments, such as video games; however, when it comes to control the problem is a bit more challenging since it has to achieve stability and robustness of the system. This project explores Physics-Informed RL (PIRL) for building energy optimization, focusing on the supervisory control level. Information from physical models is selected to accelerate learning, and the impact of reinforcement learning on a building's cooling system is studied. Key questions include selecting appropriate information from physical models, determining data requirements, and exploiting the building system architecture for the scalability of PIRL. Dynamic models developed in the Modelica language with an open-source building library are used in the thesis. Numerical experiments are then performed to evaluate the scaling potential of PIRL. One goal is to understand and apply software in the loop methods using the PIRL methodology and Carrier automated logic building control software. It will be shown that physics information helps to reduce training time and that it is possible to save energy using PIRL, in comparison with the baseline controller.

| | | |
|---|---|---|
| *Keywords* | | |
| *Classification system and/or index terms (if any)* | | |
| *Supplementary bibliographical information* | | |
| *ISSN and key title*<br>0280-5316 | | *ISBN* |
| *Language*<br>English | *Number of pages*<br>1-88 | *Recipient's notes* |
| *Security classification* | | |

http://www.control.lth.se/publications/