Scaling Deep Reinforcement Learning for the Optimization of Building Control and Management

Khoder Jneid

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII -Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'informatique de Grenoble

Apprentissage par Renforcement Profond pour l'Optimisation du Contrôle et de la Gestion des Bâtiments
Scaling Deep Reinforcement Learning for the Optimization of Building Control and Management

Présentée par :

Khoder JNEID

Direction de thèse :

Patrick REIGNIER — Directeur de thèse
PROFESSEUR DES UNIVERSITES, GRENOBLE INP - UGA

Stéphane PLOIX — Co-directeur de thèse
PROFESSEUR DES UNIVERSITES, GRENOBLE INP - UGA

Rapporteurs :

PIERRE DE LOOR
PROFESSEUR, ECOLE NATIONALE D'INGENIEURS DE BREST

ROMAIN BOURDAIS
PROFESSEUR, CENTRALESUPELEC RENNES

Thèse soutenue publiquement **le 26 octobre 2023**, devant le jury composé de :

MASSIH-REZA AMINI PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES	Président
PATRICK REIGNIER PROFESSEUR DES UNIVERSITES, UNIVERSITE GRENOBLE ALPES	Directeur de thèse
PIERRE DE LOOR PROFESSEUR, ECOLE NATIONALE D'INGENIEURS DE BREST	Rapporteur
ROMAIN BOURDAIS PROFESSEUR, CENTRALESUPELEC RENNES	Rapporteur
MOHAMMED BENBOUZID PROFESSEUR DES UNIVERSITES, UNIVERSITE DE BREST-BRETAGNE OCCIDENTALE	Examinateur
SIHEM AMER-YAHIA DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES	Examinatrice

Invités :

PAOLO ZANINI
FORMER DATA SCIENTIST, ELICHENS - GRENOBLE

Acknowledgement

I would like to thank first my supervisors Prof. Patrick REIGNIER and Prof. Stephane PLOIX for supporting me during these last four years to come up with this work. I would like to thank also eLichens for offering me the opportunity to pursue this PhD. I had difficult times, especially during the covid-19 pandemic, however, I was able to overcome these times and come back stronger due to the support I received from all my supervisors, my family, and friends.

I would like to offer my special gratitude to my parents who always stand next to me, provide me with continuous support, and motivate me to achieve my goals. Last, but not least, I am extremely grateful to the love of my life Amal for her love and for all the beauty in my life.

Contents

Executive Summary

Heating, ventilation, and air-conditioning (HVAC) systems account for high energy consumption in buildings. Conventional approaches used to control HVAC systems rely on rule-based control (RBC) that consists of predefined rules set by an expert. Model-predictive control (MPC), widely explored in literature, is not adopted in the industry since it is a model-based approach that requires to build models of the building at the first stage to be used in the optimization phase and thus is time-consuming and expensive. During the PhD, we investigate reinforcement learning (RL) to optimize the energy consumption of HVAC systems while maintaining good thermal comfort and good air quality. Specifically, we focus on model-free RL algorithms that learn through interaction with the environment (building including the HVAC) and thus not requiring to have accurate models of the environment. In addition, online approaches are considered. The main challenge of an online model-free RL is the number of days that are necessary for the algorithm to acquire enough data and actions feedback to start acting properly. Hence, the research subject of the PhD is boosting model-free RL algorithms to converge faster to make them applicable in real-world applications, HVAC control. Two approaches have been explored during the PhD to achieve our objective: the first approach combines RBC with value-based RL, and the second approach combines fuzzy rules with policy-based RL. Both approaches aim to boost the convergence of RL by guiding the RL policy but they are completely different. The first approach exploits RBC rules during training while in the second approach, the fuzzy rules are injected directly into the policy. Tests are performed on a simulated office existing in Grenoble INP building during the winter period.

Chapter 1

Introduction

1.1 Context and Motivation

1.1.1 Buildings Energy Consumption

Buildings consume considerable energy in many countries. Both residential and commercial buildings contribute between 20-40% of the total energy consumption in developed countries. In the US for example, they account for nearly 40% of the total energy consumption where half of this energy is linked to heating, ventilation, and air-conditioning (HVAC) systems (Pérez-Lombard et al. [2008]). Hence, buildings' energy management has been a substantial subject in industry and research as well. A building energy management system (BEMS) is responsible for monitoring and controlling all the energy needs of the building. BEMS is defined as a sophisticated electrical control and monitoring system that is capable of controlling the building ranging from HVAC system to lighting, fire alarm system, security, maintenance, and energy management (Manson and Mclntyre [1997]).

HVAC systems are responsible for maintaining good thermal comfort and good indoor air quality. Thermal comfort is usually represented by indoor temperature and humidity, while indoor air quality is represented by CO2 concentration. With high indoor temperature and bad indoor air quality, occupants can feel sleepy, dizzy, and have breathing issues. Thus, maintaining thermal comfort and air quality in the good range is necessary for having high-quality working and living environments.

Nowadays, BEMS' most conventional control strategies of HVAC are rule-based feedback control that relies on some pre-defined schedules to select the setpoints (high-level

				
Equipment type	*Cooling tower, air source, ground source*			*VAV box, Fan coil unit, radiator*
High (ultimate goal)				Room temp sp.
Medium	Cond. water temp Cond. water flow	Supply water temp Supply water flow	Supply air temp. Supply air flow rate	Supply air temp. Supply air flow rate
Low (actuator)	Fan speed Pump speed	Compressor speed etc.	Cooling valve Economizer	Reheat valve Fan/damper

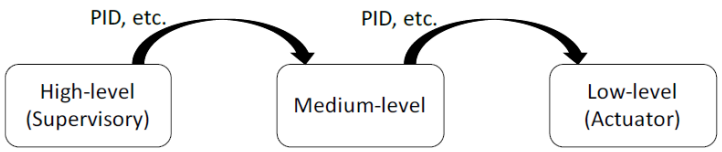


Figure 1.1: Complexity of HVAC control (Wang and Hong [2020]). temp. sp stands for temperature setpoint

control) and then uses techniques such as proportional-integral-derivative (PID, low-level control) to track these setpoints (Geng and Geary [1993]). HVAC systems account for high energy consumption that is significantly affected by their control strategies. Further improvement can be done on HVAC control strategies that can lead to better comfort while minimizing energy consumption.

1.1.2 Control Levels of HVAC Systems

HVAC control is very complex and energy-consuming. Figure 1.1 clarifies why it is difficult to have an HVAC controller that can work for all buildings (Wang and Hong [2020]). Two main reasons make this problem complex. First, an HVAC system is composed of many components: an air handling unit (AHU), a heating/cooling source, a condenser, and a terminal unit. For each component, there are different types such as variable air volume (VAV) and fan coil unit (FCU) for the terminal unit component. Second, there are different levels of controls for each device: high-level (called also supervisory level), medium-level, and low-level.

- High-level control is acting on the setpoints such as room temperature setpoint (heating/cooling temperature setpoints), and carbon dioxide (CO_2) concentration

setpoint in case the HVAC system is equipped with a CO_2 sensor and a controller for CO_2 concentration that can activate the ventilation when CO_2 concentration is high. A setpoint is a target value the system is required to maintain e.g: a room setpoint temperature of 23 °C means the system has to maintain the room at 23 °C. An example of high-level control could be using the observations of outdoor and indoor temperature to determine the room temperature setpoint. Rule-based control (RBC) is the most popular strategy in real HVAC systems and is defined by some simple rules given by experts.

- Low-level control is acting directly on the actuators such as the reheat valve and damper position. Conventional controllers like proportional-integral-derivative (PID) are used to act on the actuators and track the setpoints. An example of low-level control could be controlling the VAV damper to minimize the error between indoor temperature and indoor temperature setpoint. Medium-level control is one level between the high-level and low-level such as supply air temperature and supply air flow rate in VAV.

Most of the studies done on HVAC systems optimization are on controlling the HVAC terminals at a high level. The reason behind selecting to optimize the high-level terminal controls the most is because they are easier to access and safer to control where they exist on a thermostat and can be modified. Moreover, controlling high-level control has more effect on HVAC operations since changing one setpoint can trigger more than one HVAC actuator. Figure 1.2 shows a thermostat that monitors the room temperature and contains 2 setpoints temperature: one for cooling and the other for heating. The thermal comfort range is defined to be between the heating and cooling setpoints temperature. When the room temperature is above/below the cooling/heating setpoint, the HVAC system will cool/heat to bring the room temperature down/up to the comfort range.

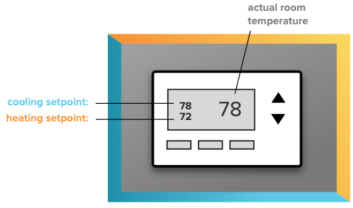


Figure 1.2: A thermostat with dual setpoints temperature.

1.1.3 Motivation

Buildings, both residential and commercial, account for a significant portion of the total energy consumption in developed countries, most of which is consumed by HVAC systems. The energy consumption of HVAC systems is greatly affected by their control strategies. The most popular HVAC control strategy used in real environments is rule-based control which is not energy efficient and can lead to discomfort in terms of temperature and air quality.

This PhD is a CIFRE fellowship within a scientific partnership between eLichens, a French startup based in Grenoble working in the gas sensors and air quality domains, and LIG and G-SCOP laboratories, of the University of Grenoble Alpes. Founded in December 2014, eLichens' mission is to provide relevant and comprehensive information about the air we breathe. The company relies on a portfolio of patents, know-how, and skills that enable a complete air quality solution (sensors and services) and address consumer electronic and industrial markets. eLichens has decided to address the HVAC market with the ambitious goal to optimize energy consumption while assuring optimal comfort to the occupants of residential or commercial buildings. One of the main products developed by eLichens is the indoor air quality station named eLsi. This station measures several environmental variables like temperature, humidity, CO_2 concentration, particulate matter, and volatile organic compound (VOC). The idea is to couple the station measurements with HVAC controls, collect enough informative data, and develop data-driven algorithms to optimize the system.

The following sections will review the existing approaches for HVAC systems control ranging from rule-based control to model predictive control (MPC) to the recently explored approach in reinforcement learning. The advantages and drawbacks of each approach will be presented.

1.2 Existing Approaches

1.2.1 Rule-Based Feedback Control

The most popular approach existing in today's HVAC systems is rule-based feedback control. This approach includes two main things: First, predefined rules (on a high-level control) that define the setpoints are provided according to the experts' experience (Wang and Ma [2008]). Second, the setpoints are maintained using conventional controllers (on the low-level) such as PID control that doesn't require an input-output model (Wang

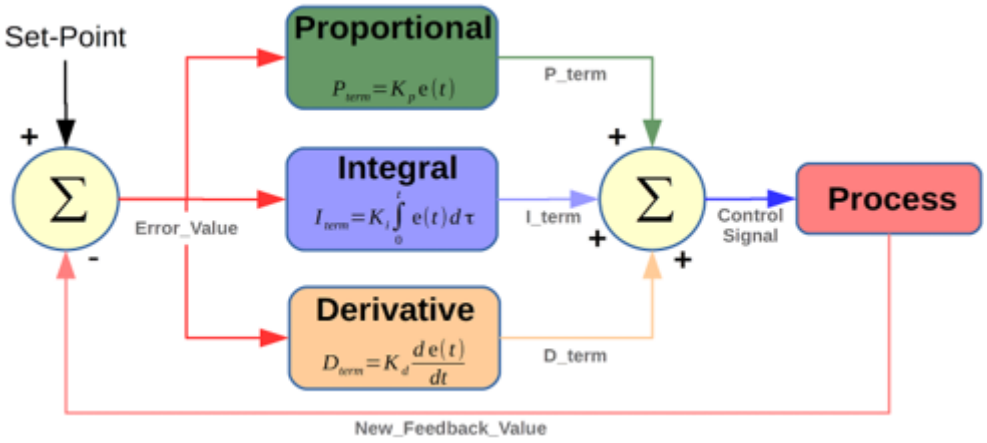


Figure 1.3: PID controller.

et al. [2001]). Figure 1.3 illustrates the main components and the functionality of a PID control. PID includes three terms: *P* for proportional, *I* for integral, and *D* for derivative. *P* accounts for present values of the error e.g if the error value is large and positive, the control output will also be large and positive. *I* accounts for past values of the error e.g if the current control output is not sufficiently strong, the integral of the error will accumulate over time and the controller will respond by applying a stronger action. *D* accounts for possible future trends of the error based on its current rate of change. Rule-based feedback control is a model-free approach that doesn't need system modeling to compute a control policy.

In some situations, it is necessary to pre-heat or pre-cool the environment some time before occupants arrive to reach the comfort level of indoor temperature when people come. And to know the exact time to do that in advance requires some knowledge of the building's thermal inertia and the state of the environment such as outdoor temperature, building temperature, and some forecast values like the outdoor temperature for the next steps. However, rule-based feedback control doesn't have such knowledge and doesn't react to such situations leading to sub-optimal performance. Moreover, this type of approach requires continuous monitoring of the system which leads to high-operational costs.

1.2.2 Model Predictive Control (MPC)

1.2.2.1 MPC Formulation

MPC is a control strategy that relies on an internal model and an optimization technique. It has been used for the first time in the oil industry in 1970 (Morari and Lee [1999]).

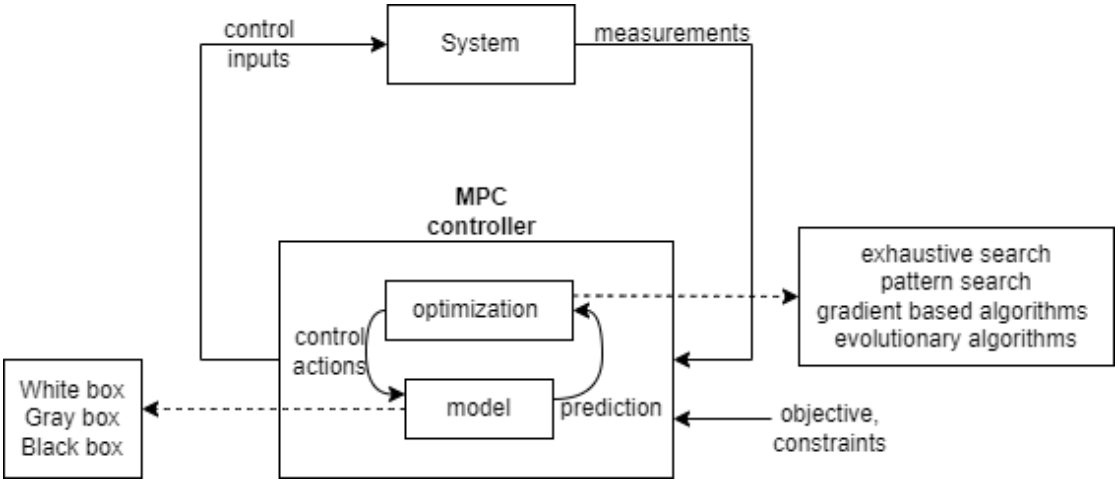


Figure 1.4: Model Predictive Control

The internal model is a predictive model that can predict the state of the system in the next time steps based on some controls and system variables. The optimizer uses the predictive model to anticipate future events over a horizon (specifies how many steps to anticipate ahead), and make better decisions to achieve its task; optimizing energy consumption under some comfort constraints. Figure 1.4 illustrates how MPC operates. At each control time step t, the MPC controller computes the optimal states with the corresponding optimal controls to reach them in the next horizon time steps using the predictive model and the optimizer together and then takes the first control to drive the system towards the objective function.
MPC is an optimization planning problem formulated as follows (Boyd):

$$\min_{\mu} \sum_{k=t}^{t+T} c_k(x_k, \mu_k)$$
$$x_{k+1} = f(x_k, \mu_k), \quad k = t, ..., t+T-1$$
$$\mu_t \in U, \quad t = t, ..., t+T$$
$$x_t \in X, \quad t = t, ..., t+T$$

x represents the state or system variable, μ represents the control variable, c represents the cost function MPC is trying to optimize, f represents the system dynamics (the internal or predictive model), and T represents the prediction horizon. X and U represent the input constraint sets. At time step t, the system is at state x_t and the MPC controller is trying to find the optimal control trajectory over one horizon that leads to such output control trajectory of length T $(\mu_t, x_{t+1}, ..., x_{t+T}, \mu_{t+T})$ but only the first control will be executed. The optimization problem will move then one step and the same procedure will be repeated for the next time step.

1.2.2.2 Simulation Models and Optimization Techniques in Buildings

The first phase of MPC for buildings is to develop the simulation models of the buildings' dynamics. Simulation models as shown in Figure 1.4 can be of different types: white-box, black-box, and gray-box (Wang and Ma [2008]).

- White-box, called also physical-based models, rely on physical models of the buildings' dynamics to predict the state and energy consumption of the system under study. White-box models use a priori knowledge of the system and detailed physical models such as laws of energy, mass, heat transfer, flow balance, etc.

- Black-box models are data-driven models that don't require prior knowledge of the buildings' dynamics. These models use data recorded in the building automation system (BAS) related to the system variables, control variables, and disturbance variables (weather forecast, occupant existence) and can predict the state of the system and the energy cost. Such models can be an artificial neural network (ANN) for instance.

- Gray-box models are hybrid models that rely on both: some physical equations and data of the building to calibrate and tune the parameters of the models.

The second phase of MPC is the optimization controller. There are many optimization techniques used in this phase such as exhaustive search, pattern search, gradient descent algorithms, and evolutionary algorithms (Nguyen et al. [2014]). The evolutionary or genetic algorithm will be revisited in section 3.5 in one experiment of MPC.

1.2.2.3 Applications of MPC

MPC is the most explored approach in literature. It has shown some success in buildings. For instance, in Aswani et al. [2011], authors tested MPC on a simulated air conditioner (AC) and succeeded to reduce the electricity consumption by taking into account the occupancy. Privara et al. [2011] used MPC to control the heating system of a university building and succeeded to achieve 17-24 % of energy savings compared to the existing controller. Ma et al. [2009] used MPC to optimize the thermal energy storage in building systems and achieved 24.5% of energy savings in simulations. As a result, MPC has shown some success in both real experiments and simulations.

1.2.2.4 Drawbacks of MPC

Although MPC has been widely explored in literature and has shown successful applications, it hasn't been adopted in the industry. The main complexity in MPC lies in the first phase which requires developing predictive models of the buildings. This phase accounts for 75 % of the time required for MPC implementation according to Rockett and Hathway [2017] and requires high expertise and a long time for calibration. For MPC to achieve good results, it requires accurate predictive models (Killian and Kozek [2016], Privara et al. [2013]). The difficulty in developing the predictive models goes back to the complexity of buildings' dynamics. Moreover, buildings are very heterogeneous, and thus models can't generalize between buildings (Lü et al. [2015]), specifically the white-box and gray-box models. Concerning the optimization phase, MPC is a non-convex optimization problem because of the non-linearities in the buildings' dynamics, and thus optimization can take a considerable time especially when the control variables are large and the control time step is short (Raman et al. [2020]). Many applications of MPC tried to simplify the first phase of MPC by linearizing the predictive models (Atam and Helsen [2015]). However, this strategy will make problems when it comes to more complex systems such as HVAC systems that are not linear and will not eliminate the problem of high expertise and a great amount of time required to have good models that can't be transferred to other buildings. Hence, it is important to develop or rely on other approaches with the following features: can generalize to different buildings and don't require to develop predictive models of the building's system.

1.2.3 Reinforcement Learning

1.2.3.1 Definition

Reinforcement Learning (RL) is a subcategory of machine learning that involves learning by interaction (Sutton and Barto [2018]). Two main players exist in this kind of learning: the agent and the environment. The agent is the learner and decision-maker at the same time. The environment is the system the agent interacts with. It generates a reward that represents evaluative feedback for the agent. A reward is a scalar value that represents the quality of the action taken by the agent at a specific state; such as +1 for rewarding a good action and -1 for penalizing a bad action.
The agent learns through trial and error and then a great number of interactions with the environment is needed to learn the optimal actions. The agent-environment interaction is shown in Figure 1.5. At each time step t, the agent observes the state S_t that includes a

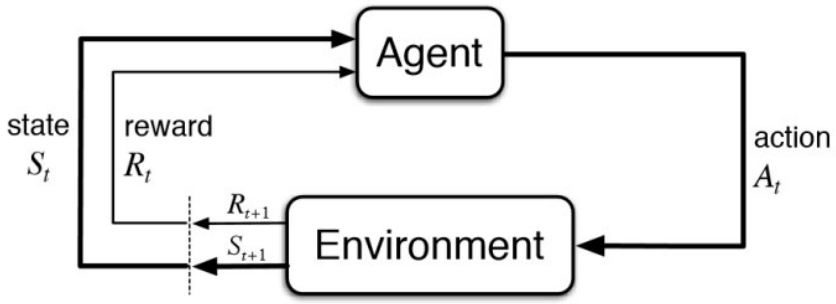


Figure 1.5: Reinforcement Learning agent-environment interaction.

reward R_t then takes an action a_t. The environment will move then to a new state S_{t+1}, and the agent will receive evaluative feedback of the action taken, reward R_{t+1}. The agent's goal is to learn an optimal policy π^* that maximizes not the immediate reward but the cumulative reward it receives in the long run called also the expected return G_t. The optimal policy specifies the best action to take in a specific state. The expected return G_t, in its simplest form, is defined as follows:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{1.1}$$

1.2.3.2 Model-Based vs Model-Free RL

RL algorithms are divided into two subcategories: model-based and model-free algorithms.

- In model-based RL algorithms, the agent learns a model of the environment by observing how the state is changed when an action is taken and thus learning the transition function. When a model of the environment is learned, it could be coupled then with a planning algorithm to obtain the optimal policy. The learned model is used to simulate transitions, which increases sample efficiency.

- Model-free RL algorithms do not need a model of the environment and they learn a policy through trial and error. They require huge data to learn a good policy. Model-free RL algorithms are divided into two classes: online and batch. Online RL algorithms update the value being in a state in an online fashion after each interaction based on the current measurement. Such an online model (called on-policy as well) is SARSA (Rummery and Niranjan [1994]). Batch RL algorithms store past transitions to be reused for the update of the value being in a state.

Transitions can be fixed such as in fitted Q-iteration (FQI) (Ernst et al. [2005]) or growing by collecting new transitions from the environment enabled by the experience replay strategy Lin [1992]. Experience replay can make learning more stable and faster than online RL algorithms.

1.2.3.3 Applications in Buildings

Reinforcement learning has shown great success in gaming and robotics. AlphaGo which is developed based on reinforcement learning has defeated the human European Go champion(Brunner [2019]). RL has been recently explored in literature in the context of buildings' control:

- In Wei et al. [2017], authors use deep-Q networks (DQN Mnih et al. [2015]) to control the fan air flow rate in a variable air volume (VAV) system to maintain thermal comfort and reduce energy consumption. The building with the HVAC system is simulated in EnergyPlus (energy simulation software: Crawley et al. [2000]). Authors prove that DQN achieves high energy savings in simulations compared with the traditional approach used in EnergyPlus.

- Zhang et al. [2018] uses Asynchronous Advantage Actor Critic (A3C Mnih et al. [2016]) to control the heating system supply water temperature of a radiant heating system to maintain thermal comfort in an intelligent workplace (IW) and reduce the energy consumption. The authors use EnergyPlus to model the building with the HVAC system. The development of the RL controller for the radiant heating system undertakes two phases. The first phase consists of training the A3C offline using the building energy model application (EnergyPlus). The second phase consists of deploying the trained DRL (deep reinforcement learning) agent offline in the actual HVAC system. In the second phase, the agent will continue to refine its control policy from the actual feedback from the real HVAC system. As a result, this work has achieved 15% of energy savings in a real experiment.

- Gao et al. [2019] uses Deep Deterministic Policy Gradient (DDPG: Lillicrap et al. [2015]) to control heating and humidity setpoints of an HVAC system; modeled in transient system simulation tool (TRNSYS). DDPG was compared with other DRL methods such as DQN, SARSA, Q-learning. Simulation results show that DDPG outperforms the other methods and improves occupants' thermal comfort while reducing the energy consumption of HVAC system.

Paper	Simulation tool	HVAC type	RL method	Convergence time
Wei et al. [2017]	EnergyPlus	VAV	DQN	100 months
Zhang et al. [2018]	EnergyPlus	Radiant heating system	A3C	$\sim 62 years$
Gao et al. [2019]	TRNSYS	HVAC not specified	DDPG	10k hours
Zhang et al. [2019]	EnergyPlus	VAV	model-based RL	7 days
Moriyama et al. [2018]	EnergyPlus	VAV	TRPO	300 years
Dalamagkidis et al. [2007]	Matlab/Simulink	Heat Pump	LRLC	4 years
Barrett and Linder [2015]	physical equations	not clear	Q-learning	not clear

Table 1.1: Reinforcement learning applications in buildings. The last column (convergence time) indicates how long it would take if the system should learn directly on a real building instead of a simulation environment.

- Zhang et al. [2019] uses model-based RL to control the air flow rate and temperature setpoint of a VAV system in a two zones data center modeled in EnergyPlus. The dynamics of the system are approximated using a neural network. Using model-based RL, the energy consumption is reduced by 17.1-21.8 % compared with the model-free approach, proximal policy optimization (PPO: Schulman et al. [2017]). Model-based RL shows as well faster convergence by 10x. However, the comparison made by the author doesn't seem fair. In more detail, in the article, model-based RL achieves almost the maximum reward after 7 days but after learning the dynamics of the buildings using historical data of 65 days. While model-free RL, PPO, succeeds to achieve almost the same reward after 70 days but starting completely without any knowledge. So overall, model-based and model-free-RL have the same result in the article.

- Moriyama et al. [2018] uses DRL to control the HVAC system in a two zones data center. Fan air flow rate and temperature setpoint are controlled using trust region policy optimization (TRPO: Schulman et al. [2015]) to reduce the cooling energy consumption and maintain thermal comfort. TRPO succeeds to reduce the cooling energy consumption by 22%.

1.2.3.4 Limitations of RL Applications in Buildings

RL for the control of HVAC systems has been recently explored in literature. It is still in the research work and has not been adopted in the industry. One main advantage of model-free RL is that it doesn't need prior knowledge of the buildings' dynamics and can learn a policy only by interacting with the environment. Hence the complexity of buildings' dynamics is not anymore a problem with model-free RL. However, this type

of machine learning requires huge data to learn a good policy since it learns through trial and error and requires exploring the state-action space to find the optimal actions leading to a lot of transitions with bad actions. For this reason, most of the RL work for HVAC done uses simulation tools like EnergyPlus, TRNSYS, or physical equations (Table 1.1) to model the environment that is considered as the real system and then RL agent interacts with the simulated environment to learn a good policy without relying on anything else other than the interactions. After obtaining a good policy using the simulation tool, RL agent can be deployed and tuned by itself in the real environment. Thus, the exploration is mostly done on the simulation tool and then eliminating to take bad actions in the real system.

However, the solution or the policy obtained on one building doesn't work on another building since their dynamics are different. This type of approach requires simulating every building inside a building simulation tool to use as an auxiliary to apply reinforcement learning. The problem that appears here is the great amount of time and expertise required to simulate a building with its equipment (HVAC, lighting, ...) and in the end, this is required for every building. What appears to be a solution for the problem of HVAC complexity is not a complete solution since it doesn't generalize to other buildings.

Table 1.1 summarizes most of the work done in the domain of RL for HVAC optimization. Most of the work done relies on RL algorithms that are considered model-free and on a physical simulation tool that allows to represent a copy of the real environment. As shown in the table, these approaches are taking unreasonable time to converge and thus cannot present a practical solution in real systems. These approaches present a solution only when the RL algorithm is pretrained in advance in a simulator that models the real system and then deployed later online. Thus, they require the modeling of every system inside the simulator which needs time and expertise. As a result, these approaches don't represent a practical and complete model-free approach as presented in the literature. In addition, the data used during the training should be accessible and easy to obtain which is not the case in some of the literature where some unreachable information are used in the state of the system such as in Gao et al. [2019] where metabolic rate and clothing insulation are required to be presented.

Although many studies are showing the importance RL in buildings, few of them have tested their work in a real experiment. Out of the seven papers presented in Table 1.1, Zhang et al. [2019] is the only one who deployed his work in a real experiment achieving energy savings by 17.1-21.8% and fast convergence time by 7 days. However, he relies

on model-based RL that requires at the first stage to learn the dynamics of the building based on historical data of 65 days. The hindrance in his work is the first stage which has high costs in terms of time and expertise.

1.3 Research Question

The research question targeted during this PhD is as follows:

- How can we develop an applicable control approach that can optimize the energy consumption of HVAC systems while maintaining good thermal comfort and good indoor air quality? The control approach should be generic (can be applied to any building) and converge in a reasonable and short time.

1.4 Main contributions

As discussed in the previous sections, MPC represents a weak solution for the optimization of HVAC systems since it depends on very accurate models that require considerable time and expertise to obtain. Moreover, it is a non-generic approach since it cannot be generalized to another building because of the heterogeneity of buildings. However, MPC can be built in a completely data-driven approach which can then be applied to another building. RBC doesn't require accurate models and consists of simple rules. However, it is not optimal and requires constant monitoring and tuning.
On the other hand, RL appears to be better than the previous two control approaches since it can be model-free where no prior knowledge of the building is needed (if we do not pre-train the controller on a building simulator) and it is not required to develop predictive models. Yet, the main problem of model-free RL is the need for huge data to reach a good control policy, something that is not applicable in real-world HVAC systems. Hence, our contribution in this PhD is mainly to drive faster the convergence of RL approaches so that they are applicable in real-world systems. At the beginning of the PhD, the privilege was to develop a data-driven framework that doesn't require the knowledge of an expert or the environment dynamics. Hence, an advanced MPC with data-driven architecture has been considered at the beginning. Then, the option of model-free RL has been preferred later. Thus, the contributions are as follows:

- Develop a data-driven MPC to optimize the air flow rate of the HVAC system.

- Combine RL with RBC to guide the RL policy at the beginning and drive faster the convergence of RL.
- Inject human knowledge into a comprehensive policy to explain better the results and have fast convergence.

1.5 Outline

The dissertation is structured into chapters as follows:

- Chapter 2 - Reinforcement Learning Fundamentals: This chapter will mainly provide state-of-the-art of reinforcement learning. The theory of RL will be detailed with the type of RL algorithms.
- Chapter 3: This chapter tackles the optimization of an HVAC system simulated in EnergyPlus using an advanced architecture of model predictive control. It illustrates the limitations of this optimization strategy and introduces a way to explore a more advanced strategy: Reinforcement learning.
- Chapter 4: Our contribution is built on top of an existing DRL algorithm: DQN. It aims at guiding the RL agent at the beginning of the learning phase and thus speeding its convergence. It is called: DQN+RBC since it is a combination of rule-based control (RBC) and DQN.
- Chapter 5: Fuzzy logic is introduced to inject human knowledge into the control policy. ANFIS, a neuro-fuzzy approach that combines the power of fuzzy inference and neural networks is used as an initial control policy. Then, a policy-based RL algorithm, REINFORCE, is used to optimize the policy.
- Chapter 6: This chapter concludes the PhD work and suggests many subjects worth exploring in future research.

Chapter 2

Reinforcement Learning Fundamentals

This chapter presents the theory behind the reinforcement learning problem. It starts by describing first the reinforcement learning framework. Second, it shows how to use dynamic programming to find the optimal policy given that the transition function is provided. Third, it focuses on model-free reinforcement learning, which searches for the optimal policy without the need to have the transition function, by introducing temporal difference learning and then presenting the three classes of model-free reinforcement learning: value-based, policy-based, and actor-critic.

2.1 Reinforcement Learning Framework

2.1.1 Markov Decision Process

A Markov decision process (MDP) is a sequential decision making where actions influence not only immediate rewards but also future states and rewards. Thus, MDP involves delayed reward and the need to tradeoff immediate and delayed reward (Sutton and Barto [2018]). The RL framework is described in section 1.2.3.1. This framework can be mathematically formulated as an MDP with the following specifications ($\mathcal{S}$, $\mathcal{A}$, p, $\mathfrak{R}$, γ):

- State space $\mathcal{S}$ is a set of all possible states.
- Action space $\mathcal{A}$ is a set of all possible actions.

- Transition function p(s'|s, a) defines the dynamics of the MDP and governs the transition between states:

- R: $\mathcal{S}\,x\,\mathcal{A} \rightarrow \mathbb{R}$ is the reward function computing the expected reward achieved on a transition starting in (s, a):

$$R(s,a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r\in\Re} r \sum_{s'} p(r,s'|s,a) \tag{2.1}$$

  where $\Re$ is the set of all possible rewards.

- $\gamma \in [0, 1]$ is a discount factor that weights the importance of later rewards to earlier ones.

The transition between states should meet the Markov property where the next state depends only on the previous state and action and is conditionally independent of all the previous states and actions.

$$P(s_{t+1}|s_t, a_t) = P(s_{t+1}|s_t, a_t, ..., s_0, a_0) \tag{2.2}$$

2.1.2 Cumulative Reward

The goal of RL is to learn a control policy that maximizes the discounted cumulative reward at each control timestep. The control policy represents the agent's behavior and can be either deterministic or stochastic. A deterministic control policy, $\pi(s) = a$, maps each state to a specific action. While a stochastic policy, $\pi(a|s) : \mathcal{S}\,x\,\mathcal{A} \rightarrow [0, 1]$, assigns a probability distribution over the actions in a given state.
Considering that the agent-environment breaks into episodes where there is a terminal state, the discounted cumulative reward G_t, called long-term return, is represented as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1} R_T = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{2.3}$$

γ is the discount factor and specifies how much importance should be given to future rewards vs. immediate reward. G_t can be defined recursively as follows:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ...) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{2.4}$$

2.1.3 Value Functions

The state value function v(s) estimates how good it is for the agent to be in state s. To be more precise, it computes the cumulative reward following the policy π. The state value function $v_\pi(s)$ is defined by:

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s, \pi] \tag{2.5}$$

The state-action value function defines how good for the agent to take an action a in a state s following a policy π:

$$q_\pi(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a, \pi] \tag{2.6}$$

The state value function can be defined in function of the state-action value function as follows:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a) = \mathbb{E}[q_\pi(S_t, A_t)|S_t = s, \pi] \tag{2.7}$$

2.1.4 Optimal Value Functions

The optimal state value function $v^*(s)$ is the maximum value function over all policies and it specifies the best possible performance in the MDP:

$$v^*(s) = \max_\pi v_\pi(s) \tag{2.8}$$

The optimal state-action value function $q^*(s,a)$ is the maximum action-value function over all policies:

$$q^*(s,a) = \max_\pi q_\pi(s,a) \tag{2.9}$$

2.1.5 Finding an Optimal Policy

An optimal policy can be found by maximizing over $q^*(s,a)$:

$$\pi^*(s,a) = \begin{cases} 1 & \text{if a} = \arg\max_{a \in A} q^*(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

2.1.6 Bellman Equations

The state value function v(s) can be defined recursively, known as the Bellman expectation equation of state value function:

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}[G_t|S_t = s, \pi] && (2.11)\\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s, \pi] && (2.12)\\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t \sim \pi(S_t)] && (2.13)\\
&= \sum_a \pi(a|s) \sum_r \sum_{s'} p(r, s'|s, a)(r + \gamma v_\pi(s')) && (2.14)\\
&= \sum_a \pi(a|s) \Big[R(s,a) + \gamma \sum_{s'} p(s'|s,a) v_\pi(s')\Big] && (2.15)
\end{aligned}
$$

The state-action value function $q_\pi(s,a)$ can be defined recursively, known as the Bellman equation of state-action value function:

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] && (2.16)\\
&= \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a] && (2.17)\\
&= \sum_r \sum_{s'} p(r, s'|s, a) \Big(r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')\Big) && (2.18)\\
&= R(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s') q_\pi(s', a') && (2.19)
\end{aligned}
$$

2.1.7 Bellman Optimality Equations

Given an MDP = <$\mathcal{S}$, $\mathcal{A}$, p, r γ>, the Bellman optimality equations of value functions obey the following expectation equations:

$$
\begin{aligned}
v^*(s) &= \max_{a \in \mathcal{A}} q_{\pi_*}(s,a) = \max_a \Big[R(s,a) + \gamma \sum_{s'} p(s'|s,a) v^*(s')\Big] && (2.20)\\
q^*(s,a) &= R(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a' \in \mathcal{A}} q^*(s', a') && (2.21)
\end{aligned}
$$

The Bellman optimality equation of v, called optimal state value function v^*, specifies the largest expected return achieved in state s under the optimal policy π^*. While the bellman optimality equation of q, called optimal state-action value function q^*, specifies the largest expected return in state-action pair (s, a) under the optimal policy π^*.

2.2 Dynamic Programming

If the transition function is provided, the Bellman optimality equation can be solved using dynamic programming. Dynamic programming refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process (MDP) Sutton and Barto [2018]. Dynamic programming methods can be divided into policy iteration and value iteration.

2.2.1 Policy Iteration

The policy iteration method consists of two main steps: policy evaluation and policy improvement.

2.2.1.1 Policy Evaluation

The first step in policy iteration is policy evaluation. This step consists of estimating the value of a policy:

$$v_\pi(s) = \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1})|s,\pi]$$

The idea of computing this equation is by starting with an initial guess of v, turning the equation into an update rule, and then iterating until convergence:

Algorithm 1 Policy Evaluation

First, initialize v_0, e.g., to zero

Then iterate

$$\forall s: \; v_{k+1}(s) = \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|s,\pi]$$

Stopping: whenever $v_{k+1}(s) = v_k(s)$, for all s, we must have found v_π

2.2.1.2 Policy Improvement

The second step in policy iteration is policy improvement. This step consists of improving the current policy by considering a new greedy policy with respect to the value of the

current policy as follows:

$$\pi_{new}(s) = \arg\max_a q_\pi(s,a) \tag{2.22}$$

$$= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \tag{2.23}$$

Then we can repeat the process of evaluating and improving of the new policy until obtaining the optimal policy as follows; E for evaluate and I for improve:

$$\pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} v_{\pi_2} \xrightarrow{I} \pi_3 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} v_{\pi_*} \xrightarrow{I} \pi_*$$

2.2.2 Value Iteration

The main drawback in policy iteration is the policy evaluation step which is time-consuming. The idea of value iteration is to truncate the policy evaluation step by updating the policy at every iteration. The process is done by turning the bellman optimality equation into an update as follows:

$$v_{k+1}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a]$$

The output is policy $\pi \approx \pi^*$ such that:

$$\pi(s) = \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

2.3 Temporal Difference Learning

In most real-world problems, the transition function $p(s'|s,a)$ is not provided and requires a complex task to obtain. Hence, temporal difference learning (TD) consists of solving the MDP problem without having the need to know the model of the environment and the associated reward of every state-action pair. For this reason, it is called model-free reinforcement learning.

2.3.1 SARSA: On-policy TD learning

The idea of TD learning is to estimate the expected return using samples $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ where the state-action values are updated towards the estimated return $R_{t+1} + \gamma q(S_{t+1}, A_{t+1})$:

$$q_{t+1}(S_t,A_t) = q_t(S_t,A_t) + \alpha\Big(\overbrace{\underbrace{R_{t+1} + \gamma q_t(S_{t+1},A_{t+1})}_{target} - q_t(S_t,A_t)}^{\text{TD error}}\Big) \tag{2.24}$$

α represents the learning rate or step size and *gamma* is the discount factor. This algorithm is known as SARSA because it uses samples $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. SARSA is an on-policy algorithm because the same policy used to make decisions is the one to be evaluated and improved. To elaborate, the policy used to generate actions is called behavior policy and is referred to by $\mu(s)$. The target policy is the policy that an agent is trying to learn and is referred to by $\pi(s)$. In the equation above, in the target, action A_{t+1} is selected according to the same policy used to select actions. Thus, SARSA is an on-policy algorithm since the target policy is the same as the behavior policy.

2.3.2 Q-learning: off-policy TD learning

The state-action values in Q-learning are updated as follows:

$$q_{t+1}(S_t, A_t) = q_t(S_t, A_t) + \alpha\Big(\overbrace{\underbrace{R_{t+1} + \gamma \max_{a'} q_t(S_{t+1}, a')}_{target} - q_t(S_t, A_t)}^{\text{TD error}}\Big) \tag{2.25}$$

$$= q_t(S_t, A_t) + \alpha\Big(R_{t+1} + \gamma q_t(S_{t+1}, \arg\max_{a'} q(S_{t+1}, a')) - q_t(S_t, A_t)\Big) \tag{2.26}$$

A table of size $\mathcal{S}$ x $\mathcal{A}$ is used to store the state-action values in both SARSA and Q-learning. Q-learning is an off-policy algorithm where it attempts to evaluate or improve a policy different from that used to generate the data Sutton and Barto [2018]. To elaborate, the behavior policy used to select actions, ε-greedy, is different from the target policy that the agent is trying to improve. The target policy here is greedy w.r.t q(S, A):

$$\pi(S_{t+1}) = \arg\max_{a'} q(S_{t+1}, a')$$

2.3.3 Exploitation-Exploration strategy

During the training phase, the RL agent attempts to learn an estimate of the optimal value function (or policy). Given that the agent has no knowledge at the beginning, it will not behave optimally since it doesn't know the optimal value function or policy. Hence, the agent has to explore new actions for the sake to find a better policy and to exploit by selecting the action that is most likely to be optimal according to the agent's past knowledge. This scenario is faced by the RL agent during training and is called the exploration-exploitation dilemma. SARSA and Q-learning use ε-greedy policy to deal with this dilemma. ε-greedy policy is one of the most common exploration methods in RL where ε controls the trade-off between exploitation and exploration. ε-greedy can be

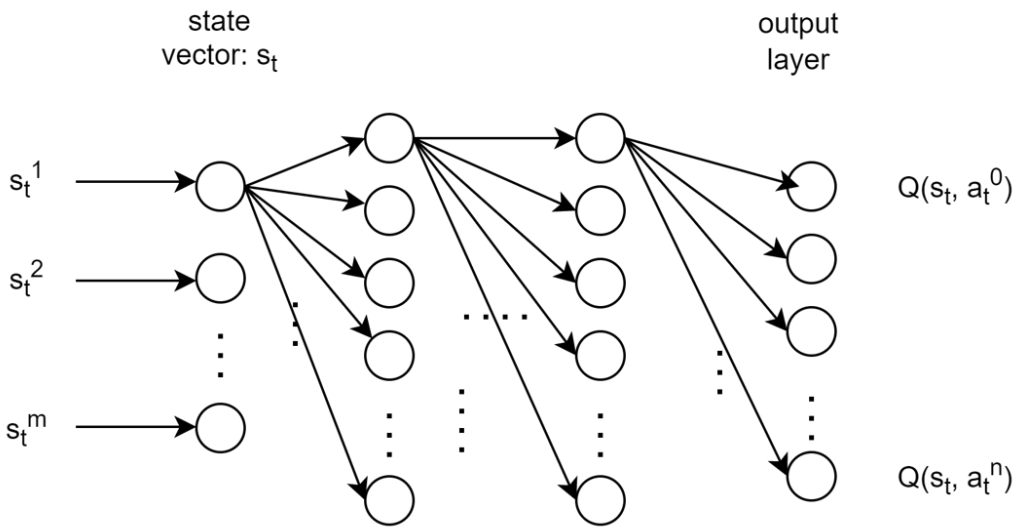


Figure 2.1: A neural network model is used to approximate the state-action values Q(s, a).

described as follows:

$$A_t = \begin{cases} \arg\max_{a \in \mathcal{A}} q(S_t, A_t) & \text{with probability } \varepsilon \\ \text{uniform random action} & \text{with probability 1 - } \varepsilon \end{cases} \tag{2.27}$$

Another way to explore in continuous action space is to add exploration noise ε (e.g. Gaussian noise) during action selection such as in deep deterministic policy gradient DDPG (Lillicrap et al. [2015]):

$$A_t = \mu(s|\theta) + \varepsilon$$

μ is the behavior policy that decides what action to follow in a specific state.

2.4 Function Approximators

SARSA and Q-learning use a lookup table to store the state-action values Q(s, a). However, for real-world problems, the state and action space can become high-dimensional, especially when they are continuous, and storing the data in a table becomes no longer practical and possible. To cope with this drawback, a function approximator can be used to estimate the value functions and even the policy which can be done using a parametric method like neural network Figure 2.1. Reinforcement learning is called deep reinforcement learning DRL when combined with neural networks. Hence, parametrized

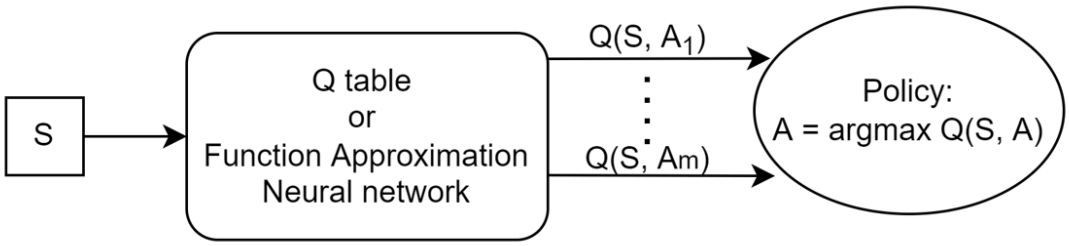


Figure 2.2: Value-based reinforcement learning.

functions are used to represent value function $v_\pi(s)$, state-action value function $q_\pi(s,a)$, and the control policy $\pi(s)$:

$$v_\pi(s) = v(s;\theta_v) \tag{2.28}$$

$$q_\pi(s,a) = q(s,a;\theta_q) \tag{2.29}$$

$$\pi(s) = \pi(s;\theta_\pi) \tag{2.30}$$

2.5 Model-Free Reinforcement Learning

Model-free RL is used when a model of the environment is not existing and the learning depends only on the interaction between the agent and the environment. Model-free RL consists of three main classes: value-based, policy-based, and actor-critic.

2.5.1 Value-based

In value-based RL, the policy to maximize the expected reward is not learned explicitly. An optimal value function or state-action value function is approximated first and then the policy is deducted in a greedy way Figure 2.2 . So far, what has been discussed before is value-based RL algorithms such: as Q-learning and SARSA which aim at approximating the optimal value function, and then the policy is deducted from the value function. Thus, the policy depends completely on the value function.
Improvements have been done to Q-learning to deal with high-dimensional state space where a function approximator is used to estimate the value function such as a neural network. Such a combination introduces deep reinforcement learning (DRL). Deep-Q network (DQN) Mnih et al. [2013] is a value-based DRL and represents a valuable improvement of Q-learning that introduces additional features like target network and experience replay to stabilize learning and break correlation between samples. DQN will

be discussed in detail in another chapter.

2.5.2 Policy-based

So far, all the methods presented depend on value functions (state of state-action) where the policy is generated at the end from the value function. However, policy gradient methods Sutton et al. [1999] attempt to model and optimize the policy directly. The policy is modeled with a parametrized function w.r.t θ, $\pi_\theta(a|s)$. RL attempts to maximize the cumulative reward G_t which can be defined according to the policy. Thus, policy gradient attempts to optimize θ to obtain the best reward.
For simplicity and to make proofs easy to understand, a one-step episodic problem (contextual bandit) will be considered to explain and prove the idea of policy gradient. The expected reward is then computed as follows:

$$J(\theta) = \mathop{\mathbb{E}}_{\pi_\theta}[R(S,A)]$$

R_{t+1} is a scalar value and doesn't depend on θ and thus it cannot be used to sample the gradient. Instead, policy gradient uses the following identity to approximate the gradient and then maximize $J(\theta)$:

$$\nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[R(S,A)] = \mathop{\mathbb{E}}_{\pi_\theta}[R(S,A)\nabla_\theta \log \pi_\theta(A|S)] \tag{2.31}$$

Sutton and Barto [2018] derives the gradient as:

$$\begin{aligned}
\nabla_\theta \mathop{\mathbb{E}}_{\pi_\theta}[R(S,A)] &= \nabla_\theta \sum_s d(s) \sum_a \pi_\theta(a|s) r_{sa} \qquad\qquad r_{sa} = \mathbb{E}[R(S,A)|S=s, A=a] \\
&= \sum_s d(s) \sum_a r_{sa} \nabla_\theta \pi_\theta(a|s) \\
&= \sum_s d(s) \sum_a r_{sa} \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \\
&= \sum_s d(s) \sum_a \pi_\theta(a|s) r_{sa} \nabla_\theta \log \pi_\theta(a|s) \\
&= \mathop{\mathbb{E}}_{d,\pi_\theta}[R(S,A)\nabla_\theta \log \pi_\theta(A|S)]
\end{aligned}$$

d(s) defines the probability of being in state s in the long run. The right-hand side of the equation is an expected gradient that can be sampled. The policy-gradient update is as follows:

$$\theta_{t+1} = \theta_t + \alpha R_{t+1} \nabla_\theta \log \pi_{\theta_t}(A_t|S_t)$$

The update doesn't use any value function and depends on the parametrized policy. This algorithm is known as Reinforce. Reinforce can be generalized to include a comparison of the expected return to a baseline value and thus called Reinforce with baseline. The goal of this generalization is to reduce variance and speed learning, done as follows:

$$\theta_{t+1} = \theta_t + \alpha(R_{t+1} - v_w(S_t))\nabla_\theta \log \pi_{\theta_t}(A_t|S_t) \tag{2.32}$$

$$w_{t+1} = w_t + \beta(R_{t+1} - v_w(S_t))\nabla_w v_w(S_t) \tag{2.33}$$

α and represent the learning rates for the policy and value updates respectively. Although a state value function is used here, reinforce with baseline is still called a policy-gradient algorithm according to Sutton and Barto [2018] since the baseline was used only as an estimate and not as a critic: The state-value function is not used for bootstrapping where the value estimate of a state is updated according to estimated values of subsequent states.

As stated at the beginning of this section, a one-step episodic problem is considered to simplify the proofs. However, an episode consists of many steps and to adapt to a multi-step problem, R_{t+1} should be replaced by G_t which is the cumulative return that could be obtained after finishing the episode. Thus, the updates are as follows:

$$G_t = \sum_{k=t+1}^{T} R_k \tag{2.34}$$

$$\theta_{t+1} = \theta_t + \alpha(G_t - v_w(S_t))\nabla_\theta \log \pi_{\theta_t}(A_t|S_t) \tag{2.35}$$

$$w_{t+1} = w_t + \beta(G_t - v_w(S_t))\nabla_w v_w(S_t) \tag{2.36}$$

2.5.3 Actor critic

The actor-critic method is a combination of value-based and policy-based methods. Actor critic RL algorithms use a critic to estimate a value or state-action value function and an actor to update the policy in the direction suggested by the critic. The cumulative reward G is estimated using the state-value function:

$$G_t = R_{t+1} + \gamma v_w(S_{t+1}) \tag{2.37}$$

The actor-critic algorithm is then as follows:

Algorithm 2 Actor critic

Input: a differentiable policy parametrization $pi_\theta(a|s)$
a differentiable state-value function parametrization $v_w(s)$
Parameters: step sizes α, β
Initialize policy parameter θ and state-value weights w
for episode=1, M **do**
  Initialize S (first state of the episode)
  for t=1, T **do**
    $A_t \sim \pi_\theta(.|S)$
    Take action A_t, observe S_{t+1}, R_{t+1}
    $\delta = R + \gamma v_w(S_{t+1}) - v_w(S_t)$
    $w_{t+1} = w_t + \beta\delta\nabla_w v_w(S_t)$
    $\theta_{t+1} = \theta_t + \alpha\delta\nabla_\theta \log \pi_{\theta_t}(A_t|S)$
  end for
end for

2.5.4 Continuous Action Space

An advantage of policy-based and actor-critic over value-based is that they can deal with continuous action space while value-based can't. In policy-based and actor-critic, the actions can be chosen from a normal distribution (Gaussian) with mean and standard deviation derived by parametric function approximators (neural network) and thus these methods can deal with high and continuous action space range Sutton and Barto [2018].

2.6 Summary of this chapter

The reinforcement learning framework is formulated into a Markov decision process. The goal of RL is to find the optimal policy π^* that maximizes the cumulative reward. Given the transition function, dynamic programming can be used to find the optimal policy, however, the transition function is generally not available and difficult to obtain. On the other hand, in model-free reinforcement learning which is classified into three categories: value-based, model-based, and actor-critic, the agent can learn the optimal policy by interacting with the environment and relying on samples (s, a, r, s', r') without requiring to have a model of the environment.
In real-world applications, especially in the domain of HVAC systems, no models of

the environment dynamics are provided, and obtaining such models is considered a complicated task as explained earlier in Chapter 1, section 1.2.2.4. Hence, model-free RL is favored over model-based RL.

In the work of the PhD, the focus was to develop an end-to-end framework to optimize the energy consumption of HVAC systems while maintaining good thermal comfort and air quality. The privilege at the beginning was to develop a generic framework that doesn't require the knowledge of an expert or the environment's dynamics. Hence, at the beginning, an advanced MPC with a generic structure has been adopted to develop this framework, and the option of RL hasn't been chosen at the very beginning.

Chapter 3

Model Predictive Control for HVAC Optimization

In this chapter, a data-driven approach is developed for optimizing the fan power consumption of a standard HVAC system while maintaining indoor temperature and CO_2 concentration in an acceptable range. The data-driven approach represented by model predictive control (MPC) is followed to achieve this objective. MPC relies on predictive models and an optimization algorithm. The predictive models describe the connection between specific outputs (indoor temperature, CO_2) and controllable (air flow rate) and uncontrollable inputs (humidity, outdoor temperature, room occupancy, ...) while the optimization algorithm is used to generate optimal controls. The building model, equipped with the HVAC system is modeled in EnergyPlus. EnergyPlus is a building energy simulation software that can model the energy consumption, indoor evolution of temperature and CO2 concentration based on outdoor conditions (temperature, humidity, solar radiation) and HVAC system variables (cooling and heating setpoints of coils, water mass flow rate, air mass flow rate). To build the MPC, the predictive models are developed and validated first using data collected from EnergyPlus. Second, the optimization algorithm represented by the genetic algorithm is developed to optimize the fan power consumption with the option to modify the weights of the objective function by the user and thus can feature one output on another. Results show considerable energy savings achieved by MPC over EneryPlus while maintaining the same degree of thermal and air quality comfort.

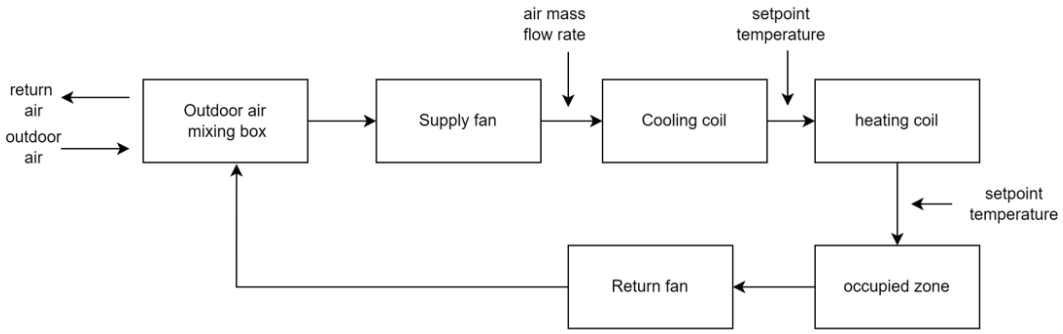


Figure 3.1: HVAC system.

3.1 HVAC system

In figure 3.1, an HVAC system is used to maintain thermal comfort and good air quality in the occupied zone. The outdoor air is injected into the occupied zone with the help of a supply fan. The air mass flow rate control specifies how much air should be injected inside the occupied space. The air will pass then by a cooling coil that cools the air if its temperature is above the setpoint temperature and then passes through a heating coil that heats the air if the temperature is below the setpoint temperature and finally it is injected into the occupied space. The return air is exhausted outside the building using a supply fan.

3.2 Problem Statement

In this chapter, air mass flow rate is the control that the MPC is trying to optimize to optimize energy consumption while maintaining thermal comfort and good air quality in the occupied zone. The other controls like setpoint temperature are not touched and are kept as in the initial controller: EnergyPlus. Fan air mass flow rate affects at the same time the fan power consumption and the heating/cooling power consumption since less air flow rate will mean less volume of air needed to be heated/cooled and thus less power consumption.

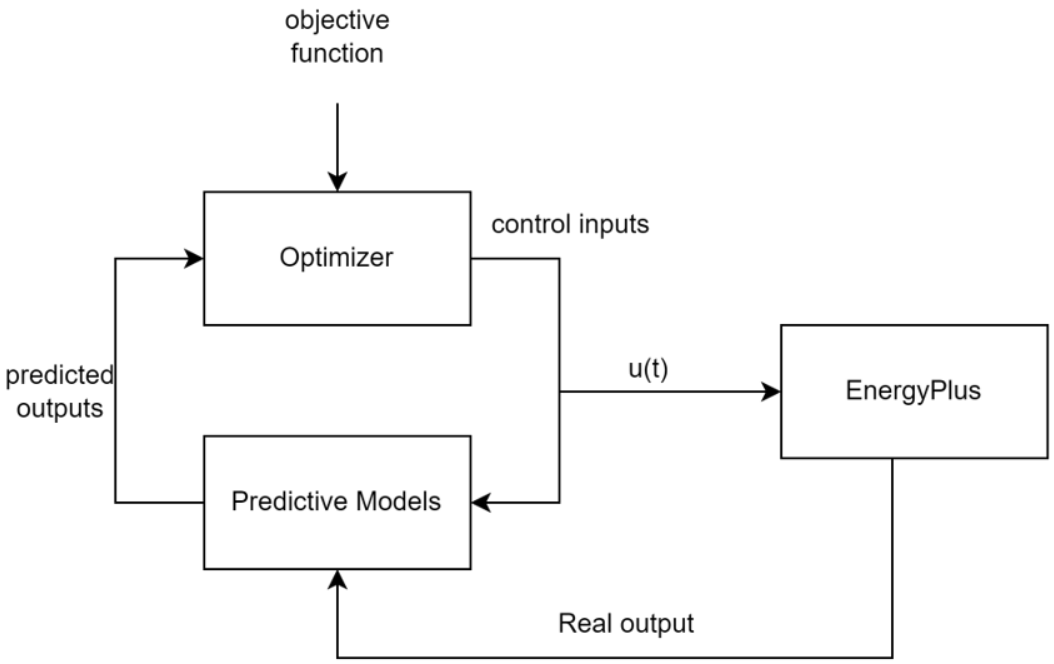


Figure 3.2: MPC feedback control.

3.3 Model Predictive Control: MPC

MPC is a control strategy that relies on internal models and an optimizer, see figure 3.2. The internal models are predictive models that can predict the state of the system in the next time steps based on some controls and system variables. The optimizer uses the predictive models to anticipate multiple future states ahead according to a specific value called "horizon", then it generates the optimal controls over a horizon but forwards with only the first control. The system, HVAC + zone, modeled in EnergyPlus uses the first control u(t) of the controls proposed by the optimizer and moves to the next timestep. The predictive models will then use the real measurements obtained at the next timestep by the system, which represent the system feedback to predict the next outputs, etc.

3.4 Predictive Models

The first phase of the MPC controller is to build a predictive model for the system variables: indoor temperature and CO2 concentration inside the zone. After analyzing different models: multilayer perceptron MLP (neural networks), linear regression, and

Model	T_{in} (° C)	C_{in} (ppm)
Linear regression	**0.1**	112
Support vector Regression	1.6	1490
Neural network	**0.1**	**48**

Table 3.1: MAE score computed over the test data set for the three predictive models described

support vector regression, neural networks have been chosen as predictive models for their best performance. The quality of the models is assessed according to the mean absolute error (MAE) metric:

$$MAE = \frac{1}{N}\sum_{i}|\hat{y}_i - y_i| \tag{3.1}$$

N is the size of the test data set, y_i is the real observed value (an EnergyPlus value in our case), and $\hat{y}_i$ is the predicted value. Table 3.1 shows the scores for the test data set.

3.4.1 Description of the Data

The data are obtained using EnergyPlus software[1]. We use two weeks of data (in February) to build the predictive models and 1 day to test the models. Data frequency is one value every 10 minutes. The training data and test data have different profiles for the number of occupants, weather temperature, and zone air flow rate (the only controllable variable under study). Data is divided into three categories:

- Controlled variables: indoor temperature (T_{in}), CO2 concentration (C_{in}), power consumption of the VAV box (P).
- control variables: zone air flow m_z (i.e damper position).
- context or uncontrollable variables: outdoor temperature (T_{out}), number of occupants (O).

3.4.2 Indoor Temperature Model

The indoor temperature is modeled as follows:

$$T_{in}(t+1) = f_T(T_{in}(t), T_{in}(t-1), T_{out}(t+1), O(t), O(t-1), m_f(t), m_f(t-1)) \tag{3.2}$$

[1]here again, this simulation software is used as if it is a real building, using a realistic small number of simulated days. We could also have used collected sensors' data if available

Tin, Tout, O, and m_f represent the indoor temperature, outdoor temperature, occupants, and fan air flow rate respectively. f_T is a multilayer perceptron (a neural network model) model that consists of 6 hidden layers with 6 neurons in each hidden layer with a logistic function used for activation. The fact that two previous values of some inputs are fed to the prediction model goes back to the inertia in the zone where some time is required to see the effect of an action on the state of the zone. Figure 3.3 shows the prediction of the indoor temperature over one day vs the simulated indoor temperature in EnergyPlus.

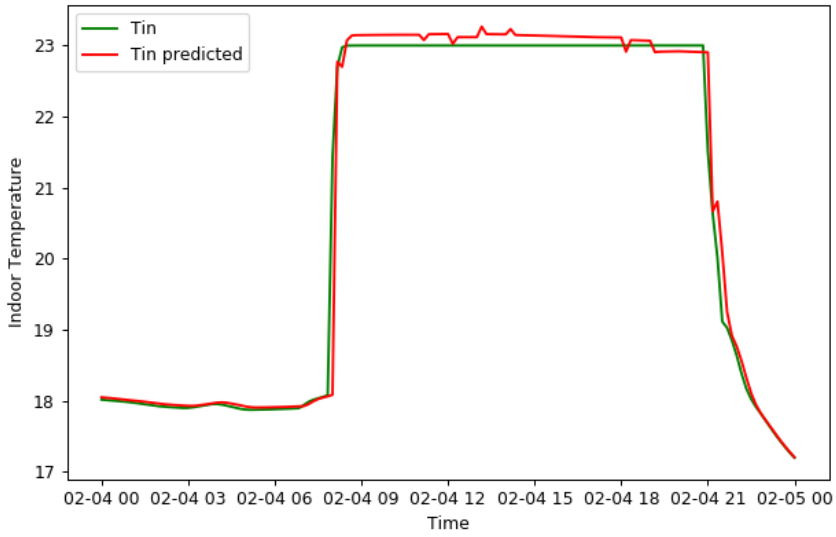


Figure 3.3: Indoor temperature predictive model.

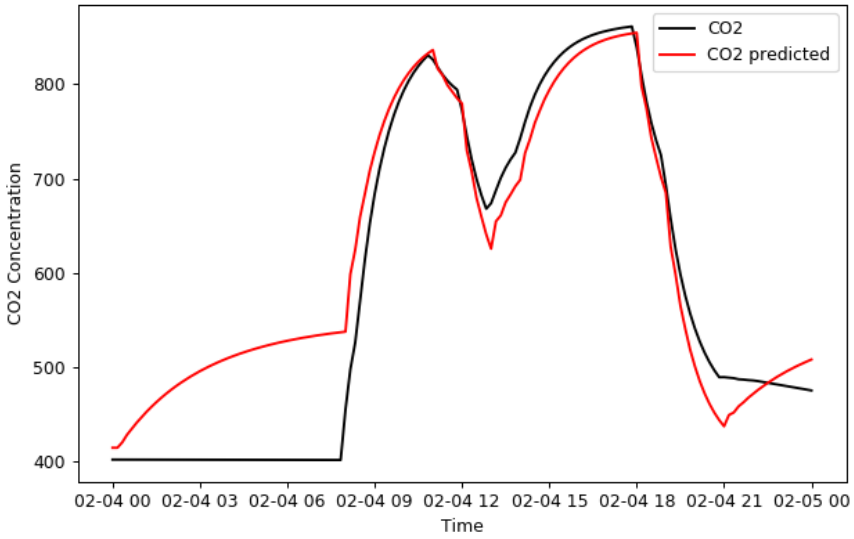


Figure 3.4: Indoor CO2 concentration predictive model

3.4.3 CO2 Concentration Model:

The indoor CO2 concentration is modeled as follows:

$$C_{in}(t+1) = f_C(C_{in}(t), C_{in}(t-1), O(t), O(t-1), m_f(t)) \tag{3.3}$$

Cin represents the indoor CO2 concentration. f_C is a multilayer perceptron that consists of 5 hidden layers with 4 neurons in each layer with a logistic function used for activation. Figure 3.4 shows the results obtained by the CO2 predictive model vs the CO2 simulated data in EnergyPlus. It is clear that there is a high error between the CO2 predicted and the CO2 simulated at the beginning of the day. This error is due to error propagation and in this case, it could have been smoothed to 400 ppm since there are no occupants.

3.4.4 Fan Energy Consumption Model:

The power consumption of the fan (in Watts) is linearly dependent on the fan air flow rate. It is represented by a linear model as follows:

$$P(t) = c * m_f(t) \tag{3.4}$$

c is a constant value. Figure 3.5 shows the accuracy of the prediction model over one day. The energy (E) at time t + 1 equals:

$$E(t+1) = P(t) * \Delta time \tag{3.5}$$

3.5 Optimization Algorithm

The second phase of the MPC controller is the optimizer or controller that decides what action to take to meet the objective function. The objective function is to optimize the energy consumption of the fan while keeping the indoor temperature and the CO2 concentration in the optimal range. The optimization equation is considered a multi-objective function since optimizing one feature can affect reversely the other. For example, reducing fan energy consumption will reduce the injection of outdoor air in the zone and thus will affect reversely the CO2 concentration. To cope with this situation, all outputs are multiplied by an importance weight that can be modified according to the importance needed. The optimization equation can be formulated as follows:

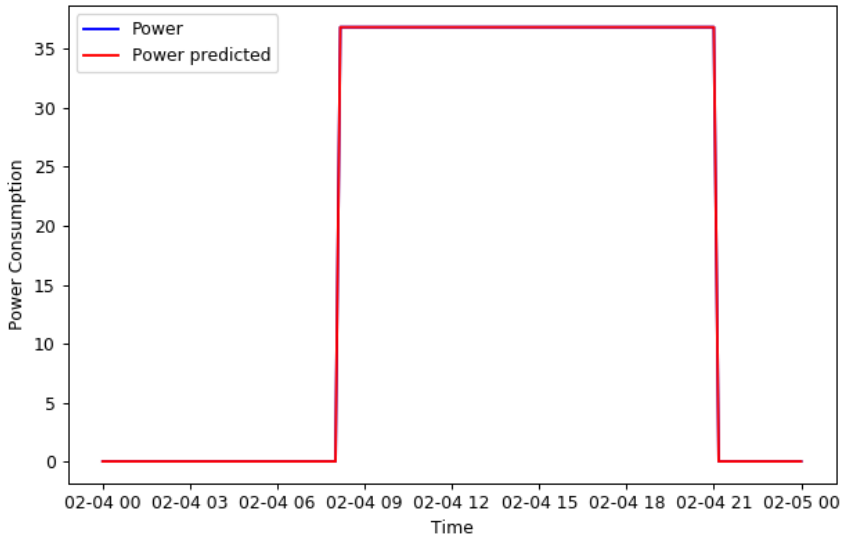


Figure 3.5: Power consumption predictive model.

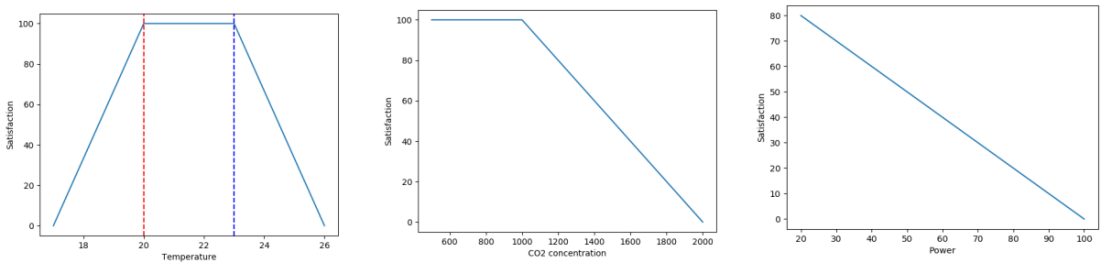


Figure 3.6: Satisfaction functions of indoor temperature, CO2 concentration, and power consumption.

$$\max_{\mu} \sum_{k=t+1}^{t+H} c_k(x_k, \mu_k) \tag{3.6}$$
$$c_k = w1 * sf_T(Tin(k)) + w2 * sf_C(Cin(k)) + w3 * sf_P(P(k))$$
$$\mu_k \in U, \quad k = t, ..., t+H-1$$

H represents horizon and sf_T, sf_C, and sf_P are respectively the user satisfaction with respect to the indoor temperature, the CO2, and the power consumption. In the optimization equation, the MPC controller generates H controls (μ_t ... μ_{t+H-1}) that maximize the objective function over the horizon. It relies on the developed predictive models to look H steps ahead to compute the system's evolution. However, the MPC takes only the first control of the generated H controls and then moves one step forward and repeats the same procedure. The idea of taking one step instead of H steps is to reduce the error

propagation that can be produced by the predictive models by taking the real measurement obtained by the system. c_k is the fitness function which represents the quality of the action taken. It is a weighted sum of thermal satisfaction, air quality satisfaction represented by CO2, and energy satisfaction (see equation 3.6). The weights, $w_1 ... w_3$, indicating the importance given to each of the features can be modified according to the comfort profile of the user. Figure 3.6 illustrates how satisfaction functions are computed.

Genetic algorithm (GA) McCall [2005], inspired by the process of natural selection, is used as an optimization algorithm. GA has the ability to search for a solution in parallel from a population of points and thus is less vulnerable to being trapped in a local solution. In more detail, the steps followed at time t by the genetic algorithm to control the fan air flow rates are as described as follows:

- Initialize a population of 50 individuals (an individual here is a window of H controls: m_z (kg/s)): Horizon = 6 steps and every step is over 10 minutes. e.g: [t = 10:00: 0.1, t = 10:10: 0.1, t = 10:20: 0.1, t = 10:30: 0.1, t = 10:40: 0.1, t = 10:50: 0.1].
- Compute the fitness function of the individuals:

$$c_t = F(t) = w_1 * sf_T(T_{in}(t)) + w_2 * sf_C(C_{in}(t)) + w_3 * sf_P(P(t)) \tag{3.7}$$

- Take the best-performing individuals (30% of the population) having the highest fitness to be the parents for the next generation.
- Breed together the parents to repopulate the population to its desired size according to a crossover operator (e.g., (p1 = [0.1, 0.1, 0.1, 0.1, 0.1, 0..1]) X (p2 = [0.2, 0.2, 0.2, 0.2, 0.2, 0.2]) = [0.1, 0.1, 0.1, 0.2, 0.2, 0.2]). In this example, the crossover occurs in the middle. There are many types of crossover operators, see Soni and Kumar [2014]. The most common crossover operator, used in the experiments, is the single-point (called as well one-point) crossover where a random crossover point is selected to generate the child.
- Merge together the parents and children to constitute the next generation population.
- Apply mutation for a small part of the population to increase population diversity: add small noise ε.
- Repeat previous steps n times: n is the number of generations.

- In the last generation, compute the fittest individual that defines the best controls over the horizon.

The MPC controller takes the first control of the generated controls over a horizon and moves one step forward. The optimization algorithm depends on the weights in the fitness function. For example, more priority can be given to air quality by increasing its weight and thus having a lower CO2 concentration.

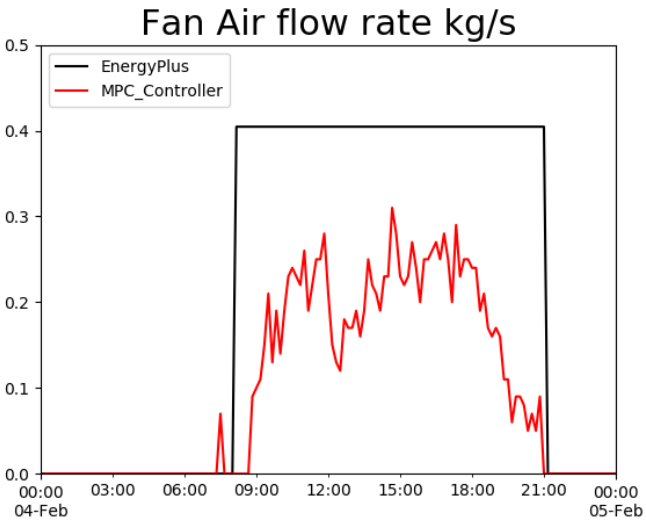


Figure 3.7: Fan air flow rate control.

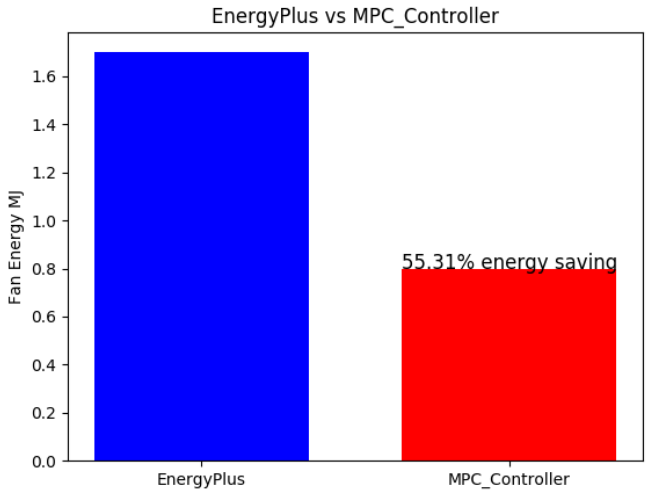


Figure 3.8: The results of the fan energy consumption.

3.5.1 MPC controller first configuration results

The MPC controller is tested over one day in February. It is applied one hour before the zone is occupied (at 7 am) until the zone is not occupied (9 pm). When the room is not occupied, the fan air flow rate is set to 0. As mentioned above, the objective function includes weights for thermal, CO2, and energy satisfaction. These weights can be updated according to the occupants' preferences. the first configuration is set with the following weights: 0.2, 0.4, 0.4 for temperature, CO2, and energy consumption respectively. The MPC controller results are compared with the reference controls obtained by EnergyPlus.

3.5.1.1 Fan air flow rate and energy consumption results

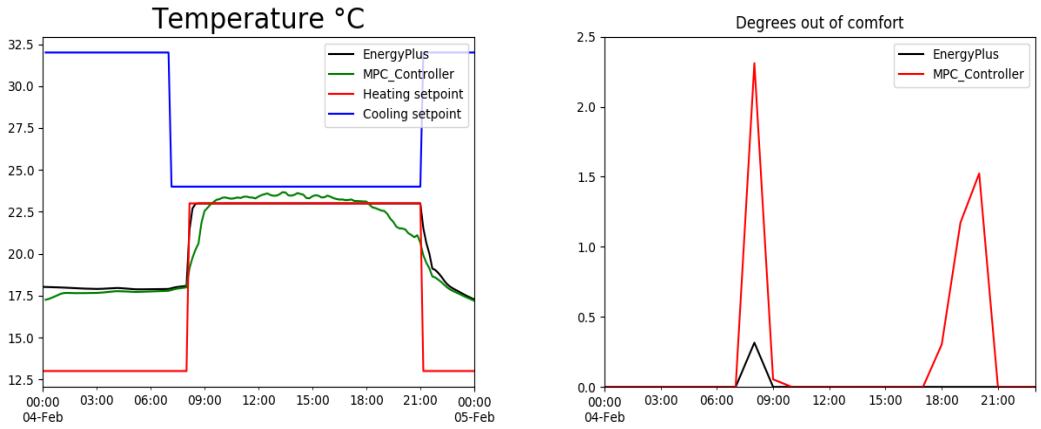
With the first weights' configuration, the MPC controller generates the controls in figure 3.7 that leads to 55.31 % energy reduction compared with EnergyPlus, figure 3.8.

3.5.1.2 Indoor CO2 and temperature results

With the controls generated by the MPC controller, the indoor temperature and CO2 concentration inside the zone have the following results illustrated in figures 3.9 and 3.10 respectively. Heating and cooling setpoint temperatures are fixed in EnergyPlus. The highest thermal satisfaction is considered between 20 °C and 23 °C where good air quality is represented by CO2 < 1000 ppm (figure 3.6). With the high energy savings obtained, the CO2 overcomes most of the day 1000 ppm leading to low air quality, figure 3.10. Concerning thermal comfort, the indoor temperature falls highly outside the comfort level at the beginning and the end of the day as shown in figure 3.9 b. As a result, the energy consumption of the fan is highly reduced with respect to EnergyPlus, however, thermal and air quality comfort are not maintained as in EnergyPlus. To improve the results, the configuration of the weights is updated where the weight of power is decreased and the importance of CO2 is increased.

3.5.2 MPC controller second configuration results

The weights are updated as follows: 0.2, 0.5, 0.3 for temperature, CO2, and energy consumption respectively.



(a) Indoor temperature control.

(b) Average degrees out of comfort per hour.

Figure 3.9: The degrees out of comfort metric computes the number of degrees outside the comfort level.

3.5.2.1 Fan air flow rate and energy consumption results

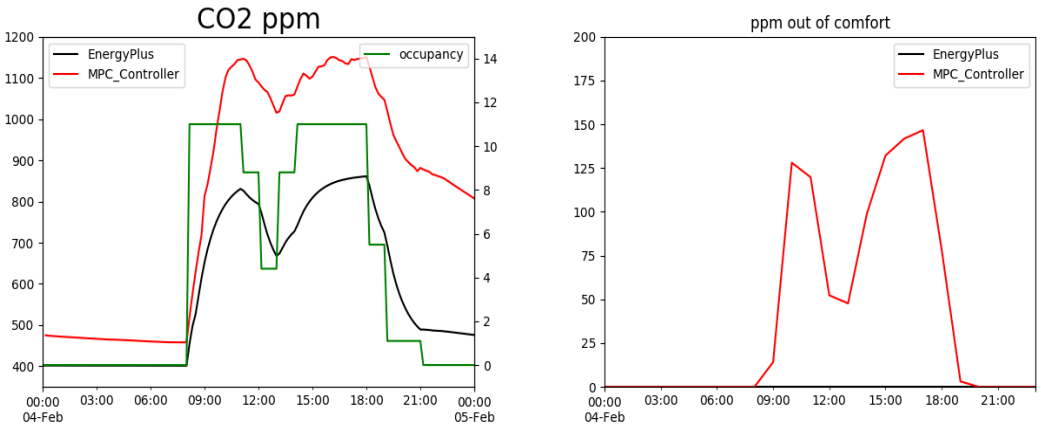
With the first weights' configuration, the MPC controller generates the controls in figure 3.11 that leads to 36.48 % energy reduction compared with EnergyPlus, figure 3.12.

3.5.2.2 Indoor CO2 and temperature results

With the controls generated by the MPC controller, the indoor temperature and CO2 concentration inside the zone have the following results illustrated in figures 3.13 and 3.14 respectively. Increasing the importance of the CO2 concentration, the CO2 achiever a better level now leading to better air quality. Figure 3.14 b shows that CO2 maintains a good level all day. Concerning the thermal comfort, it is improved as well where the indoor temperature falls slightly ($\sim$ 1°C) outside the comfort level at the beginning and the end of the day as shown in figure 3.13 b. As a result, the energy consumption of the fan is well reduced with respect to EnergyPlus while maintaining good comparable good air quality and slightly lower thermal comfort.

3.6 Conclusion

In this chapter, MPC controller was able to reduce greatly fan energy consumption while maintaining good thermal comfort and good air quality with the option of modifying the importance weights. MPC is built in a generic way where the predictive models



(a) Indoor CO2 concentration control.

(b) Average ppm out of comfort per hour.

Figure 3.10: The ppm out of comfort metric computes how much the CO2 falls outside the comfort level.

are not given by an expert but developed from data collected. Yet, these predictive models require considerable work of developing and validating. In addition, they must be very accurate so that the MPC generates good controls. Hence, in the coming work, model-free RL that doesn't require knowledge of the environment's dynamics is explored. Yet, reinforcement learning suffers from the huge time required to learn a good controller and thus our contribution is elaborated in the coming chapters, which illustrates the approaches developed to speed the convergence of reinforcement learning.

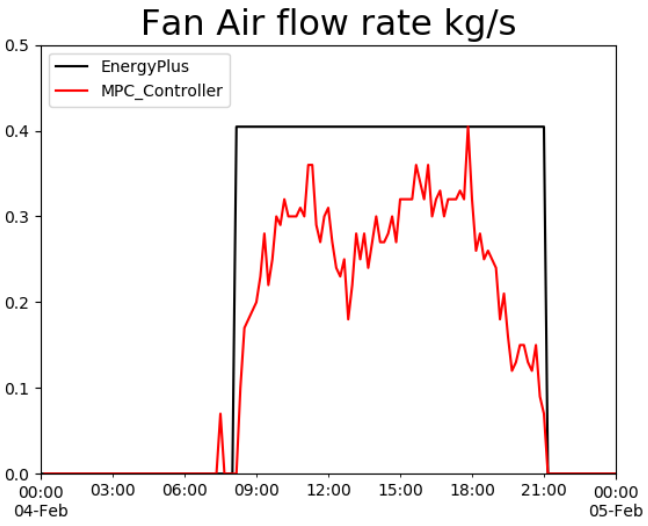


Figure 3.11: Fan air flow rate control.

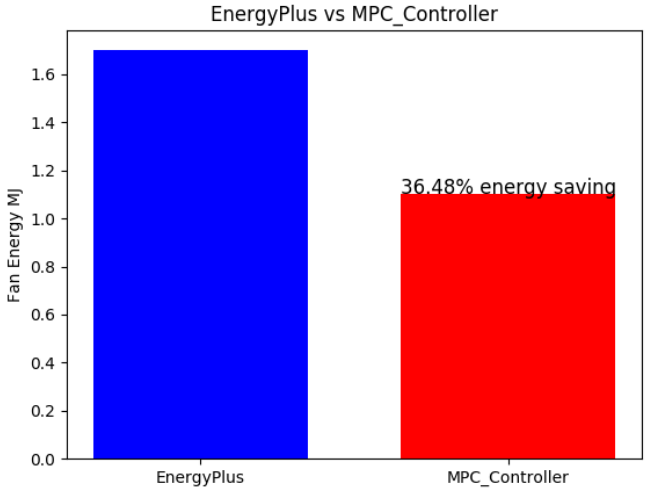
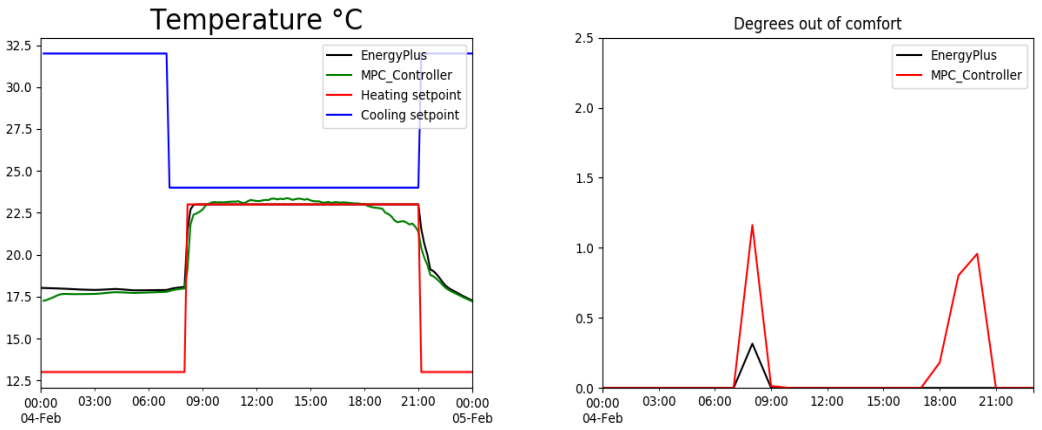


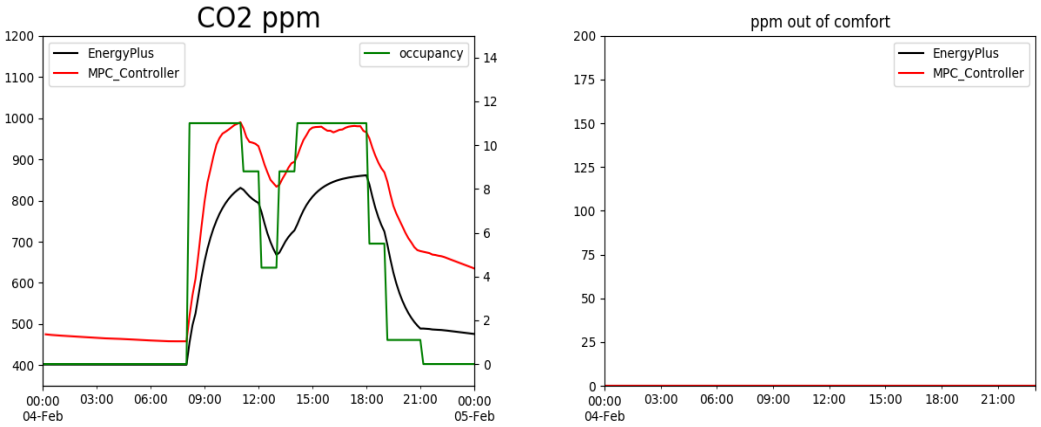
Figure 3.12: The results of the fan energy consumption.



(a) Indoor temperature control.

(b) Average degrees out of comfort per hour.

Figure 3.13: The degrees out of comfort metric computes the number of degrees outside the comfort level.



(a) Indoor CO2 concentration control.

(b) Average ppm out of comfort per hour.

Figure 3.14: The ppm out of comfort metric computes how much the CO2 falls outside the comfort level.

Chapter 4

Deep Q-Network Boosted with External Knowledge for HVAC Control

This Chapter is an updated version of the published article in Jneid et al. [2021]. Traditional techniques such as rule-based control (RBC) fail to control HVAC systems optimally. As shown in previous chapters, MPC has been widely explored in literature but it doesn't represent a practical solution due to the complexity of buildings' dynamics that it relies on. Hence, another type of control called model-free deep reinforcement learning (DRL) that doesn't require initial knowledge of the environment has been well explored during the thesis for the objective of optimizing HVAC energy consumption while maintaining thermal comfort and good indoor air quality. However, DRL requires a lot of data and time to converge, see section 1.2.3.4, and thus the goal is to come up with a way to boost the convergence of DRL. In this chapter, two main DRL approaches, which belong to deep-Q network (DQN) Mnih et al. [2013], have been explored. The first approach, the basic approach, represents a DQN agent with no initial knowledge of the environment and the second approach represents a DQN agent with initial knowledge: A hybrid approach DQN+RBC. The idea behind combining RBC with DQN is to guide the DQN agent at the beginning by exploiting the knowledge of RBC rules, which speeds the convergence. We evaluate the performance of these two approaches against an RBC approach in a simulated office, equipped with a heating radiator, during the heating period.

4.1 Study Case Formulation

An office located in Grenoble-INP building has been simulated and validated against real data to be used in the experiments. The office has a door and window and is equipped with a heating radiator. The office simulator contains models for estimating the output variables: indoor temperature, CO2 concentration, and power consumption using controllable, uncontrollable contextual variables. Controllable variables represent the variables that can be controlled: radiator heating setpoint temperature, window control, and door control. Uncontrollable variables represent variables that affect the output variables but can't be controlled such as outdoor temperature, solar radiation, CO2 concentration in the corridor next to the office, corridor temperature, and the number of occupants inside the office. The output variables at the next time step depend on the current state of the office and the current controls. As a result, the office optimization problem can be formulated as a Markov decision process (MDP) and solved using reinforcement learning as follows:

Control actions: The office consists of three main controls: radiator temperature setpoint with discrete levels [13°C, 19°C, 21°C, 23°C, 25°C], door control and window control with discrete values [0 for closed, 1 for open]. The window allows the injection of outdoor air in the office while the door enables the flow of the corridor air.

System States: The office state at time t consists of different observations or features: current indoor temperature, current indoor CO_2 concentration, number of occupants, outdoor temperature, solar radiation, corridor temperature, corridor CO_2 concentration, and outdoor temperature at next time step (forecast feature). The optimal controls are computed w.r.t the current state of the office under the objective of maintaining good indoor temperature and good indoor CO_2 concentration while saving energy consumption.

Reward function: The objective of the DRL algorithm is to find the optimal policy (controlling the radiator temperature setpoint, door control, and window control) leading to the sequence of (states, actions) $s_1, a_1, s_2, a_2, ..., s_m, a_m$ with the highest cumulative reward $G_t = \sum_t \gamma_t R(s_t, a_t)$. m represents the length of the episode: 1 day in our study case and the length of timestep is 1 hour. γ is the discount rate specifying how much importance should be given to future rewards over immediate rewards. The optimal controls are computed at each timestep by the optimal policy. After taking an action a_t at timestep t, the agent receives an immediate reward (Figure 4.1) as follows:

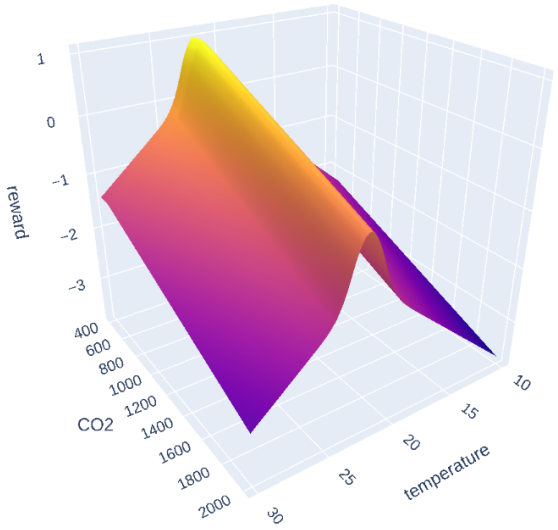


Figure 4.1: Reward received by the agent after taking an action and moving to the next state: The reward in the plot takes into consideration the thermal comfort and the air quality assuming the power consumption equals zero. The power consumption will shift the complete plot downward.

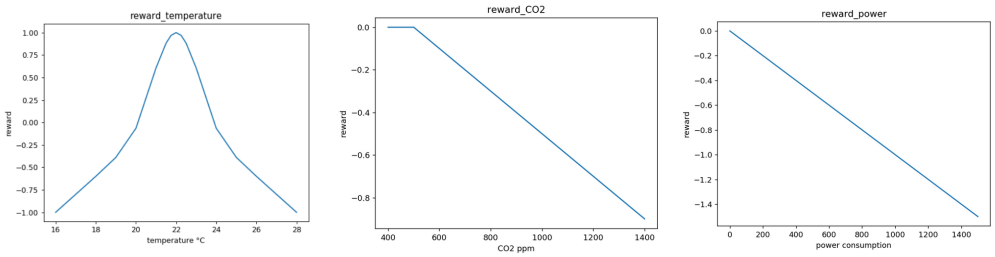
$$R_t = \lambda_1 * R_T(t) + \lambda_2 * R_C(t) + \lambda_3 * R_P(t) \tag{4.1}$$

$$R_T(t) = \begin{cases} e^{-(T_t - \underline{T}_{comfort})^2} - [T_t - \underline{T}_{comfort}]_+ & \text{if } T_t < \underline{T}_{comfort} \\ e^{-(T_t - \overline{T}_{comfort})^2} - [T_t - \overline{T}_{comfort}]_+ & \text{if } T_t > \overline{T}_{comfort} \\ 1 & \text{if } T_t \text{ is between } [\underline{T}_{comfort}, \overline{T}_{comfort}] \end{cases} \tag{4.2}$$

$$R_C(t) = \min(0, (500 - C)/(1500 - 500)) \tag{4.3}$$

$$R_P(t) = -P(t) * cost(kwh) \tag{4.4}$$

λ_1, λ_2, and λ_3 specify the importance given to each feature during the training. In the experiments, they are all set to one. To increase the importance of a feature, its lambda can be increased. T, C, and P represent indoor temperature, indoor CO_2 concentration, and power consumption of the radiator respectively. $[\underline{T}_{comfort}, \overline{T}_{comfort}]$ represents the comfort range of the temperature where it is in the range of [21°C, 23°C] during the occupied time from 8 am until 6 pm and in the range of [13°C, 30°C] otherwise. Figure 4.2 shows the reward for every indoor temperature, indoor CO2, and power consumption respectively. The training of DRL algorithm starts every day one hour before the occupied time and ends at 6 pm to allow the algorithm to preheat if needed so that the



(a) Indoor temperature (b) Indoor CO2 (c) Power

Figure 4.2: Reward functions considered in the training of RL algorithms.

office reaches the thermal comfort range during the occupied time, otherwise, the heating setpoint temperature is set at 13°C and the window/door are closed.

4.2 Rule Based Control vs DRL Control

4.2.1 Rule-based control (RBC) approach

RBC, a model-free approach, is mostly used in industry and thus is used for comparison with the DQN approaches. RBC is a simple approach containing predefined rules that specify the HVAC operation. Algorithm 3 illustrates the rules used to control the radiator heating setpoint, the window, and the door in the simulated office.

4.2.2 Model Free DRL approaches

4.2.2.1 DQN without Knowledge

DQN Mnih et al. [2013] uses a neural network that works as a function approximator where it takes as input a continuous state vector and outputs the Q values with every possible action in a single forward pass as shown in figure 4.3. We have five possible values for the radiator heating setpoint, and 2 possible values (0/1) for the window and the door. Thus, the final layer outputs the Q values of 20 possible combinations of actions. Q value represents the expected future discounted reward for taking an action a at a specific state. During the training of the DQN, the weights θ of the Q network are updated by applying a mini-batch gradient descent step on the loss function L with respect to θ:

Algorithm 3 RBC Approach

Require: time, day, office_temperature, corridor_temperature, outdoor_temperature, indoor_CO_2, office_CO_2

window, door, temperature setpoint = 0, 0, 21

if time not in [7 am, 8 pm] or day is weekend **then**

temperature setpoint = 13

else {time in [7 am, 8 pm] and working day}

if office_$CO_2 \geq 1000$ **then** {This approach gives more priority to CO_2 comfort than temperature comfort}

window = 1

if corridor_CO2 < office_CO_2 **then**

door = 1

if office_temperature < 21 **then**

temperature setpoint = 23

if office_temperature > 23 **then**

temperature setpoint = 19

else {office_CO_2 < 1000}

if office_temperature < 21 **then**

if outdoor_temperature > office_temperature **then**

window = 1 {To heat}

if corridor_temperature > office_temperature **then**

door = 1 {To heat}

if office_temperature > 23 **then**

if outdoor_temperature < office_temperature **then**

window = 1 {To cool}

if corridor_temperature < office_temperature **then**

door = 1 {To cool}

$$L = 1/2n \sum_{i=1}^{n} [\hat{Q}(s_t, a_t^i) - Q(s_t, a_t^i)]^2 \tag{4.5}$$

$$\hat{Q}(s_t, a_t) = R_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) \tag{4.6}$$

Algorithm 4 DQN with target network and experience replay

Initialize replay memory D to capacity N

Initialize action-value function Q with random weights θ

Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$

for episode 1, M **do**

for t=1,m **do**

Observe state s_t

following ε-greedy policy, action $a_t = \begin{cases} \text{random action} & \text{with probability } \varepsilon \\ \arg\max_a Q(s_t, a_t; \theta) & \text{otherwise} \end{cases}$

Execute action a_t in the simulator and observe reward R_t and state s_{t+1}

Store experience (s_t, a_t, R_t, s_{t+1}) in D

Sample random minibatch of experiences (s_j, a_j, R_j, s_{j+1}) from D

SET $y_j = \begin{cases} R_j & \text{if episode terminates at step j + 1} \\ R_j + \gamma \max'_a \hat{Q}(s_{j+1}, a'; \theta^-) & \text{otherwise} \end{cases}$

Perform a minibatch gradient descent step on the loss function L with respect to the weights θ

Every C steps reset $\hat{Q}$ = Q

end for

end for

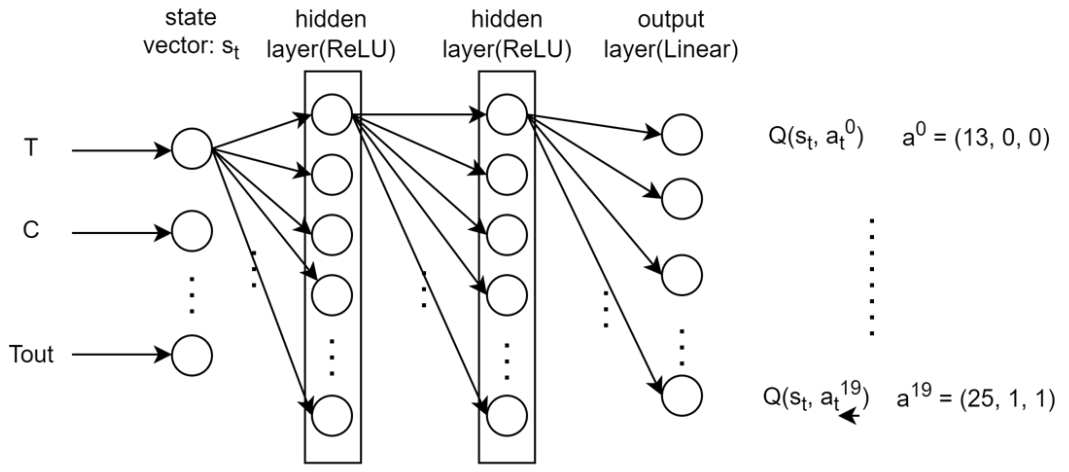


Figure 4.3: The structure of the Q network used to approximate the Q values.

Another $\hat{Q}$ network called the target network is used during the update to stabilize the learning process. In addition, a minibatch is randomly sampled from an experience replay buffer to break the correlation between consecutive samples. The policy used during the training is called ε-greedy policy. ε is used to control the trade-off between exploration and exploitation. A random action is proposed with probability ε, otherwise the action proposed with probability $(1-\varepsilon)$ is:

$$a_t = arg\max_a Q(s_t, a_t) \tag{4.7}$$

This approach is called DQN without knowledge (Algorithm 4) because it doesn't use any knowledge of the environment and learns the optimal policy by interacting only with it (model-free).

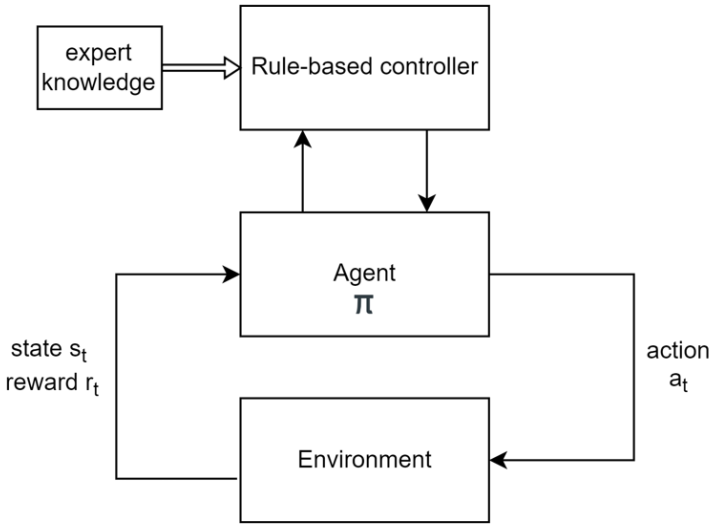


Figure 4.4: The structure of DQN with knowledge approach is used to exploit the RBC rules to boost the convergence and learn a better policy faster.

4.2.2.2 DQN with knowledge: DQN + RBC

DQN with knowledge, which represents the contribution made in this chapter, is a hybrid approach that relies on both approaches: RBC and DQN (Figure 4.4). The idea of DQN with knowledge is to guide the policy at the beginning using RBC rules instead of complete random exploration. The rules don't need to be perfect and they will help in guiding the policy at the beginning. After some time, the agent has gained some knowledge aiming at learning a better policy than the initial one and thus the external knowledge is not needed anymore. Policy guidance is added by updating the ε-greedy

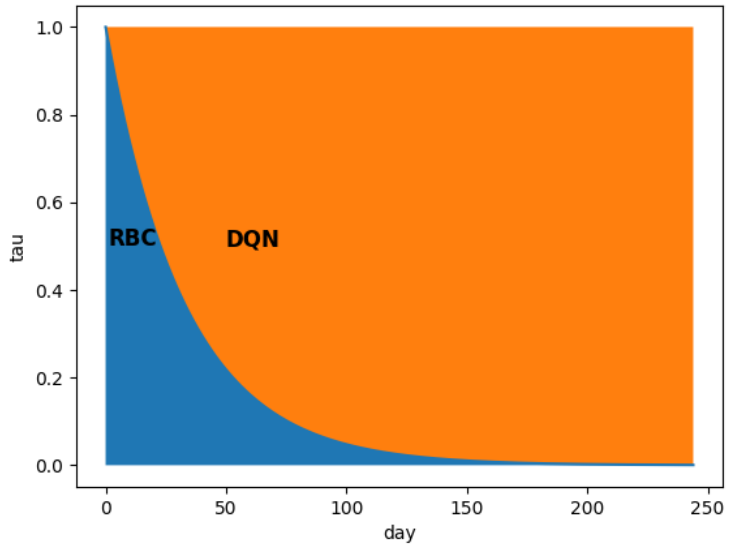


Figure 4.5: DQN+RBC starts with a high percentage of exploitation of RBC rules at the beginning since there is no knowledge of the environment. Then with time, the agent relies completely on DQN to control the system.

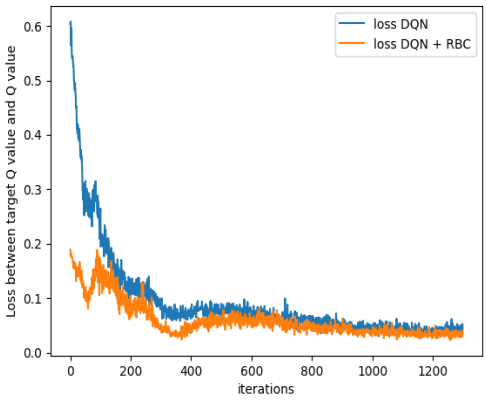
policy where the new policy contains another parameter τ that specifies how much to exploit the RBC rules, which represent the external knowledge, as shown in figure 4.5.

Algorithm 5 DQN+RBC policy

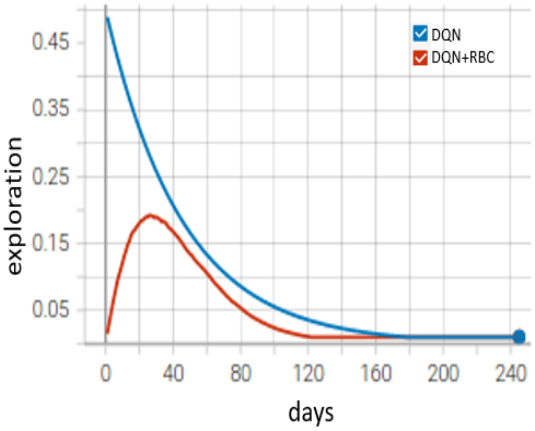
Require: state, τ
  if random.uniform() < τ **then**
    action = RBC(state)
  else:
    action = ε-greedy policy(state)

4.3 Training

The hybrid approach DQN+RBC is trained against the RBC controller and DQN controller in the heating period, [February, March, April, October, November, and December] since the office is equipped with a heating system. The approaches are tested using a simulator of the office to perform more experiments and to be able to compare the different approaches. The simulator is used as a real environment and then training is done in online way where an episode starts at 7 am and ends at 6 pm. The goal is to find



(a) Squared error between target $\hat{Q}$ value and Q value over time.



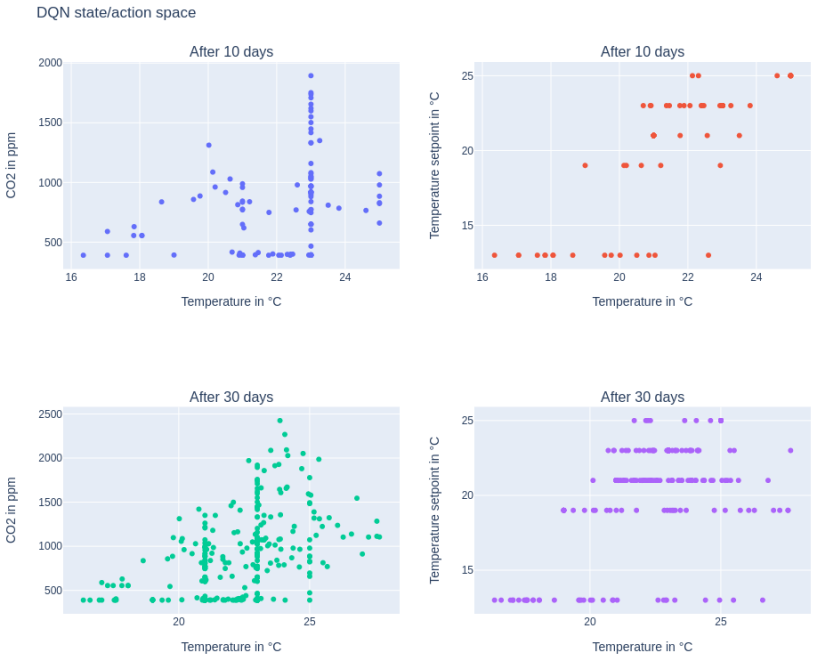
(b) DQN vs. DQN+RBC exploration

Figure 4.6: Figure (a) shows that the loss function of DQN starts with much higher value than the loss function of DQN+RBC because of the high initial exploration of the environment using the DQN controller as shown in figure (b).

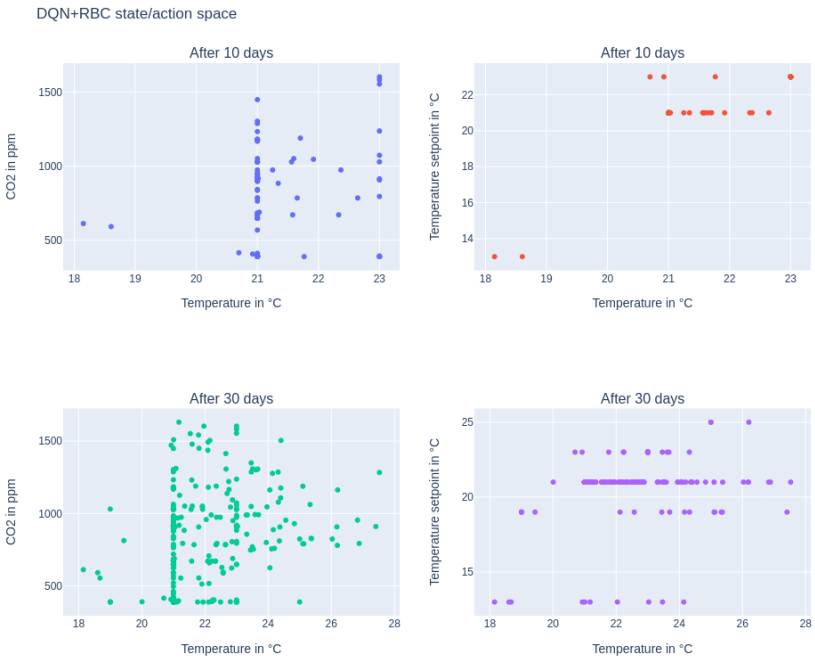
how many days are required in a real environment for a model-free DRL approach to converge to a good policy better than the RBC controller.

4.3.1 Loss, Exploration

The squared error between the target $\hat{Q}$ value and the Q value, figure 4.6 a, is used to train the DRL approaches: DQN+RBC and DQN. The loss function converges faster in the hybrid approach than the DQN approach that has high exploration at the beginning (figure 4.6 (b)) unlike the hybrid approach that relies more on the exploitation of the external knowledge introduced by the RBC controller at the beginning (figure 4.5).
DQN doesn't benefit from any initial knowledge and relies only on interacting with the environment to learn a good policy. Thus, it tries more different actions and explores more states at the beginning than the DQN+RBC approach. This functionality is visualized in Figure 4.7 where more states have been visited after 10 days in DQN compared to DQN+RBC which tries to exploit the RBC rules. Then with time, DQN+RBC relies less on RBC and explores more to visit new states, illustrated in Figure 4.7 as well after 30 days. After some time, DQN and DQN+RBC have gained some knowledge and thus they exploit what they have learned with a very small percentage of exploration (1% in the experiments).



(a) DQN exploration



(b) DQN+RBC exploration

Figure 4.7: DQN state/action space vs. DQN+RBC state/action space exploration after 10 and 30 days of execution.

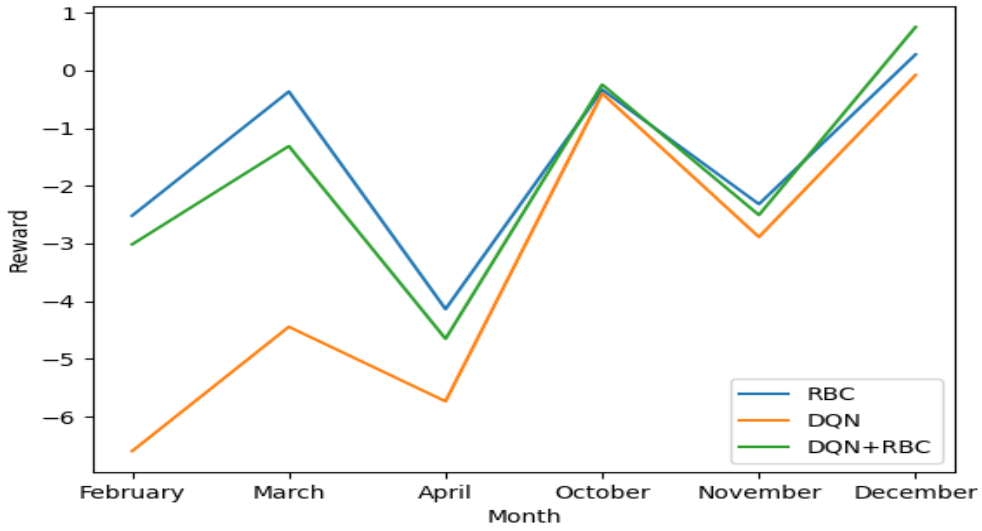


Figure 4.8: Reward value averaged over one month during the heating period.

4.3.2 Metric Performance

To analyze the performance of the hybrid approach, the comparison is made according to the reward value (figure 4.1) used during the training that reflects the quality of the action concerning thermal comfort, air quality, and energy consumption. Figure 4.8 shows the reward value during the training. The DQN+RBC takes good advantage of the RBC rules at the beginning to have nearly performance to RBC while DQN has very far and bad performance from the others.
The training is done day by day and the policy obtained at the end of the day is used at the beginning of the next day. It is difficult to visualize the reward performance over days thus the reward is averaged over one month and visualized month by month in figure 4.8. Even though the DQN+RBC approach benefits from the guidance of RBC rules at the beginning of the training, it is taking 4 months of training to reach the performance of a simple RBC approach and more than 5 months to outperform it.

4.4 Testing the policies

The three policies are tested in three different scenarios to show the functionality of the controls: door opening, window opening, and temperature setpoint temperature. For the DRL approaches, the policy obtained at the end of the heating period (December) is

saved and used for testing and comparing the controls' outputs against the RBC outputs in different situations.

4.4.1 Door control

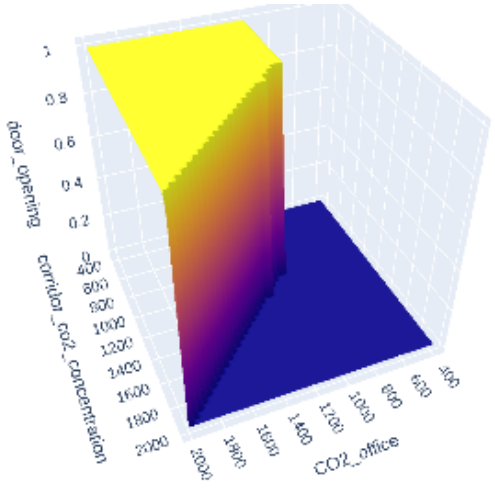
The office has a door next to a corridor and then the CO2 in the corridor has some effect on the CO2 inside the office. To test the door control obtained by the different approaches, two vectors of CO2 office and CO2 corridor are fed to the approaches with fixed values for the other state features.

- RBC approach (figure 4.9 a): The door is closed when the CO2 office is less than 1000 ppm regardless of the CO2 in the corridor. When the CO2 in the office starts to increase above 1000 ppm, the door is open whenever the CO2 in the corridor is less than the CO2 in the office.
- DQN approach (figure 4.9 c): The door is open whenever the CO2 in the corridor has a low range where its value is not greater than 1100 ppm. When CO2 in the corridor overcomes 1100 ppm, the door is always closed.
- DQN+RBC approach (figure 4.9 e): When the CO2 in the office is very low, less than 800 ppm, the door is closed. The door is open when the CO2 in the office increases above 800 ppm and the CO2 value in the corridor is less than 1600 ppm and less than the CO2 in the office. After the CO2 in the corridor reaches a value above 1600 ppm, the door is kept closed even if it has a value less than the CO2 in the office.

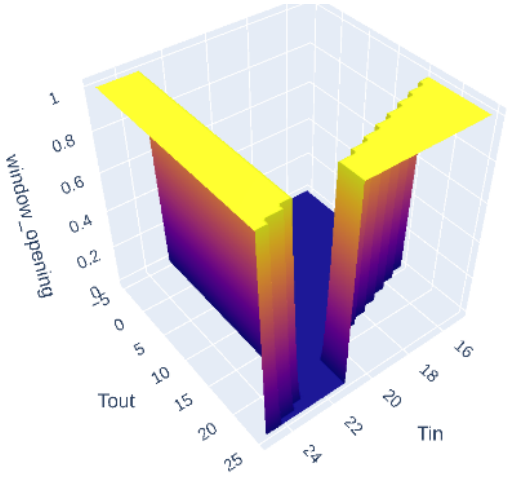
4.4.2 Window control

The window has a direct effect on the indoor temperature of the office. Two vectors of the indoor temperature of the office and outdoor temperature are injected in the control approaches to test the control functionality of the window.

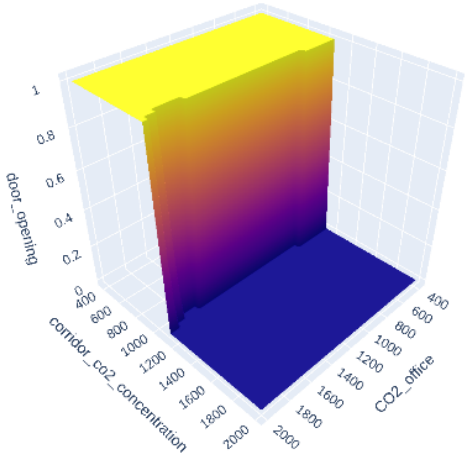
- RBC approach (figure 4.9 b): The window is closed when the indoor temperature is in the comfort range [21 °C, 23 °C] regardless of the outdoor temperature. When the indoor temperature reaches a value greater than 23 °C, the window is open if the outdoor temperature has a lower value than the indoor temperature in order for the office to cool down. When the indoor temperature goes below 21 °C, the



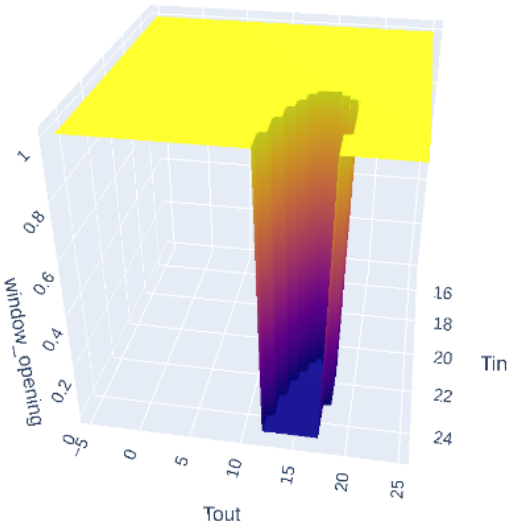
(a) RBC control



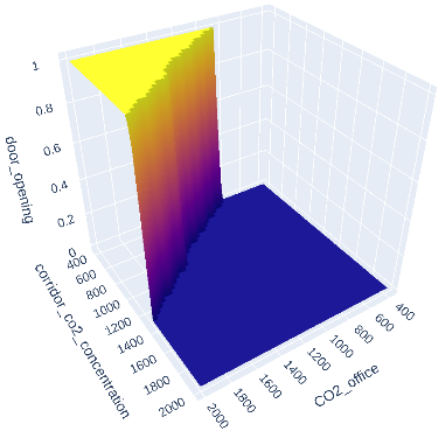
(b) RBC control



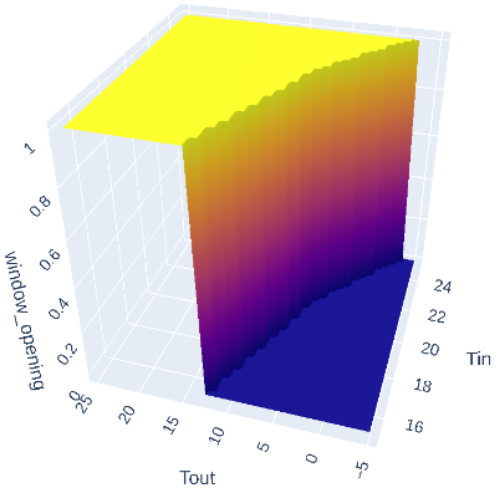
(c) DQN control



(d) DQN control



(e) DQN+RBC control



(f) DQN+RBC control

Figure 4.9: The door control (left) and the window control (right) obtained by the three approaches.

window is opened if the outdoor temperature has a greater value than the indoor one to allow for outdoor air injection in the office and then heating the office; otherwise, it is closed.

- DQN approach (figure 4.9 d): The window is closed when the indoor temperature is above 21 °C and the outdoor temperature is between 12 °C and 17 °C, otherwise, it is open. The window control obtained by the DQN approach doesn't reflect a good control strategy and seems that it hasn't converged towards a good policy yet.

- DQN+RBC approach (figure 4.9 f): The window is open when the indoor temperature is low and the outdoor temperature has a greater value which allows for heating up the office. When the outdoor temperature has a very low value, the window is kept closed to prevent a cold indoor temperature. The window is open as well even if the indoor temperature is in the comfort range and the outdoor temperature is not sharply cold to prevent a high increase in indoor CO2. The policy obtained cannot generate good control when the indoor temperature is in the comfort range ([21 °C, 23 °C]) and the outdoor temperature is very high (>= 25 °C). A good control policy should keep the window closed in such a case but since the training is done in the winter season, there are not enough samples that are close to this situation during the training to learn this reaction, see figure 4.10. To resolve this issue, the hybrid approach, DQN+RBC, could be updated to include an exploitation parameter for every rule in the RBC rules that has been triggered to keep track of the rules visited, rule by rule. Then, every rule's exploitation parameter declines every time its rule is visited. Thus, if the rule is not visited at all, it will be kept and used by the policy to ensure a good initial control if encountered later instead of a random control.

4.4.3 Temperature setpoint control

The office is equipped with a heating radiator to heat up the indoor temperature. The heating radiator is considered to have a set of five discrete setpoints temperatures [13 °C, 19 °C, 21 °C, 23 °C, 25 °C]. The setpoint temperature operating is visualized with respect to indoor temperature and CO2 of the office for the three different approaches: RBC, DQN, and DQN+RBC.

- RBC approach (figure 4.11 a): The temperature setpoint is 19 °C when the indoor temperature is greater than the upper temperature of the comfort range, i.e 23 °C

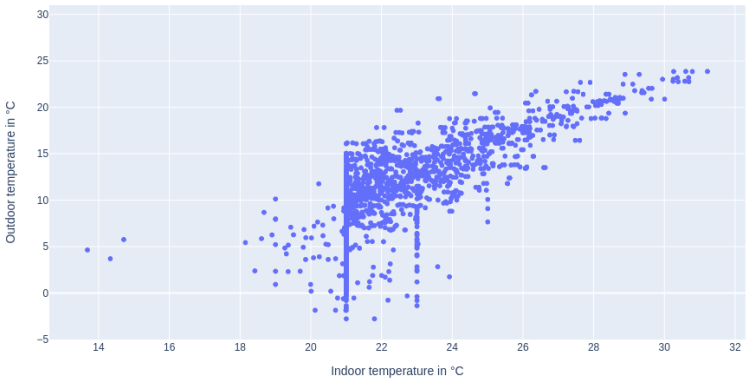
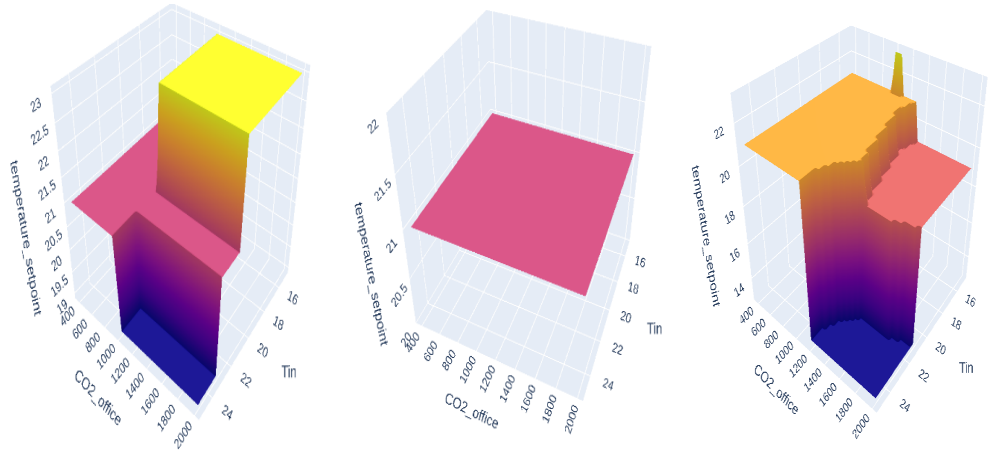


Figure 4.10: Tin vs Tout exploration during the training phase ...

and the air quality inside the office starts to drop, i.e CO2 office > 1000 ppm. In this case, the office is highly occupied and tools that consume power and generate heat are used which both lead to increasing the indoor temperature and thus temperature setpoint control could be lowered under 21 °C to take advantage of the heat generated and reduce power consumption of the radiator. When the indoor temperature is in the comfort range, the temperature setpoint is 21 °C regardless of the CO2 office. The temperature setpoint is kept at 21 °C when indoor temperature is lower than 21 °C and CO2 office is lower than 1000 ppm while it is increased to 23 °C when the indoor temperature is low as well but the CO2 office is higher than 1000 ppm. This rule doesn't look good and should be the opposite but it is used intentionally to check if the DQN+RBC will be able to modify it to a good one.

- DQN approach (figure 4.11 b): The DQN hasn't converged yet to a good control policy. The setpoint temperature is always 21 °C.

- DQN+RBC approach (figure 4.11 c): When indoor temperature increases above 21 °C and CO2 office above 1100 ppm, the temperature setpoint generated by the policy equals to 13 °C. When indoor temperature is above 21 °C but the CO2 inside the office is low (< 1100 ppm), the temperature setpoint equals to 21 °C. In the RBC approach, there is a bad rule that generates, in the case of low indoor temperature and high CO2 office, a higher setpoint temperature than the case of low indoor temperature and low CO2 office. In this case, this rule is updated by DQN+RBC to a better one that generates in the case of low indoor temperature and high CO2 office a lower temperature setpoint (19 °C) than in the case of low indoor temperature and low CO2 office (temperature setpoint 21 °C). Some samples in



(a) RBC control (b) DQN control (c) DQN+RBC control

Figure 4.11: temperature setpoint control obtained by the three approaches.

the graph also show that the temperature setpoint is set to a high value (23 °C) when the indoor temperature is less than 16 °C.

4.5 Conclusion

Guiding the DQN approach using RBC rules has a positive effect on the control policy obtained and improves the convergence towards better control faster than DQN used alone. Yet, this hybrid approach, DQN+RBC, takes a lot of time (Figure 4.8) to outperform the simple RBC approach. Moreover, during the training, some of the rules are not well visited and thus are not taken advantage of. These rules with time passing will have a low probability to be used since the parameter τ that specifies how much to exploit RBC rules will converge to zero. This problem can be solved by keeping a τ parameter for every rule and thus maintaining the rules that are not visited yet to be used for guiding the control policy in the future. This work can be investigated in future research. Overall, the hybrid approach doesn't represent an applicable control approach since it is taking a long period to outperform a simple approach such as RBC in a simple system. Hence, the investigation for an applicable control approach continues in Chapter 5.

Chapter 5

Adaptive-Network-Based Fuzzy Inference System (ANFIS) for HVAC Control

This chapter represents a continuity of Chapter 4 where the goal in this Chapter is to inject some knowledge, represented by human-interpretable rules, directly into the control policy instead of starting with a random one in order to obtain an applicable control approach. In this chapter, an adaptive-network-based fuzzy inference system called ANFIS is used as the initial policy that allows the use of human-interpretable rules at the beginning instead of starting with no knowledge. These rules correspond to the intuitive idea that a human operator may have on how to control the system. As these rules will be far from optimal and may be in some cases incorrect, a policy gradient reinforcement learning algorithm is used to train ANFIS policy to fine-tune these rules or find new ones.

This chapter starts by illustrating the definition and basic concepts of fuzzy logic. In the second part, the Mamdani fuzzy inference system with a use case scenario is presented, and then the Takagi, Sugeno, and Kang (TSK) model is introduced. In the third part, the neuro-fuzzy approach with the extension layer proposed is presented. In the final part, the office use case is presented.

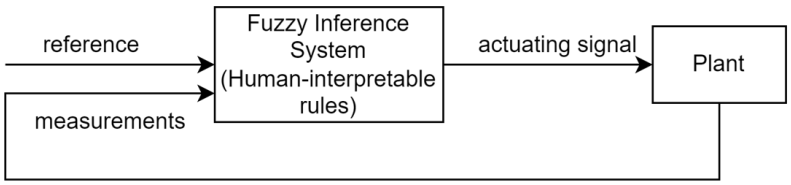


Figure 5.1: Fuzzy inference system

5.1 Fuzzy logic

The history of fuzzy logic started with the concept of fuzzy sets introduced by Zadeh [1965]. Fuzzy logic main objective is the ability to represent uncertainty and vagueness in natural language. Fuzzy logic is close to the way humans interpret things, i.e temperature is cold, warm, hot. In fuzzy logic, the truth of a statement is a degree of truth between 0 and 1 rather than a binary value 0 or 1 as in classical crisp logic. Fuzzy logic relies on a fuzzy inference system that uses human-interpretable rules to map a set of inputs to outputs. Fuzzy logic is used in control applications where reference and measurements are fed in the fuzzy inference system to generate an actuating signal as shown in figure 5.1. It can be used as well in any decision-making process such as in a banking system that decides the risk of loaning money to a person based on personal and financial information.

5.2 Basic concepts of Fuzzy Logic

5.2.1 Fuzzy Sets

A fuzzy set is an extension of classical crisp set. In classical crisp set, an element either belongs to a universe of discourse or doesn't belong. The membership of an element x, represented by $f_A(x)$ in a crisp set A is represented as follows:

$$f_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{5.1}$$

On the other hand, a fuzzy set is associated with a membership function that allows for partial membership in the interval [0, 1]. For example, in classical set, answering a question like Is this guy young will be either yes or no. However, words like young, tall,

and good are fuzzy and there doesn't exist a crisp/exact value for defining such terms. For some, age 25 is young and for others age 35 is young, and for others age 35 is adult. Hence, the word young is better to be represented as a fuzzy set that allows for partial membership and the possibility to belong to more than one fuzzy set.

5.2.2 Membership Functions

A fuzzy set is associated with a membership function MF that assigns to an element its degree of membership. There exist many shapes of membership functions such as trapezoid MF, triangular MF, gaussian MF, and bell MF.
Trapezoidal membership function is defined according to four scalar parameters a, b, c, and d as follows:

$$trapezoid(x;a,b,c,d) = \begin{cases} 0 & x \leq a \text{ or } x \geq d \\ \frac{x-a}{b-a} & a \leq x \leq b \\ 1 & b \leq x \leq c \\ \frac{d-x}{d-c} & c \leq x \leq d \end{cases} \tag{5.2}$$

Triangular MF is defined with three parameters a, b, and c as follows:

$$triangular(x;a,b,c,d) = \begin{cases} 0 & x \leq a \text{ or } x \geq c \\ \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b \leq x \leq c \end{cases} \tag{5.3}$$

Bell-shaped MF has the form of a bell. It is determined by three parameters: a that specifies the width of the bell, c that specifies the center of the bell, and b that specifies with c the slope of the curve.

$$bell(x;a,b,c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \tag{5.4}$$

Gaussian MF has a smooth curve computed by two parameters: c for determining the center and σ for determining the width of the curve.

$$gaussian(x;c,\sigma) = e^{-\frac{1}{2}\left(\frac{x-c}{\sigma}^2\right)} \tag{5.5}$$

Figure 5.2 illustrates the different membership functions that can be used with fuzzy sets. Figure 5.3 shows an example of age that can be represented using three different fuzzy sets: young, middle, and old where each is associated with a membership function. As clarified previously, fuzzy sets allow for partial membership which can be seen in the age example where 28 has the following membership value: [0.47, 0.53, 0] meaning it is 47% young-age, 53% middle-age, and 0% old-age.

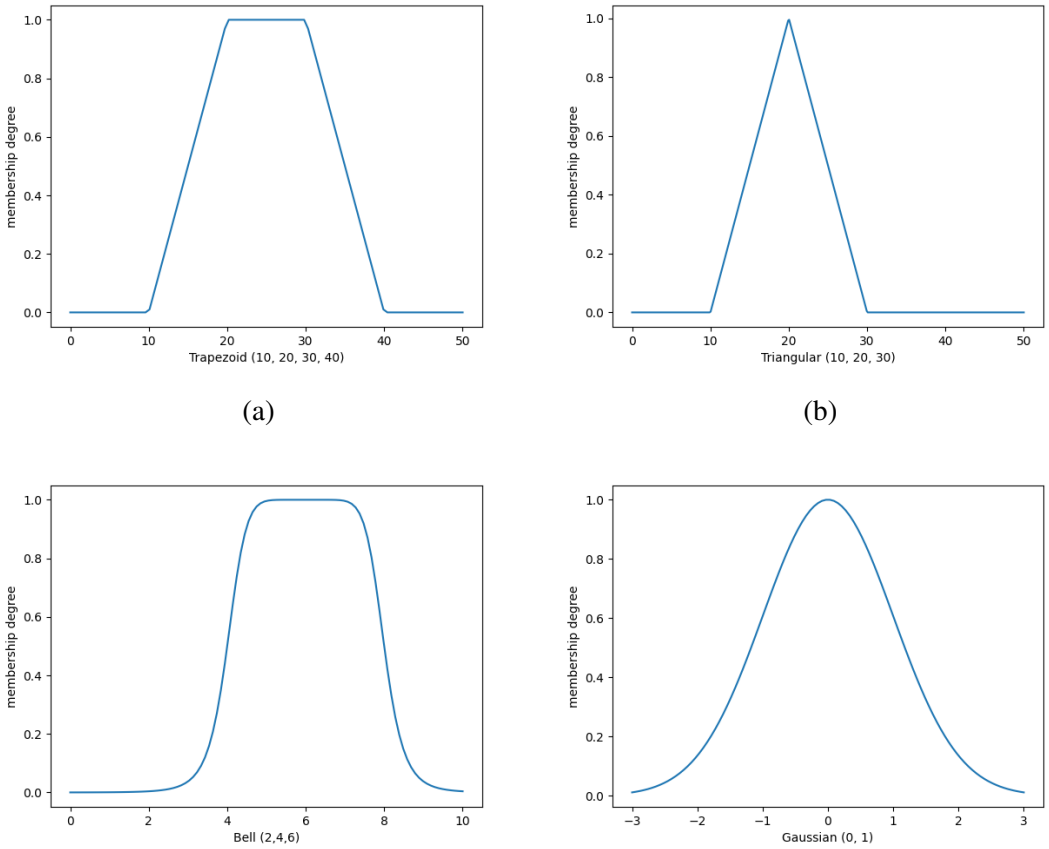


Figure 5.2: Different shapes of membership functions

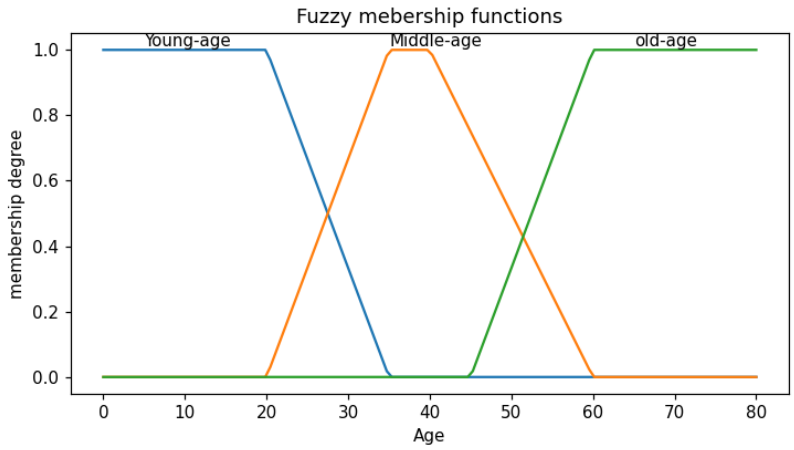


Figure 5.3: Age represented in fuzzy sets.

5.2.3 If-Then Rules

If-then rules, called fuzzy rules, specify the inference mechanism that projects the input variables into the output space. A fuzzy rule has a simple form:

$$\text{If x is A, then y is B} \tag{5.6}$$

where x is the input variable and y is the output variable. "If x is A" is called the antecedent or premise. "Then y is B" is called the consequent or conclusion. A and B are fuzzy sets, called as well linguistic values, represented by membership functions. According to membership functions, this rule will fire at a specific strength to give a crisp output. Hence, fuzzy logic allows to reason about the problem the way humans do and can show which rules are affecting the output.

5.2.4 Fuzzy Operators

Sometimes, the premise part of the rule contains many fuzzy sets combined with a fuzzy operator such as:

$$\text{if } x_1 \text{ is } A_1 \text{ AND } x_2 \text{ is } A_2, \text{ then y is } B \tag{5.7}$$

Here there are two fuzzy sets in the premise part of the rule with two input variables x_1 and x_2 associated with two membership functions A_1 and A_2 respectively. *AND* is called a fuzzy operator and is translated as follows:

$$f_{A_1 \text{ AND } A_2}(x) = min(f_{A_1}(x), f_{A_2}(x)) \tag{5.8}$$

Another common fuzzy operator is *OR* and is translated as follows:

$$f_{A_1 \text{ OR } A_2}(x) = max(f_{A_1}(x), f_{A_2}(x)) \tag{5.9}$$

5.3 Mamdani Fuzzy Inference System

Fuzzy inference system has different names: a fuzzy associative memory, a fuzzy rule-based system (FRBS), a fuzzy model, and a fuzzy controller if used for control. It is firstly introduced by Mamdani Mamdani and Assilian [1975] to control a steam engine and boiler combination using a set of linguistic control rules obtained from experts. Mamdani fuzzy inference system (MFIS) consists of four stages:

- Fuzzification of the input variables: The first stage takes the crip inputs and projects them into fuzzified inputs according to membership functions.

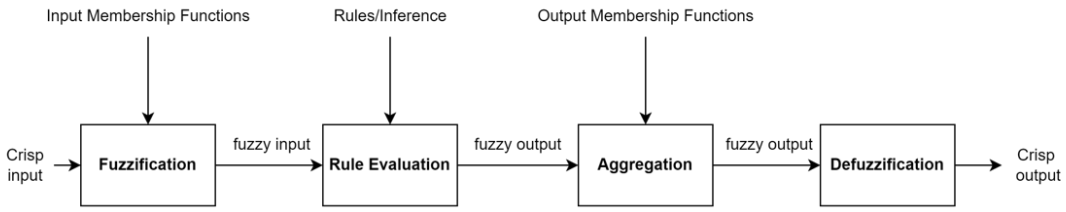


Figure 5.4: Fuzzy inference system.

- Rule evaluation: The second stage consists of taking the fuzzified inputs and applying them to the antecedents of the fuzzy rules. If a fuzzy rule has more than one antecedent, the fuzzy operator is used to obtain a single number that represents the result of the antecedent evaluation. This number is then used with the output membership functions to compute the consequent rule's output.
- Aggregation of the rules' outputs: This stage aggregates all the outputs of the previous stage and combines them into a single fuzzy set.
- Defuzzification: The last stage evaluates the rules into a crisp output.

5.3.1 A simple example to illustrate MFIS

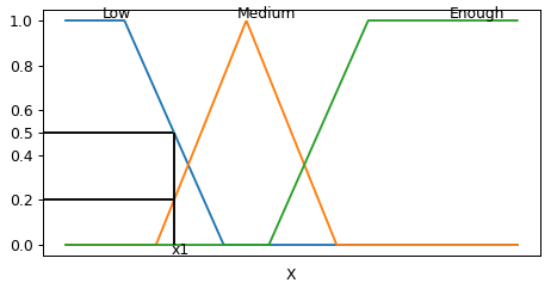
To illustrate briefly the different stages of MFIS, a simple example (Mumolo) related to risk assessment analysis is used, which computes the risk according to the project finance and the project costs (selecting, recruiting and training). There are 3 given rules:

- Rule 1: If the project finance is enough or project costs are small then risk is low.
- Rule 2: If the project finance is medium and project costs are high then risk is normal.
- Rule 3: If the project finance is low then risk is high.

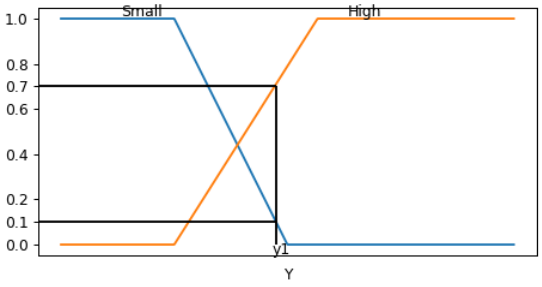
Assuming a project with finance x1 and costs y1, the four stages will be used to compute the risk associated with this project.

5.3.1.1 Step 1: Fuzzification

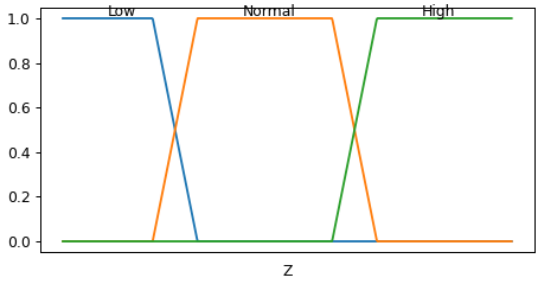
The first step is to fuzzify the crisp inputs (x1, y1) and compute their membership degrees to the corresponding fuzzy sets. To do so, inputs and output fuzzy sets should be



(a) project finance fuzzy sets



(b) project costs fuzzy sets



(c) risk fuzzy sets

Figure 5.5: Input and output fuzzy sets associated with membership functions.

determined first. There are three input fuzzy sets for project finance: low, medium, and enough. There are two input fuzzy sets for project costs: small, and high. For the output risk, there are three fuzzy sets: low, normal, and high. Input and output fuzzy sets are shown in figure 5.5. The membership degree of x1 in [low, medium, enough] equals:

$$f_{(x1=low)} = 0.5 \tag{5.10}$$

$$f_{(x1=medium)} = 0.2 \tag{5.11}$$

$$f_{(x1=enough)} = 0. \tag{5.12}$$

The membership degree of y1 in [small, high] equals:

$$f_{(y1=small)} = 0.1 \tag{5.13}$$

$$f_{(y1=high)} = 0.7 \tag{5.14}$$

5.3.1.2 Rule Evaluation

The fuzzified inputs are used with the fuzzy operator to evaluate the antecedent of the rule.

Rule number 1 evaluation:

$$f_{\text{enough OR small}}(x_1, y_1) = max(f_{enough}(x1), f_{small}(y1)) = max(0, 0.1) = 0.1 \tag{5.15}$$

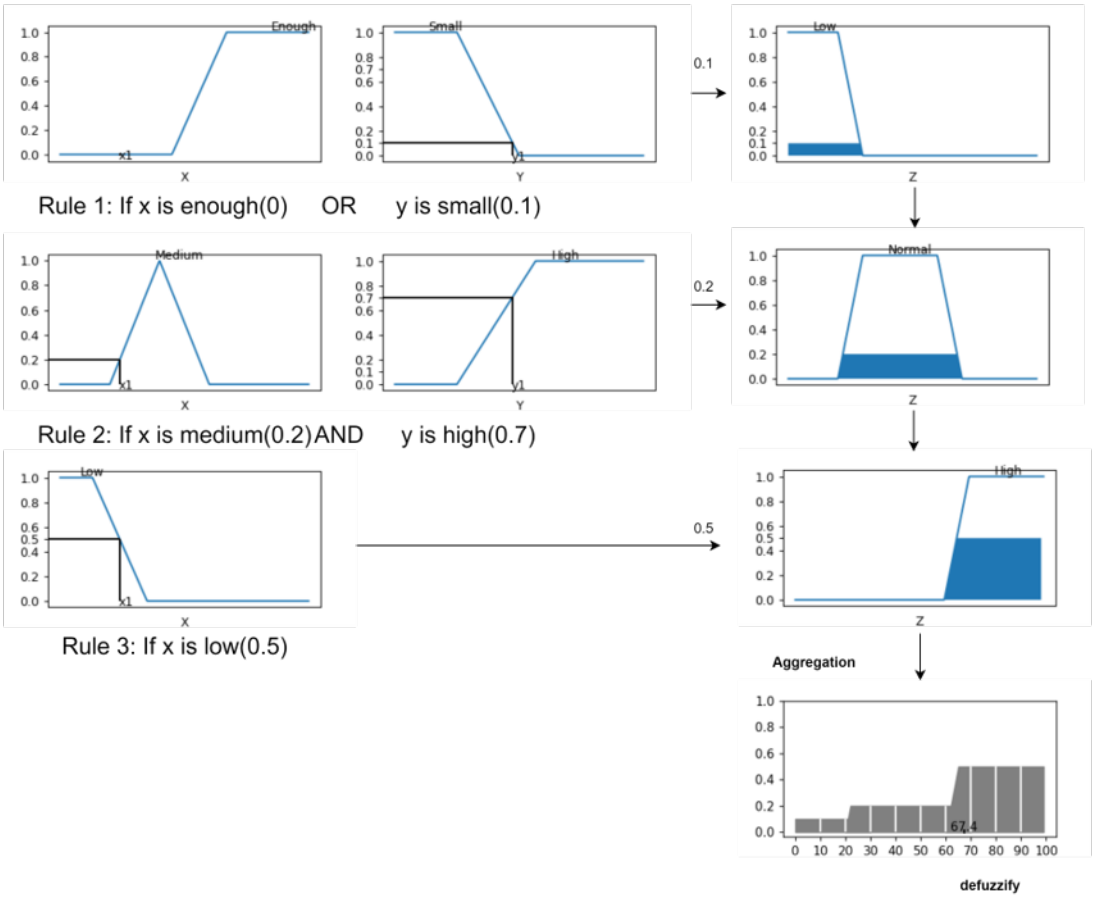


Figure 5.6: Rules evaluation, rules aggregation, and defuzzification stages.

Rule number 2 evaluation:

$$f_{\text{medium AND high}}(x_1, y_1) = min(f_{medium}(x1), f_{high}(y1)) = min(0.2, 0.7) = 0.2 \quad (5.16)$$

Rule number 3 evaluation:

$$f_{low}(x1) = 0.5 \quad (5.17)$$

These evaluations represent the corresponding firing strengths of the rules. Thus, the output (consequent part) in rule 1 will fire with a value of 0.1, and so on. The procedure is visualized in figure 5.6.

5.3.1.3 Rules' Outputs Aggregation

The decision at the end depends on all the firing strengths of the rules, obtained in the previous step and which must be combined together. Aggregation is the procedure to combine and aggregate all the firing strengths of the rules into a single fuzzy set as shown in 5.6. The Max operator is used to unify the firing strengths of rules.

5.3.1.4 Defuzzification

The last stage in MFIS is defuzzification which takes as input the aggregate fuzzy set and outputs a crisp number that represents here the risk value. Centroid defuzzification, one of the defuzzification methods, is being used to obtain the risk value given the fuzzy set (a two-dimensional shape). Centroid defuzzification outputs the center of gravity (COG) of the fuzzy set along the x-axis. The defuzzified value equals the x-coordinate of the centroid which is computed mathematically as follows:

$$\text{defuzzified value} = xCentroid = \frac{\int_z f(z) * z}{\int_z f(z)} \tag{5.18}$$

where f represents the membership function of the aggregated set. An estimate of the defuzzified value can be obtained by discretization using different sample points:

$$\text{defuzzified value} = xCentroid = \frac{\sum_i f(z_i) * z_i}{\sum_i f(z_i)} \tag{5.19}$$

In this example, the defuzzified value of risk is as follows:

$$risk = \frac{(0+10+20)*0.1+(30+40+50+60)*0.2+(70+80+90+100)*0.5}{0.1*3+0.1*4+0.5*4} = 67.4 \tag{5.20}$$

5.3.1.5 Drawbacks of MFIS

MFIS requires to integrate across a varying function to find the centroid of a two-dimensional fuzzy set. Hence, it is computationally expensive. Takagi and Sugeno [1983] introduces better rules to overcome this problem by updating the consequent part and enabling the system to learn the rule if needed. The rule in Takagi, Sugeno, and Kang (TSK) model has the following form:

if X is small and Y is large then z = ax + by + c

where X and Y are the fuzzy variables with small and large being their corresponding fuzzy sets (linguistic variables). x and y are the inputs' values associated with X and Y respectively. a, b, and c are either random parameters or initial ones set by experts. These parameters can be learned using linear regression Takagi and Sugeno [1983]. The output is polynomial with respect to x and y. The output is called a fuzzy singleton since it has a complete membership degree at a particular point. The equation can be of zero-degree order:

if X is small and Y is large then z = c

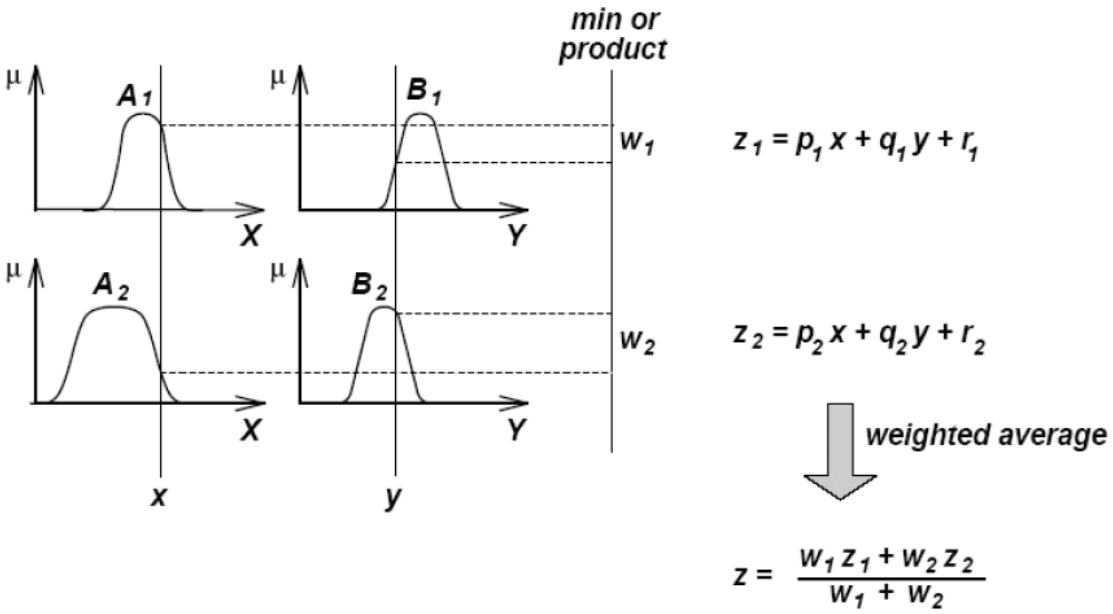


Figure 5.7: TSK fuzzy inference system.

The final output in TSK fuzzy model is computed using the weighted average:

$$z = \frac{\sum_i w_i * z_i}{\sum_i w_i} \tag{5.21}$$

where w_i, called the firing strength of the rule, is the value obtained in the antecedent evaluation of the rule and z_i is obtained using the consequent part of the rule. TSK procedure flow to compute the final output is illustrated in figure 5.7.

5.4 Neuro-fuzzy Approach

Adaptive-network-based fuzzy inference system, ANFIS Jang [1993], combines the power of fuzzy logic and neural networks. It is a neuro-fuzzy approach that architects the fuzzy inference system as a neural network and thus takes advantage of the learning power of neural nets. ANFIS is a comprehensive neural network model unlike original neural networks that are seen as black box models. It has the ability to explain the output thanks to the rules' firing strengths. Fuzzy rules used in ANFIS follow TSK fuzzy rules. ANFIS architecture consists of five layers, illustrated in figure 5.8. Square nodes contain parameters that can be tuned and thus they are adaptive nodes while circle nodes have no parameters and thus they are fixed nodes. Suppose an example with two TSK fuzzy rules of the form:

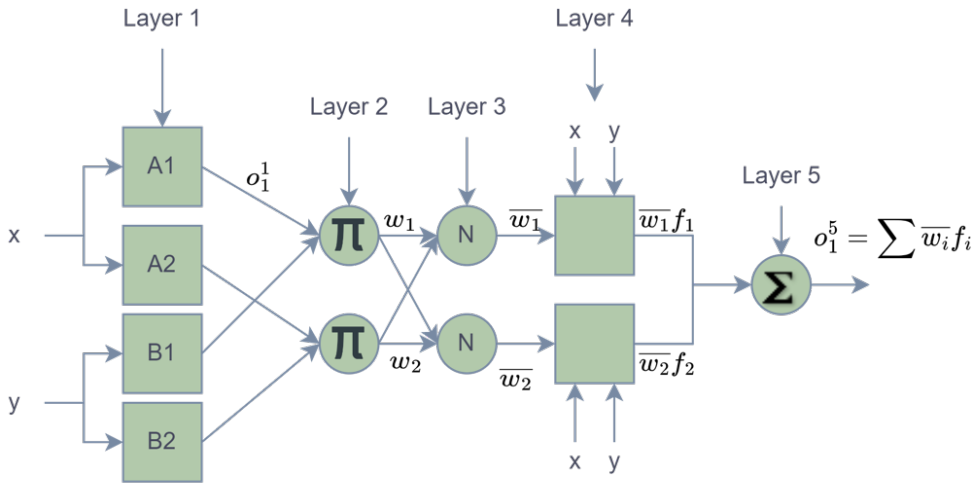


Figure 5.8: ANFIS architecture.

- Rule 1: if X is A_1 and Y is B_1 then $f_1 = p_1x + q_1y + r_1$
- Rule 2: If X is A_2 and Y is B_2 then $f_2 = p_2x + q_2y + r_2$

where f_i represents the output of the rules. μ_i will be used to represent the membership function to be consistent with the notations in the paper Jang [1993]. Considering a crip input (x, y), the layers will function as below

5.4.1 Layer 1

The nodes in this layer are adaptive and thus can be tuned during the training. The output of this layer represents the membership degree of the input with respect to the fuzzy sets used (linguistic variables: low, medium, enough). The membership function used can be bell-shaped, gaussian shaped, or any function of the ones presented previously.

$$o_i^1 = \mu_{A_i}(x) \qquad \text{for i = 1, 2} \tag{5.22}$$

$$o_i^1 = \mu_{B_{i-2}}(x) \qquad \text{for i = 3, 4} \tag{5.23}$$

These membership functions contain some parameters that represent "premise parameters" and can be updated during the training.

5.4.2 Layer 2

This layer evaluates the premise of the rule using the product (and) operator. The output specifies the firing strength of the rule and is computed as follows:

$$w_i = \mu_{A_i}(x) * \mu_{B_i}(y) \text{ , i = 1, 2} \tag{5.24}$$

During the forward propagation, specific rules will fire according to the inputs' values and then will participate in the value of output. This procedure allows to understand which rules are used for the output evaluation and thus gives an understanding of the final control unlike neural networks which is considered as a black box model.

5.4.3 Layer 3

This layer normalizes the incoming signal with respect to the sum of all rules' firing strengths:

$$\overline{w_i} = \frac{w_i}{\sum_i w_i} \tag{5.25}$$

The output is called normalized firing strength.

5.4.4 Layer 4

This layer evaluates the consequent part of the rule. The nodes in this layer are adaptive since they contain parameters that are learned during training. The output of node i in this layer is computed as follows:

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i) \tag{5.26}$$

The parameter set $\{p_i, q_i, r_i\}$ represents the consequent parameters.

5.4.5 Layer 5

This layer computes the final output of the network as a sum of all incoming signals:

$$O_1^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \tag{5.27}$$

5.5 Environment formulation using ANFIS policy

ANFIS is used as an initial control policy that takes advantage of existing knowledge encoded as fuzzy rules to start with a meaningful control policy instead of starting

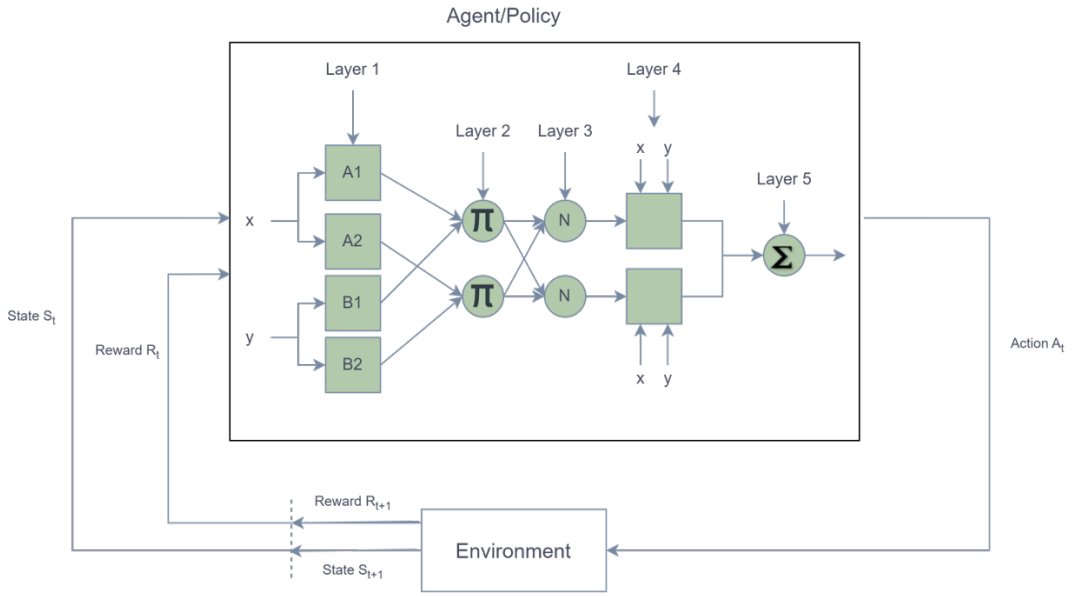


Figure 5.9: ANFIS-Environment interaction.

with a random one. The environment is formulated as a Markov decision process where the objective is to find the optimal policy that results in the highest discounted cumulative reward $G_t = \sum_t \gamma_t R(s_t, a_t)$. The ANFIS is used in the framework of model-free reinforcement learning where it represents the agent/decision-maker in the environment-agent interaction system, illustrated in figure 5.9. At every time step t, the agent (ANFIS policy) observes the state S_t that includes the reward R_t and decides what action A_t to take. The environment (office) moves then to state S_{t+1} and receives a reward R_{t+1}.

5.5.1 Learning algorithm

Since an initial policy in the form of ANFIS is provided, policy gradient is used to update the initial policy towards a better one that maximizes the discounted cumulative reward. Policy gradient is a model-free reinforcement learning algorithm that attempts to learn directly the policy without the need to learn a value function as it is the case in value-based methods. Value-based methods cannot be used since there is no information of the value of the initial policy that requires more time and data to be obtained. Reinforce is the basic policy gradient algorithm which is presented in section 2.5.2. An update is added to Reinforce where ANFIS policy is used as the initial policy instead of learning completely a random policy. The corresponding algorithm is as follows:

Algorithm 6 Reinforce

Input: a differentiable policy parameterization $\pi(a|s,\theta)$: ANFIS policy with TSK fuzzy rules

Execute one episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$ following the policy π_θ

foreach step t = 0, 1, ..., T-1 **do**

$G = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

$\theta = \theta + \alpha\gamma^t G \nabla \log \pi_\theta(A_t|S_t)$

end

Where α is the learning rate and γ is the discount factor. θ represents the consequent parameters and the antecedent parameters used in the TSK fuzzy rules. If the TSK fuzzy rules are of first-order and the membership functions used in the premise part of the rules are Gaussian then θ represent p_i, q_i, r_i and c_i, σ_i. Most commonly used TSK rules are of zero-order and then only r_i is learned in the consequent part. In addition, r_i could be initialized with a good value according to the expert knowledge. As a result, instead of starting from somewhere very random and bad in the control space, the starting point is somewhere near the optimal control policy which leads to faster convergence. Even if the first-order TSK rules are used with completely random parameters, the ANFIS agent will continue to be better than a pure neural network agent since, after being trained, it includes the comprehension feature that allows to explain the result much better.

5.5.1.1 Catastrophic forgetting

The original version of Reinforce uses a classical perceptron like neural network as input instead of the ANFIS neural policy. With the policy gradient that tries to learn online and doesn't use historical data, the policy gradient is prone to catastrophic forgetting of the neural network parameters especially with limited data or time. The learning is continual and occurs day by day meaning that the data of a specific day are used to update the policy at the end of the day without any use of historical data of other days. As a result, when finishing the learning during the winter season and moving to the summer season, the policy gradient will deal with contextual data related to summer only and after some time it will forget what has been learned during the winter. However, with the updated version of reinforce, the policy is an ANFIS model. During training, only a small part of the rules will fire at some strengths and others will not be activated at all. Hence, the update of the ANFIS model is local at some rules which is much less prone to catastrophic forgetting. To elaborate, the rules that will be fired during the winter are

different than the ones that will be fired during the summer and thus the learning during the summer will not affect catastrophically what has been learned during the winter.

5.5.2 Extension Layer

The example illustrated with ANFIS contains only two fuzzy rules. Considering zero-order TSK fuzzy rules initialized by the expert, the rules have the following form:

- Rule 1: if x is A_1 and y is B_1 then $f_1 = r_1$
- Rule 2: If x is A_2 and y is B_2 then $f_2 = r_2$

Zero-order rules are considered because they are easier to be given by the expert and to interpret than the first-order rules where the consequent part is a linear combination of the inputs. Thus, the policy gradient will finally tune the consequent parameters for these two rules only r_1 and r_2. Most problems consist of a high number of inputs associated with many membership functions and thus the number of rules provided by the expert will be very small which limits the solution space. Here comes the importance of the extension layer that we propose that can generate additional rules that represent all the possible combinations of the premise parts with default values for the consequent parts as illustrated in figure 5.10.

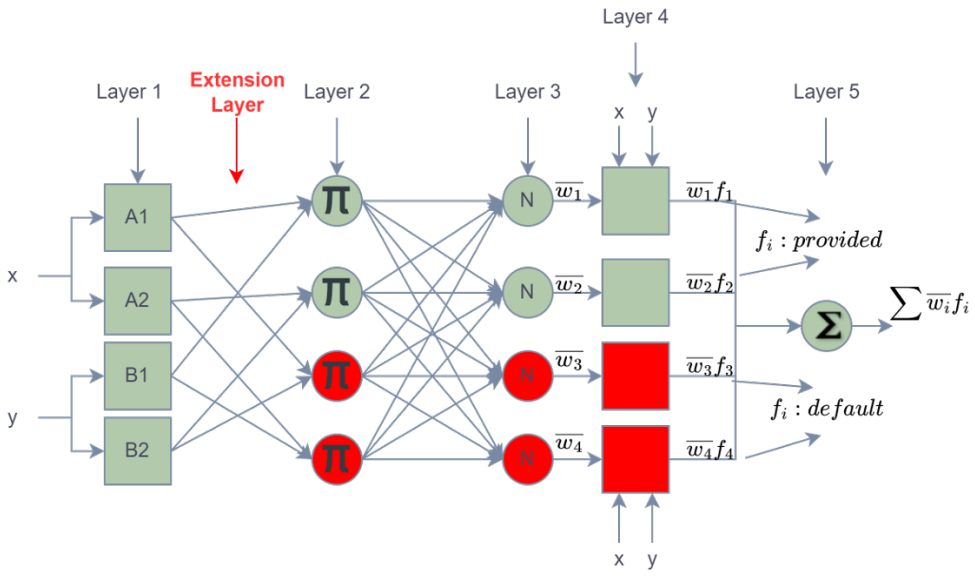


Figure 5.10: ANFIS with extension layer: the rules with red color are generated by the extension layer.

In addition, the initial form of TSK fuzzy rules requires writing all the inputs in the premise part which turns into a complex rule to be provided by the expert when the number of inputs increases. Therefore, an expert should be able to provide easier rules that include in the premise part the relations between some of the input variables and not all of them especially when there are many input variables. Such fuzzy rules can have a general form as the following:

- Rule 1: if x is A_1 then $f = r$

The extension layer will then extend this type of fuzzy rules to include all the input variables in the premise part and thus extend it to two fuzzy rules:

- Rule 1: if x is A_1 and y is B_1 then $f_1 = r_1 = r$
- Rule 2: If x is A_1 and y is B_2 then $f_2 = r_2 = r$

In some problems, there can be more than one output where the rules can have the following the form

- Rule 1: if x is A_1 then $f = r1$ and $g = c1$
- Rule 2: if y is B_1 then $f = r2$ and $g = c2$

With the extension layer, these rules will be converted to the following fuzzy rules:

- Rule 1: if x is A_1 and y is B_1 then $f = r1$ and $g = c1$
- Rule 2: if x is A_1 and y is B_2 then $f = r1$ and $g = c1$
- Rule 3: if x is A_1 and y is B_1 then $f = r2$ and $g = c2$
- Rule 4: if x is A_2 and y is B_1 then $f = r2$ and $g = c2$

Note that there is a collision after extending the initial rules between R1 and R3 since they have the same premise part. Thus, one rule can only exist and the choice of choosing R3 is preferred over R1 depending on the last rule generated. Another feature called "neutral" can be added to the rule to:

- Specify what values not to modify during a collision.

For example, in the previous example, rule 3 can have the following form:

- Rule 3: if x is A_1 and y is B_1 then $f = r2$ and $g = neutral$

which means to override the value of f in Rule 1 and keep the same value of g as in R1.

5.6 Study Case

The office formulation is well elaborated in section 4.1 but it will be presented again to remind of the objective to be solved and the modification with the previous formulation. The office optimization problem is formulated as a Markov decision process as illustrated in figure 5.9 with the extended ANFIS policy (figure 5.10). At each timestep t and after taking an action a_t, the agent represented by the extended ANFIS policy will receive an immediate reward R_t: s as follows:

$$R_t = \lambda_1 * R_T(t) + \lambda_2 * R_C(t) + \lambda_3 * R_P(t) \tag{5.28}$$

$$R_T(t) = \begin{cases} e^{-(T_t - \underline{T}_{comfort})^2} - [T_t - \underline{T}_{comfort}]_+ & \text{if } T_t < \underline{T}_{comfort} \\ e^{-(T_t - \overline{T}_{comfort})^2} - [T_t - \overline{T}_{comfort}]_+ & \text{if } T_t > \overline{T}_{comfort} \\ 1 & \text{if } T_t \text{ is between } [\underline{T}_{comfort}, \overline{T}_{comfort}] \end{cases} \tag{5.29}$$

$$R_C(t) = \min(0, (500 - C)/(1500 - 500) \tag{5.30}$$

$$R_P(t) = -P(t) * cost(kwh) \tag{5.31}$$

λ_1, λ_2, and λ_3 specify the importance given to each feature during the training. , C and P represent indoor temperature, indoor CO_2 concentration, and power consumption of the radiator respectively. $[\underline{T}_{comfort}, \overline{T}_{comfort}]$ represents the comfort range of the temperature where it is in the range of [21°C, 23°C] during the occupied time from 8 am until 6 pm and in the range of [13°C, 30°C] otherwise. The objective is to maximize the cumulative reward in the long term with $G_t = \sum_t \gamma_t R(s_t, a_t)$ where the episode represents one day. The cumulative reward represents minimizing energy consumption while maintaining thermal comfort and good indoor CO2 concentration in the long term. The extended ANFIS policy will be updated using the policy gradient algorithm, check Reinforce algorithm 6.

5.6.1 Fuzzy Rules

There are 13 rules provided which are the following:

- Rule 1: If the CO2 is high, then window is open, door is neutral, and temperature setpoint is neutral.

- Rule 2: If the indoor temperature is cold and the corridor temperature is normal, then window is neutral, door is open, and temperature setpoint is comfort setpoint temperature.
- Rule 3: If the indoor temperature is cold and the corridor temperature is hot, then window is neutral, door is open, and temperature setpoint is comfort setpoint temperature.
- Rule 4: If the indoor temperature is hot and the corridor temperature is normal, then window is neutral, door is open, and temperature setpoint is neutral.
- Rule 5: If the indoor temperature is hot and the corridor temperature is cold, then window is neutral, door is open, and temperature setpoint is neutral.
- Rule 6: If the indoor temperature is cold and the outdoor temperature is normal, then window is open, door is neutral, and temperature setpoint is comfort setpoint temperature.
- Rule 7: If the indoor temperature is cold and the outdoor temperature is hot, then window is open, door is neutral, and temperature is comfort setpoint temperature.
- Rule 8: If the indoor temperature is hot and the outdoor temperature is normal, then window is open, door is neutral, and temperature setpoint is neutral.
- Rule 9: If indoor temperature is hot and outdoor temperature is cold, then window is open, door is neutral, and temperature setpoint is neutral.
- Rule 10: If the outdoor temperature is cold, the indoor CO2 concentration is high, and the corridor CO2 concentration is low, then window is open , door is open, and temperature setpoint is neutral. (window should be closed)
- Rule 11: If the day is not a working day, then window is closed, door is closed, and temperature setpoint is at lowest.
- Rule 12: If it is early morning, then window is closed, door is closed, and temperature setpoint is at lowest.
- Rule 13: If it is late evening, then window is closed, door is closed, and temperature setpoint is at lowest.

Lowest and comfort setpoint temperatures correspond to 13 and 21 °C respectively. 0 represents closed and 1 represents open. These rules are encoded into zero-order TSK

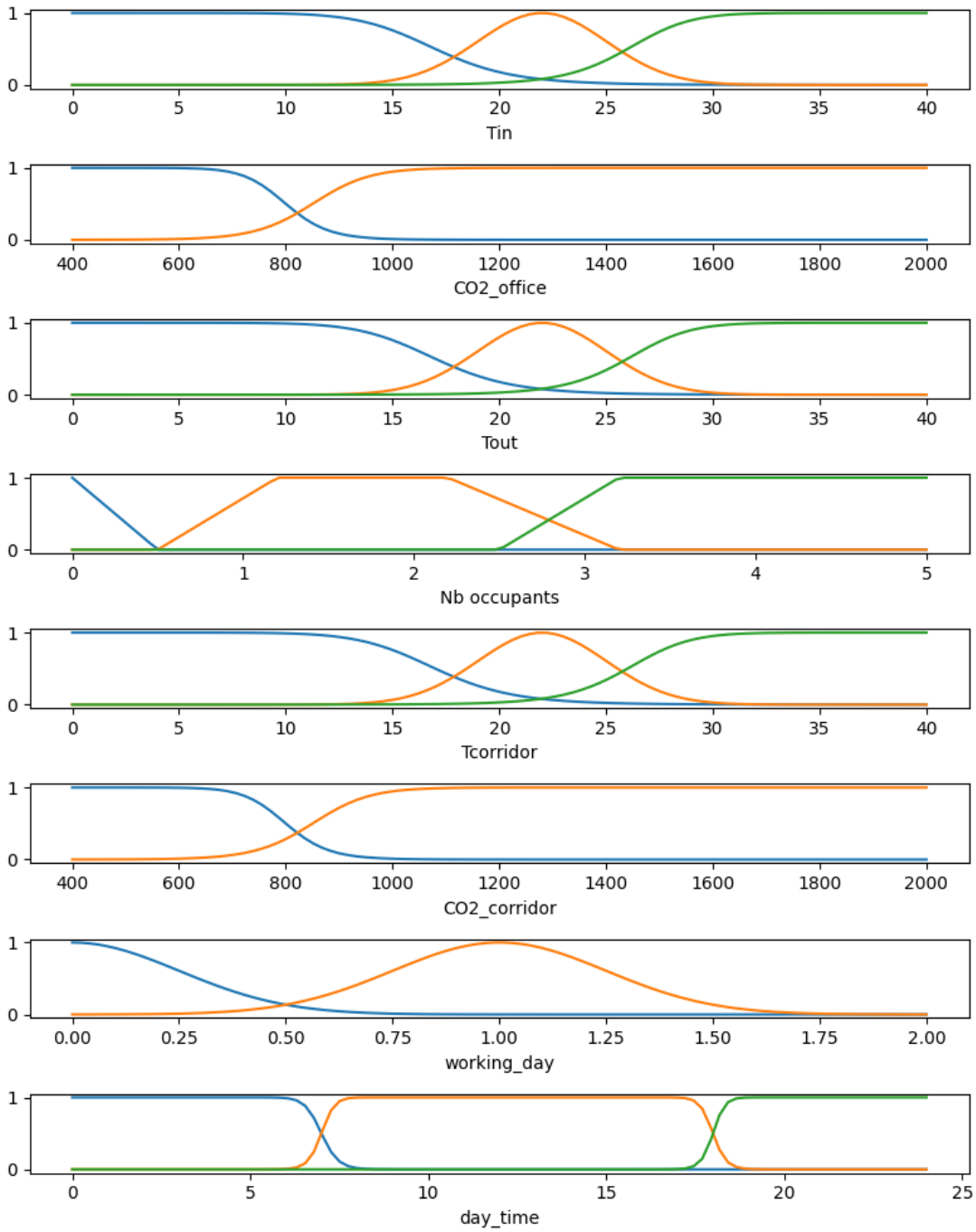


Figure 5.11: Membership functions associated with every state feature.

fuzzy rules using fuzzy sets (Linguistic variables) associated with membership functions. There are eight features used in the system state:

- indoor temperature (Tin: x1): cold (A1), normal (A2), hot(A3).
- office CO2 concentration (CO2_office: x2): low (B1), high (B2).

- outdoor temperature (Tout: x3); cold (C1), normal (C2), hot (C3)
- number of occupants (Nb ocupants: x4): none (D1), medium (D2), high (D3).
- corridor temperature (Tcorridor: x5): cold (E1), normal (E2), hot (E3).
- corridor CO2 concentration ((CO2_corridor: x6): low (F1), high (F2).
- working_day (x7): working day (G1), weekend day (G2).
- day_time (x8): early morning (H1), working_time (H2), late evening (H3).

Membership functions associated with every feature are illustrated in figure 5.11. zero-order TSK fuzzy rules are used in the ANFIS policy and thus the consequent part is a constant number. The TSK fuzzy rules associated with the above rules will be as follow:

- Rule 1: If x2 is B2 then z = [1, neutral, neutral][1].
- Rule 2: If x1 is A1 and x5 is E2 then z = [neutral, 1, comfort_setpoint].
- ...
- Rule 12: if x8 is H1, then z = [0, 0, lowest_setpoint].
- Rule 13: if x8 is H3, then z = [0, 0, lowest_setpoint].

These fuzzy rules are injected into the ANFIS policy that is used as an initial controller. The consequent part has the form $[c_{i1}, c_{i2}, c_{i3}]$ that corresponds to the constants window, door, and temperature setpoint respectively of rule number i. The learning algorithm (Reinforce algorithm 6) is trying to learn the consequent parameters (constants) of all rules in the policy.

5.6.2 Rules' Generation

With the extension layer added to the ANFIS architecture, there will be 1944 fuzzy rules that represent all possible combinations of premise parts. The generated fuzzy rules that don't rely on any information from the provided rules will have default actions which are [0.5 for window, 0.5 for door, 19 °C for temperature setpoint] where 0.5 means half-time (of one hour) open.

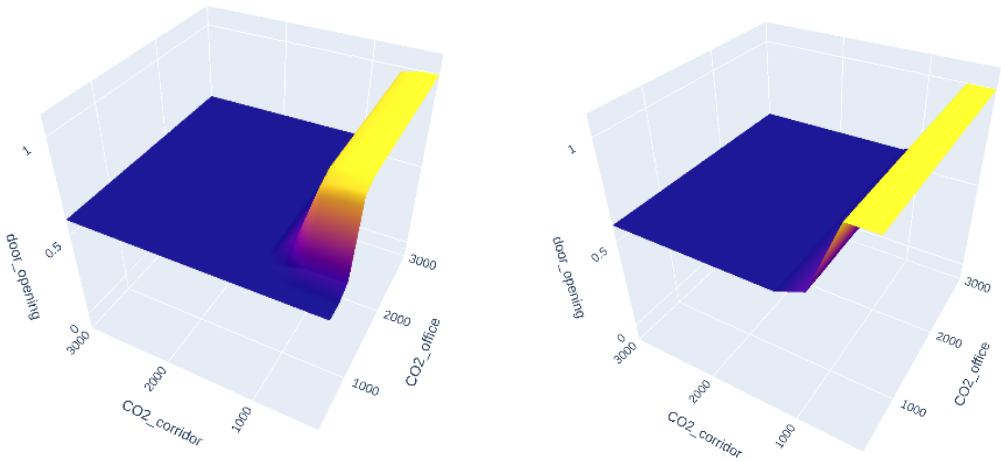
[1] Output is *window*, *door* and *temperature setpoint*

5.6.3 Initial vs Trained ANFIS Controller visualization

ANFIS policy is used as the initial control policy with the given fuzzy rules that are used to generate other rules. ANFIS policy is then trained for 15 days and compared with the initial control policy with respect to the different controls and in different observations:

5.6.3.1 Door Control

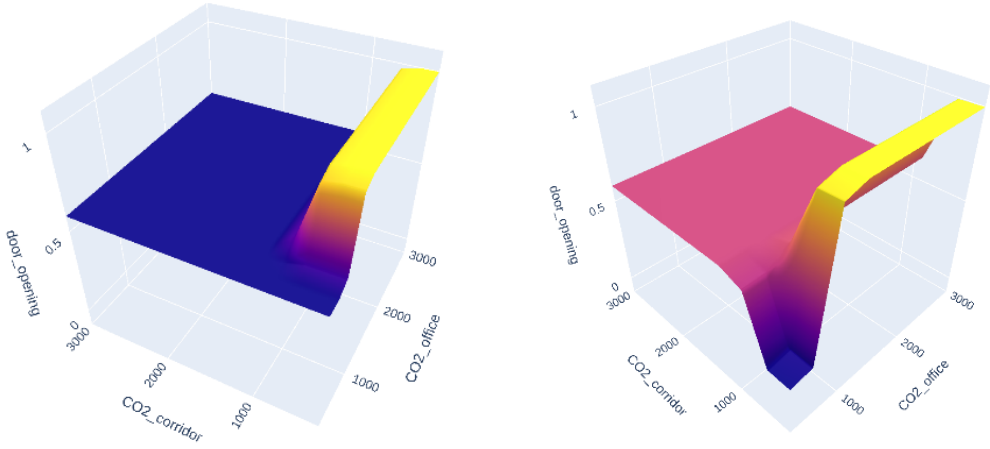
- Office occupied: Door control is affected greatly by the CO2 inside the office and the CO2 in the corridor. Note that the other state's values affect as well the value of the door control especially the number of occupants and corridor temperature. Taking the following observation: [indoor temperature: 20 °C, outdoor temperature: 10 °C, number of occupants: 1, corridor temperature: 20 °C, working day 1, time: 10 am], the door control is visualized with different values of CO2 office and CO2 corridor.
  The initial policy opens completely the door when the CO2 in the office is high and the CO2 in the corridor is low. Otherwise, the door is 60 % of the time open. The ANFIS policy tends to open completely the door after 15 days (figure 5.12 b) whenever the CO2 in the corridor has a low range which prevents the CO2 inside the office to increase (there is one occupant in the office). The initial surface that represents the door results with high corridor CO2 concentration didn't even change with the trained policy and this goes back to the reason that observations



(a) Initial policy.

(b) Trained policy.

Figure 5.12: The door control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b) given that the office is occupied.



(a) Initial policy.

(b) Trained policy.

Figure 5.13: The door control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b) given that the office is unoccupied.

with high corridor CO2 concentration weren't met during the training time.

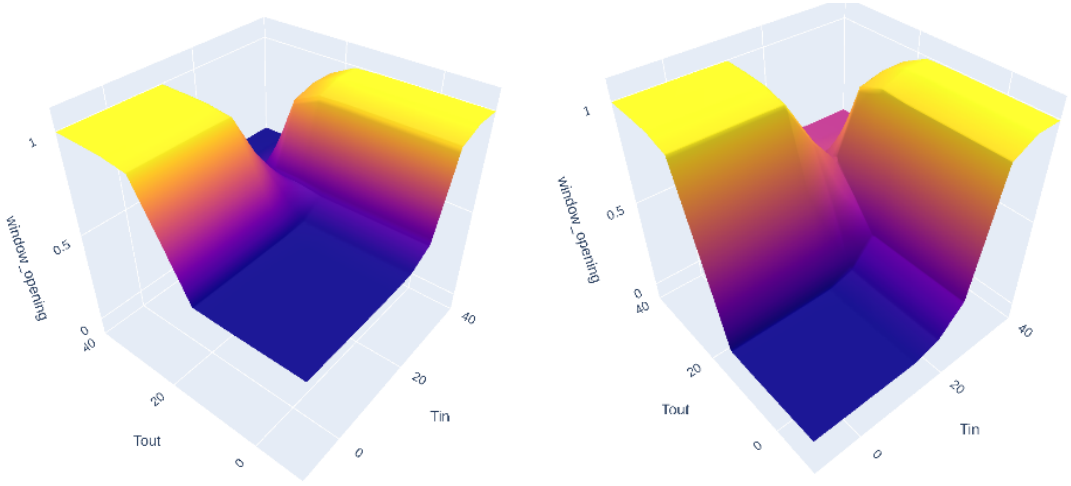
- Office unoccupied: Changing only the number of occupants in the observation from 1 to 0 (figure 5.13) will lead to another control policy. The trained policy tends to close the door when the CO2 inside the office is low even if the CO2 in the corridor is low since the CO2 office will not increase if there are no occupants inside the office.

5.6.3.2 Window control

- Low CO2 concentration: Considering the following state vector [office CO2: 400 ppm, number of occupants: 1, corridor temperature: 10 °C, corridor CO2: 400 ppm, working day: 1, time: 10 am], the window control with respect to indoor and outdoor temperature is illustrated in figure 5.14. To elaborate, the window initially is completely open when the indoor temperature is low and the outdoor temperature is high in order to heat the room. It is as well open when the indoor temperature is high and the outdoor temperature is low in order to cool the room. Otherwise, the window is initialized to be open half of the time. With the trained policy, the modification is made when the indoor temperature is low and the outdoor temperature is low where the window is almost closed in order to prevent the office temperature from decreasing. The surface that represents the window results with high indoor and outdoor temperatures stayed intact since observations

with high indoor and outdoor temperatures were not met during training time (no catastrophic forgetting).

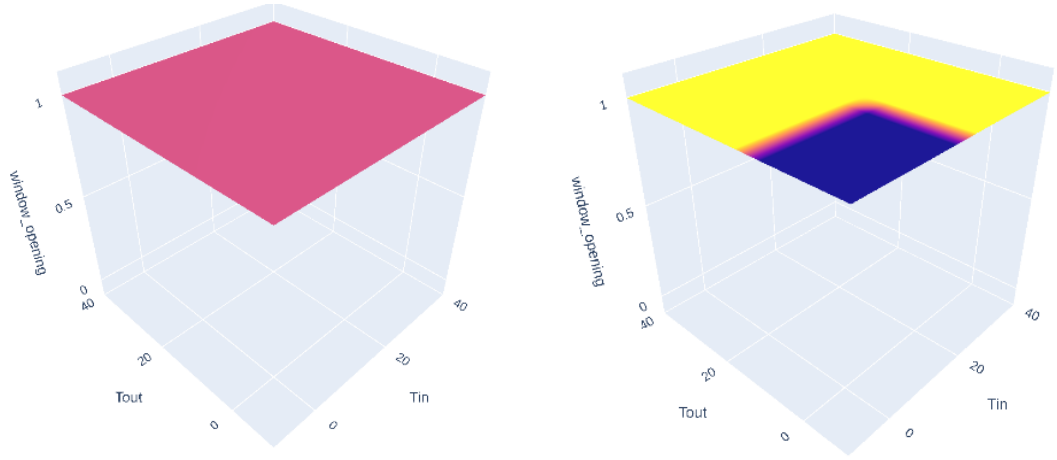
- High CO2 concentration: Increasing the CO2 concentration in the office and corridor until 1200 ppm will lead to completely different results. The initial and trained policy will tend to completely open the window (figure 5.15 regardless of the indoor and outdoor temperature in order to decrease the indoor CO2 concentration.



(a) Initial policy.

(b) Trained policy.

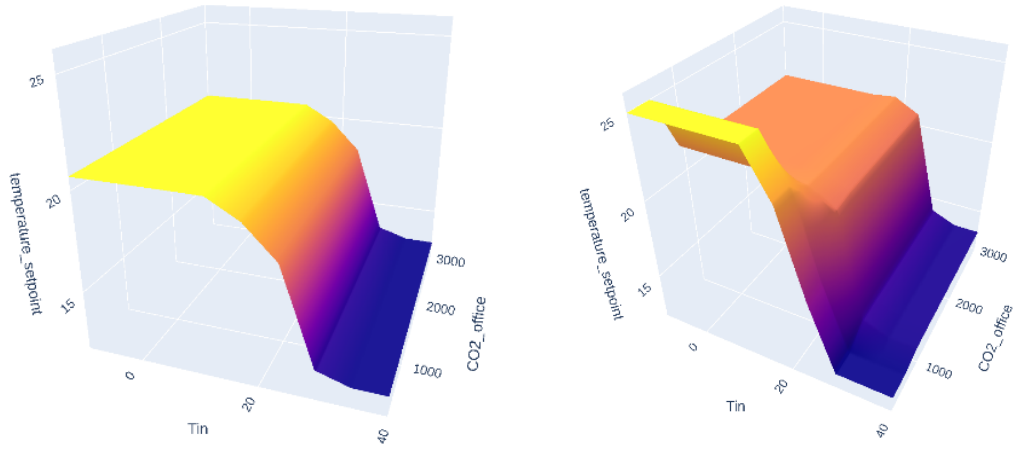
Figure 5.14: The window control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b).



(a) Initial policy.

(b) Trained policy.

Figure 5.15: The window control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b) with high CO2 concentration in the office and the corridor.



(a) Initial policy.

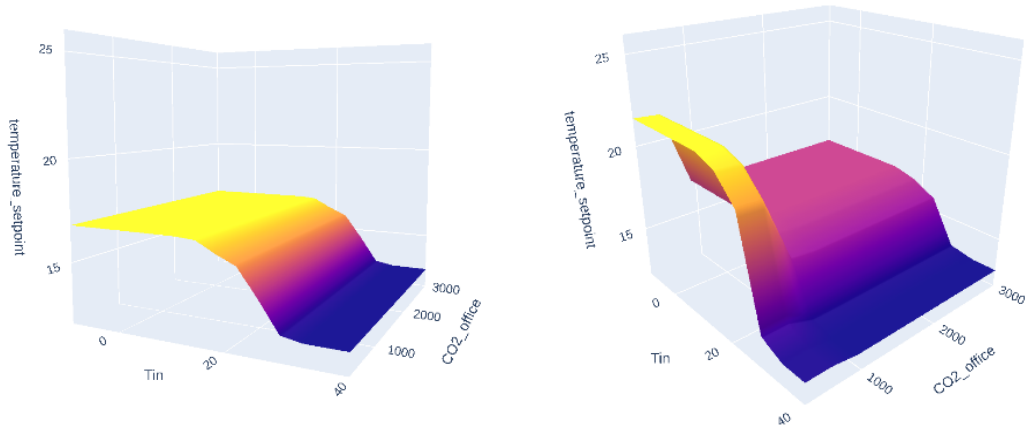
(b) Trained policy.

Figure 5.16: The temperature setpoint control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b).

5.6.3.3 Temperature setpoint control

- Working Time: Considering the following observation [outdoor temperature: 10 °C, number of occupants: 1, corridor temperature: 20 °C, corridor CO2: 400 ppm, working day: 1, time: 10 am], the initial control policy, figure 5.16 a, sets the temperature setpoint at lowest (13 °C) when the indoor temperature is high. Otherwise, the temperature setpoint is initialized around 20.6 °C. With the trained policy, figure 5.16 b, the temperature setpoint stayed almost the same when the indoor temperature is high. However, when the indoor temperature is low and the CO2 office is low, the temperature setpoint is set at maximum (25 °C) greater than the case with the initial control policy in order to achieve the comfort temperature rapidly. When the indoor temperature is low and the CO2 office is high, the setpoint temperature is increased up to 21.4 °C.

- Early morning (includes the time from 12 midnight until 7 am): Considering the previous observation but with time equals 7 am which is 1 hour before the office is occupied and thus the number of occupants equals zero. The initial control policy, figure 5.17 a, sets the temperature setpoint around 16 °C where the indoor temperature is cold. With the trained control policy, figure 5.17 b, the temperature setpoint is increased until around 21.5 °C when the indoor temperature is cold and the CO2 office is low. This functionality will allow preheating the office in the morning a while before the occupants arrive. The other surfaces that stayed almost



(a) Initial policy.

(b) Trained policy.

Figure 5.17: The temperature setpoint control obtained by the initial ANFIS policy (figure a) and the trained policy (figure b) 1 hour before occupation.

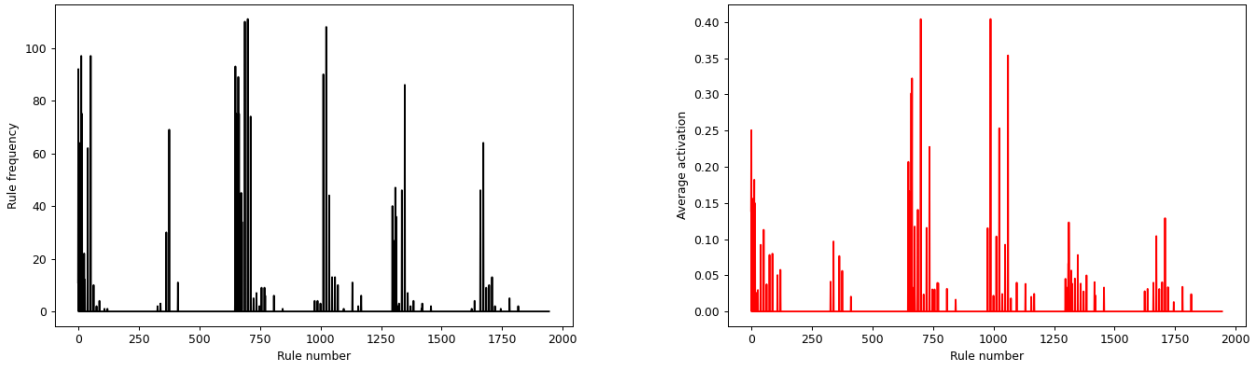
intact are not visited during the training.

5.7 Analysis of Rules before and after training

The results presented before used a trained policy for 15 days and they were able to show that good control has been learned in a short interval of time. In this and the coming sections, the trained policy benefits from one month of training to gather more data and obtain a better control policy. The advantage of ANFIS policy is in the ability to clearly explain the result and thus can be called a comprehensive neural network. Every output can be explained and traced back to the activation of the premise and consequent parts of the rules. The ANFIS policy trained in the study case consists of 1944 rules which represent all possible combinations of the membership functions of the state features.

5.7.1 Rules frequency versus activation

It is important to keep track of the rules met during the training and how much every rule is activated. This allows explaining why some rules have or haven't changed. To do so, every rule has a frequency and an activation. The frequency represents how much a rule is visited and the activation represents how much a rule is activated on average



(a) The frequency of every rule.

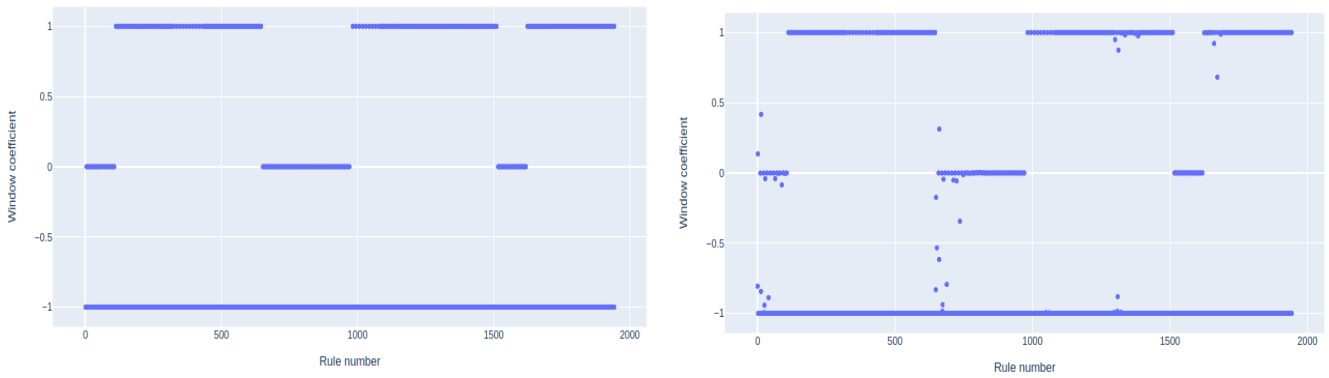
(b) The average activation of every rule.

Figure 5.18: The frequency and activation of rules after training of 1 month.

during the training. If a rule has a frequency and activation equal to zero, it means that no observations which allow this rule to fire (activate) have been met during the training.

5.7.2 Rules' consequent coefficients

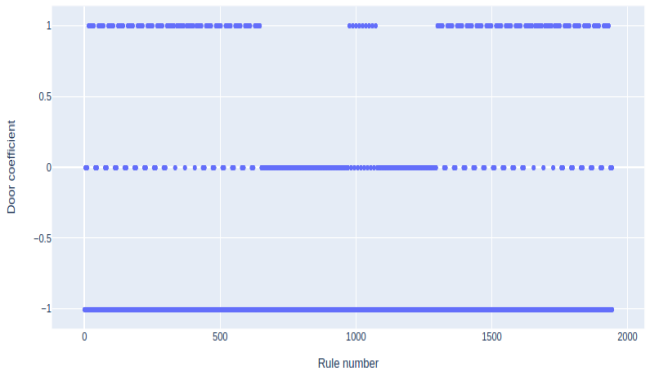
Every rule has an antecedent part and a consequent part. The consequent part consists of the coefficients that represent the window, door, and temperature setpoint controls. These coefficients are the parameters of every rule that the training algorithm is trying to learn. In this section, the coefficients of every control will be presented to show how rules are updated with the trained policy. Figures 5.19, 5.20, and 5.21 show the coefficients of window, door, and temperature setpoint controls respectively obtained by the initial



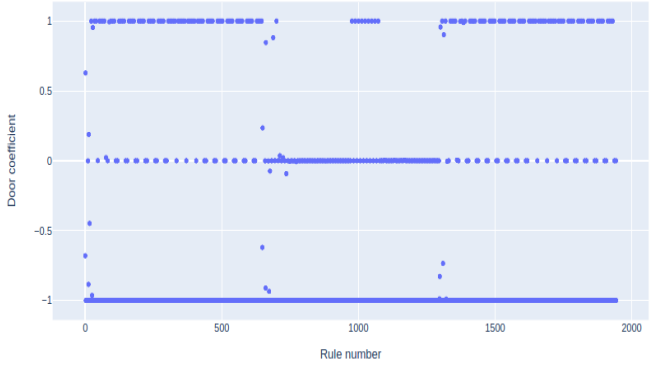
(a) The initial consequent coefficients of the window.

(b) The trained consequent coefficients of the window

Figure 5.19: The window coefficients of every rule obtained by the initial ANFIS policy (a) and the ANFIS trained policy (b).

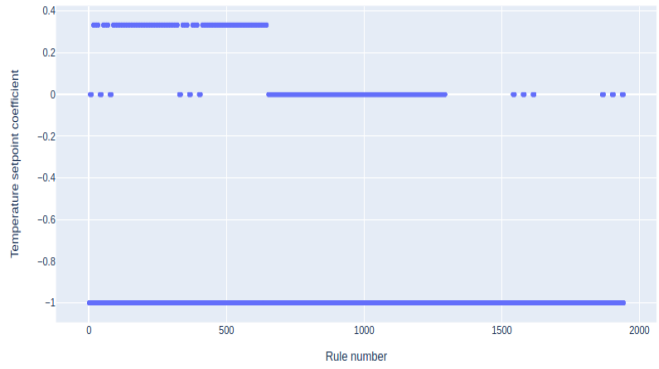


(a) The initial consequent coefficients of the door.

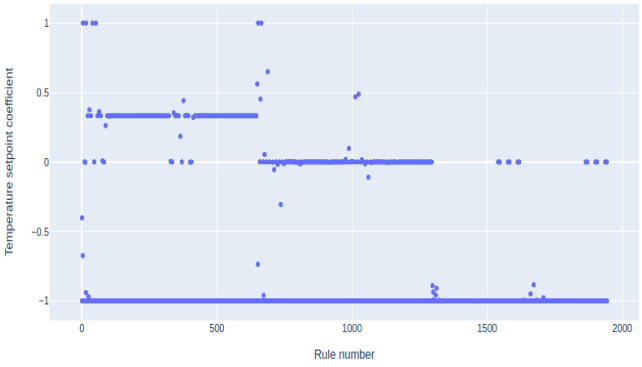


(b) The trained consequent coefficients of the door.

Figure 5.20: The door coefficients of every rule obtained by the initial ANFIS policy (a) and the ANFIS trained policy (b).



(a) The initial consequent coefficients of the temperature setpoint.



(b) The trained consequent coefficients of the temperature setpoint.

Figure 5.21: The temperature septoint coefficients of every rule obtained by the initial ANFIS policy (a) and the ANFIS trained policy (b).

ANFIS policy and the trained ANFIS policy. Some rules have stayed with the same coefficients for all the controls and this is because of 2 reasons: First reason is that observations that activate these rules weren't met during the training and thus consequent coefficients of these rules were not updated (figure 5.18). The second reason is that the trained policy has reached the same result as the initial policy concerning these rules.

Rule number 52 in the initial policy can be written as follows (the policy is called every hour):
IF Tin is cold and CO2_office is low and Tout is cold and Nb occupants is medium and Tcorridor is normal and CO2_corridor is low and working_day is working day and

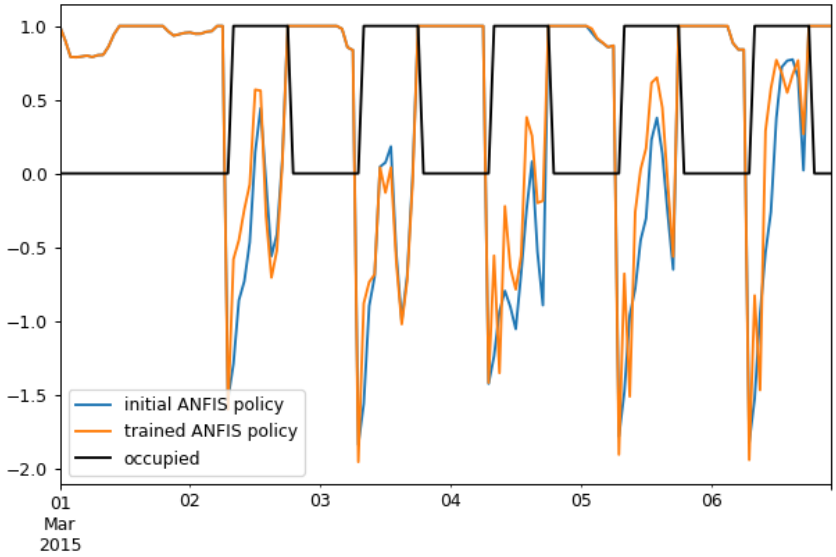
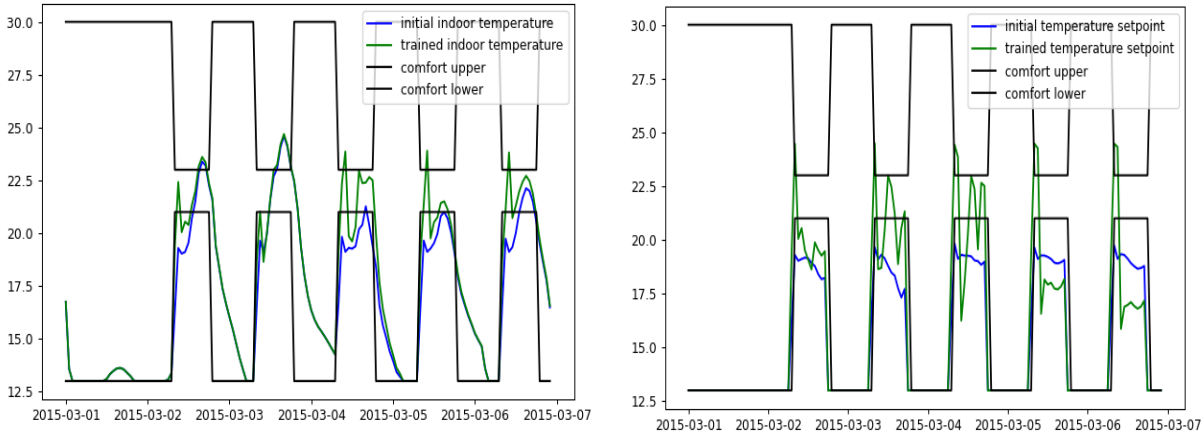


Figure 5.22: The reward obtained by the initial ANFIS policy vs the trained one.

day_time is working time **THEN** [window: 0.0, door: 1.0, temperature setpoint: 0.333]. The actions here are scaled. After rescaling to the real values, window is 0.5 (half-time open), window: 1 (open all the hour), temperature setpoint: 21 °C.
Rule number 52 in the trained policy has been updated to the following:
IF Tin is cold and CO2_office is low and Tout is cold and Nb occupants is medium and Tcorridor is normal and CO2_corridor is low and working_day is working day and day_time is working time **THEN** [window: - 1.0, door: 1.0, temperature setpoint: 1.0]. The actions here are scaled. After rescaling to the real values, window is 0 (closed), window: 1 (open), temperature setpoint: 25 °C.
The trained policy has learned with the state of rule number 52 to close the window and set the temperature setpoint at maximum in order to reach thermal comfort faster inside the office. The result can be well explained and thus the policy is comprehensive.

5.7.3 Metric performance

The performance is measured by the reward that represents a weighted sum of thermal reward, air quality reward, and power reward. The policies are tested in the first week of March. Figure 5.22 shows that the trained ANFIS policy outperforms the initial policy most of the time. To understand more about the performance of the ANFIS policy, detailed results are shown in the following section.



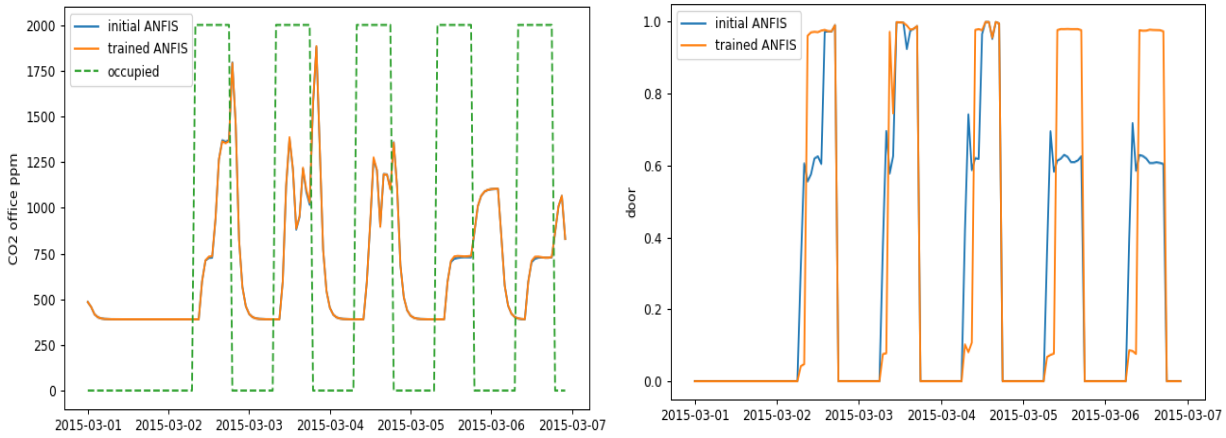
(a) Temperature results.

(b) Temperature setpoint results.

Figure 5.23: Temperature (left) and temperature setpoint results (b) obtained after testing the initial and the trained ANFIS policies over the first week of March.

5.7.4 Temperature, CO2 and power results

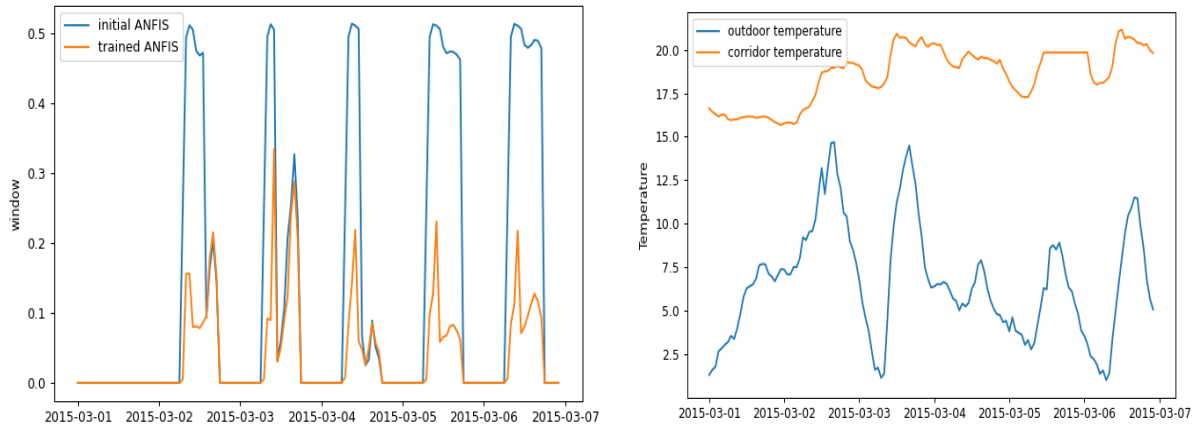
Figure 5.23 shows that the trained ANFIS policy has improved thermal comfort. It activates highly the temperature setpoint (23 °C) in advance during the morning in order to reach thermal comfort faster. However, it overshoots the comfort range sometimes. Concerning the CO2 concentration inside the office, the performance of the trained ANFIS policy is very close to the initial ANFIS policy (Figure 5.24 a). To understand more about the CO2 results, the door and window controls are illustrated in figures 5.24 (a) and 5.25 (a) as well. At 8 am, the updated policy (trained) keeps the door and window near zero in order to allow the office to reach thermal comfort faster in the morning. In addition, it tends to open more the door in order to maintain good thermal comfort since the corridor temperature has a better effect on the office temperature than the outdoor temperature (figure 5.25 b). The fact that the door is opened more with the updated policy than the initial policy should improve the CO2 quality inside the office since the CO2 concentration in the corridor is very low; however, the updated policy decreases at the same time the opening of the window which leads to a neutral effect on the CO2 concentration inside the office. As a result, the trained ANFIS policy in this scenario shows better improvements in thermal comfort but a neutral effect on the air quality. Concerning power consumption, the trained policy consumes more than the initial policy since it activates more the radiator in order to reach thermal comfort. In the case of the initial policy, the radiator consumes nearly 20 KWh over one week whereas it consumes 25 KWh in the case of the trained policy.



(a) CO2 office.

(b) Door control.

Figure 5.24: On the left, CO2 office obtained by the initial and the trained ANFIS policies on the first week of March. On the right, the door control is computed by the two policies: initial and trained ANFIS.



(a) Window control.

(b) Outdoor vs corridor temperature.

Figure 5.25: On the left, the window control is computed by the two policies: initial and trained ANFIS. On the right, the temperatures of the corridor and outdoor are shown to understand the functionality of the updated policy.

5.7.5 Conclusion

In this chapter, an advanced approach that combines the power of the neural network and fuzzy logic, ANFIS with the extension layer added, has been developed and studied to optimize energy consumption while maintaining thermal comfort and good air quality inside the office. This approach has been trained using policy gradient algorithm and

has shown good results in a short interval of training for 15 days. The ANFIS approach trained using policy gradient represents an applicable control approach for the office's system while maintaining thermal comfort and good indoor air quality inside the office. Using ANFIS, a policy with initial fuzzy rules injected is used at the beginning unlike the DQN+RBC presented in Chapter 4 which starts with a random policy and relies on the guidance given by the exploiting of RBC rules during the training to converge faster. The ANFIS approach is much preferred over the DQN+RBC approach for four main reasons: a good policy that can be used from the beginning, the comprehension power that gives the ability to explain the results at the end, the local firing of rules which limits the problem of catastrophic forgetting, and the fast convergence towards a better policy which makes it realistic to be used in a real environment. The last reason is led by the fact of using a good policy that incorporates human knowledge into the initial policy. The human knowledge provided by the expert doesn't necessitate to be optimal but to be meaningful and then with time, this knowledge will be updated and improved during training. Many improvements can be added to the ANFIS approach to improve its results: first, increase the fuzzy rules, secondly, increase the membership functions, and thirdly, increase the time of training. Last but not least, the ANFIS policy can be upgraded to handle additional knowledge easily, if provided and required by the expert, without the need to retrain everything.

Chapter 6

Conclusion and Future Research

6.1 Summary

HVAC systems, with the goal of maintaining occupants' comfort inside buildings, account for high energy consumption in buildings. These systems rely on conventional controllers such as RBC approach which depends on rules provided by an external expert. These rules are static and limited by the expert's knowledge and thus are far from optimal and require continuous monitoring. Hence, this work investigates different control approaches with the objective of optimizing HVAC energy consumption while maintaining thermal comfort and good air quality represented by CO2 concentration. The approach must abide by two main features: generic (can be applied to any building) and second it should converge in a short time.

First, Model-Predictive control (MPC), well explored in literature, has been developed to achieve this approach. In the experiments done in Chapter 3, MPC shows good results, however, they rely on the predictive models that require representative data and considerable work of validation which can affect tremendously the MPC results. Hence, another feature is added to the control approach which is model-free which implies having a control approach without the need to develop models of the environment. For this reason, it was important to tackle model-free approaches like model-free RL. Yet, model-free RL faces a big issue which is the long time and the huge data needed to have a good control policy. Thus, Chapter 4 and Chapter 5 suggest two main contributions that rely on model-free RL with some improvements to solve this issue.

In Chapter 4, model-free RL, specifically deep-Q network (DQN), is combined with rule-based control (RBC) to guide the DQN policy during training. This technique, called DQN+RBC, exploits the RBC rules at the beginning to guide the policy. The results show better performance for DQN+RBC over DQN at the beginning. However, DQN+RBC takes a very long time to outperform RBC, which makes it not an applicable approach.

While in Chapter 5, an ANFIS policy with an extension layer added has been proposed as the initial RL policy. Fuzzy rules which represent the expert's knowledge are extended by the extension layer and injected into the RL policy represented by the ANFIS policy. Then, the updated version of Reinforce, a policy gradient algorithm, is used to improve directly the ANFIS policy. Experiments done in Chapter 5 show that the ANFIS approach trained by the policy gradient algorithm represents an applicable approach since it generates good control in a short time.

6.2 Future Research

This work represents an exploring study in the field of optimizing the energy consumption of HVAC systems. This section tackles different aspects that are worth exploring in future research:

- First, there are many hyperparameters that can affect the results such as the learning rate, discount factor, neural network architecture (number of hidden layers, activation function, number of neurons per layer), and epsilon (in Chapter 4) which affects the exploration-exploitation of the RL agent. The length of history in the state, the number of membership functions (chapter 5), and the width of the membership functions as well affect the results. These hyperparameters are not well-tuned and require deep tuning to improve the performance of the algorithms developed and then may generate a better control.

- Second, multi-objective optimization: This work represents a multi-objective optimization with multiple and conflicting objectives: minimizing energy consumption, maintaining good thermal comfort, and maintaining good air quality. The objective of maintaining thermal comfort leads to energy consumption and thus the objective of minimizing energy consumption conflicts with the objective of maintaining thermal comfort. However, most of the work done in RL so far and in this PhD assumes a single reward function that represents the weighted sum of all objectives

together. Hence, reward design to handle multi-objective reinforcement learning is worthwhile exploring in future research.

- Third, testing on a real system: The approach developed in Chapter 5, an ANFIS approach with the extension layer added and trained using policy gradient algorithm, represents an applicable control approach with three main aspects: generic, converges in a short time, and model-free. It is important to deploy and test this approach on a real HVAC system in the future. More analysis can be conducted thus on the policy learned which can be explained as mentioned in Chapter 5 since the policy is comprehensive.

- Fourth, multi-agent reinforcement learning: In the experiments done, we work on a single zone system and the developed RL approaches consider a single RL agent. However, real-world HVAC systems are much more complex and consist of multiple zones. Hence, it is important to consider multi-agent reinforcement learning algorithm that aims at training different RL agents at the same time and thus controlling different controls simultaneously and leading to energy efficiency.

- Fifth, testing on different environments: Buildings are very heterogeneous. To analyze well the approaches developed in this work, specifically, the one developed in Chapter 5, many HVAC systems in different buildings should be considered for testing the approach. It is important to conduct some of the experiments in simulation before deploying directly the approach on a real system. This step is important at the beginning to make sure that the approach can handle different environments safely.

- Last, transfer learning: Transfer learning consists of using the knowledge acquired by a model after training it on a building to be the starting point in another building. Our work done in RL is tested only on one system in simulation (the office case study). Thus, the topic of transfer learning has not been investigated in this research. This topic represents a challenging branch in reinforcement learning and is valuable to explore in future research.

Bibliography

A. Aswani, N. Master, J. Taneja, D. Culler, and C. Tomlin. Reducing transient and steady state electricity consumption in hvac using learning-based model-predictive control. *Proceedings of the IEEE*, 100(1):240–253, 2011.

E. Atam and L. Helsen. A convex approach to a class of non-convex building hvac control problems: Illustration by two case studies. *Energy and Buildings*, 93:269–281, 2015.

E. Barrett and S. Linder. Autonomous hvac control, a reinforcement learning approach. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 3–19. Springer, 2015.

S. Boyd. Model predictive control. URL https://stanford.edu/class/ee364b/lectures/mpc_slides.pdf.

F. Brunner. Mastering the game of go with deep neural networks and tree search (silver et al., 2016). 2019.

D. B. Crawley, L. K. Lawrie, C. O. Pedersen, and F. C. Winkelmann. Energyplus: energy simulation program. *ASHRAE journal*, 42(4):49–56, 2000.

K. Dalamagkidis, D. Kolokotsa, K. Kalaitzakis, and G. S. Stavrakakis. Reinforcement learning for energy conservation and comfort in buildings. *Building and environment*, 42(7):2686–2698, 2007.

D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

G. Gao, J. Li, and Y. Wen. Energy-efficient thermal comfort control in smart buildings via deep reinforcement learning. *arXiv preprint arXiv:1901.04693*, 2019.

G. Geng and G. Geary. On performance and tuning of pid controllers in hvac systems. In *Proceedings of IEEE International Conference on Control and Applications*, pages 819–824. IEEE, 1993.

J.-S. Jang. Anfis: adaptive-network-based fuzzy inference system. *IEEE transactions on systems, man, and cybernetics*, 23(3):665–685, 1993.

K. Jneid, S. Ploix, P. Reignier, and P. Jallon. Deep q-network boosted with external knowledge for hvac control. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 329–332, 2021.

M. Killian and M. Kozek. Ten questions concerning model predictive control for energy efficient buildings. *Building and Environment*, 105:403–412, 2016.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

X. Lü, T. Lu, C. J. Kibert, and M. Viljanen. Modeling and forecasting energy consumption for heterogeneous buildings using a physical–statistical approach. *Applied Energy*, 144:261–275, 2015.

Y. Ma, F. Borrelli, B. Hencey, A. Packard, and S. Bortoff. Model predictive control of thermal energy storage in building cooling systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 392–397. IEEE, 2009.

E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International journal of man-machine studies*, 7(1):1–13, 1975.

L.-G. Manson and Mclntyre. Technical synthesis report: A summary of annexes 16 17 building energy management systems. *Energy Conservation in Buildings and Community Systems*, 40(3):394–398, 1997.

J. McCall. Genetic algorithms for modelling and optimisation. *Journal of computational and Applied Mathematics*, 184(1):205–222, 2005.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

T. Moriyama, G. De Magistris, M. Tatsubori, T.-H. Pham, A. Munawar, and R. Tachibana. Reinforcement learning testbed for power-consumption optimization. In *Asian Simulation Conference*, pages 45–59. Springer, 2018.

E. Mumolo. Logica fuzzy. Università degli studi di Trieste. https://www2.units.it/mumolo/logicaFuzzy.pdf.

A.-T. Nguyen, S. Reiter, and P. Rigo. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy*, 113:1043–1058, 2014.

L. Pérez-Lombard, J. Ortiz, and C. Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

S. Privara, J. Širokỳ, L. Ferkl, and J. Cigler. Model predictive control of a building heating system: The first experience. *Energy and Buildings*, 43(2-3):564–572, 2011.

S. Privara, J. Cigler, Z. Váňa, F. Oldewurtel, C. Sagerschnig, and E. Žáčeková. Building modeling as a crucial part for building predictive control. *Energy and Buildings*, 56: 8–22, 2013.

N. S. Raman, A. M. Devraj, P. Barooah, and S. P. Meyn. Reinforcement learning for control of building hvac systems. In *2020 American Control Conference (ACC)*, pages 2326–2332. IEEE, 2020.

P. Rockett and E. A. Hathway. Model-predictive control for non-domestic buildings: a critical review and prospects. *Building Research & Information*, 45(5):556–571, 2017.

G. A. Rummery and M. Niranjan. *On-line Q-learning using connectionist systems*, volume 37. Citeseer, 1994.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

N. Soni and T. Kumar. Study of various crossover operators in genetic algorithms. *International Journal of Computer Science and Information Technologies*, 5(6):7235–7238, 2014.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. *IFAC Proceedings Volumes*, 16(13):55–60, 1983.

TRNSYS. Transient system simulation tool. URL http://www.trnsys.com.

S. Wang and Z. Ma. Supervisory and optimal control of building hvac systems: A review. *Hvac&R Research*, 14(1):3–32, 2008.

Y.-G. Wang, Z.-G. Shi, and W.-J. Cai. Pid autotuner and its application in hvac systems. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 3, pages 2192–2196. IEEE, 2001.

Z. Wang and T. Hong. Reinforcement learning for building controls: The opportunities and challenges. *Applied Energy*, 269:115036, 2020.

T. Wei, Y. Wang, and Q. Zhu. Deep reinforcement learning for building hvac control. In *Proceedings of the 54th annual design automation conference 2017*, pages 1–6, 2017.

L. A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

C. Zhang, S. R. Kuppannagari, R. Kannan, and V. K. Prasanna. Building hvac scheduling using reinforcement learning via neural network based model approximation. In *Proceedings of the 6th ACM international conference on systems for energy-efficient buildings, cities, and transportation*, pages 287–296, 2019.

Z. Zhang, A. Chong, Y. Pan, C. Zhang, S. Lu, and K. P. Lam. A deep reinforcement learning approach to using whole building energy model for hvac optimal control. In *2018 Building Performance Analysis Conference and SimBuild*, volume 3, pages 22–23, 2018.