energies

MDPI

Article

A Multi-Agent Reinforcement Learning Approach to Price and Comfort Optimization in HVAC-Systems

Christian Blad [1,2,*,†,‡], Simon Bøgh [1,‡] and Carsten Kallesøe [2,3,‡]

[1] Robotics & Automation Group, Department of Materials and Production, Aalborg University, 9220 Aalborg, Denmark; sb@mp.aau.dk
[2] Technology and Innovation, Control Department, Grundfos, 8850 Bjerringbro, Denmark; csk@es.aau.dk
[3] Department of Electronic Systems, Aalborg University, 9220 Aalborg, Denmark
* Correspondence: cblad@m-tech.aau.dk
† Current address: Fibigerstræde 16, 9220 Aalborg, Denmark.
‡ These authors contributed equally to this work.

Abstract: This paper addresses the challenge of minimizing training time for the control of Heating, Ventilation, and Air-conditioning (HVAC) systems with online Reinforcement Learning (RL). This is done by developing a novel approach to Multi-Agent Reinforcement Learning (MARL) to HVAC systems. In this paper, the environment formed by the HVAC system is formulated as a Markov Game (MG) in a general sum setting. The MARL algorithm is designed in a decentralized structure, where only relevant states are shared between agents, and actions are shared in a sequence, which are sensible from a system's point of view. The simulation environment is a domestic house located in Denmark and designed to resemble an average house. The heat source in the house is an air-to-water heat pump, and the HVAC system is an Underfloor Heating system (UFH). The house is subjected to weather changes from a data set collected in Copenhagen in 2006, spanning the entire year except for June, July, and August, where heat is not required. It is shown that: (1) When comparing Single Agent Reinforcement Learning (SARL) and MARL, training time can be reduced by 70% for a four temperature-zone UFH system, (2) the agent can learn and generalize over seasons, (3) the cost of heating can be reduced by 19% or the equivalent to 750 kWh of electric energy per year for an average Danish domestic house compared to a traditional control method, and (4) oscillations in the room temperature can be reduced by 40% when comparing the RL control methods with a traditional control method.

Keywords: deep reinforcement learning; artificial intelligence; HVAC-systems; underfloor heating; energy in buildings; predictive analytics

Citation: Blad, C.; Bøgh, S.; Kallesøe, C. A Multi-Agent Reinforcement Learning Approach to Price and Comfort Optimization in HVAC-Systems. *Energies* **2021**, *14*, 7491. https://doi.org/10.3390/en14227491

Academic Editor: Adel Merabet

Received: 30 September 2021
Accepted: 3 November 2021
Published: 9 November 2021

1. Introduction

In the USA and Europe, roughly 35% of the energy consumption in 2008 was used in HVAC systems [1]. To reduce energy consumption and hence the carbon footprint from heat and energy production for HVAC systems, the regulation regarding insulation of buildings has increased [2]. Another way to reduce energy consumption in buildings is to use more advanced control techniques, reducing energy waste, and increasing comfort. For large buildings, Model Predictive Controllers (MPCs) have shown to be effective [3,4], but MPCs require a full thermodynamic model of the building, which is not economically feasible for regular households. Smart controllers based on scheduling according to energy prices are proposed [5,6]. These algorithms require less commissioning than an MPC's but are still comprehensive to commission. To handle the issues with commissioning and still harvest the benefits with advanced control techniques, model-free Reinforcement Learning (RL) is proposed for the control.

This article focuses specifically on control of Underfloor Heating systems (UFH) in domestic houses. Traditionally, hysteresis control with room temperature as input is used

for controlling these UFHs. This hysteresis controller fully opens or closes the control valve supplying heat to the floor dependent on the room temperature. The main issue with using hysteresis control in UFH is the slow thermodynamic properties of the system, which can lead to time constants between 10 min to 3 h depending on the floor type and material [7]. Due to the slow responses in the system, the hysteresis controller is not able to keep the temperature constant because of its inability to predict the energy demand for the rooms. The temperature of the supply water is traditionally controlled by an *ambient temperature compensated controller*. This type of controller is, as the hysteresis controller, also affected by the slow response in the convection of heat from the water to the room, but also the delayed response associated with transporting the water in the pipes.

Reinforcement Learning (RL) is a model-free adaptive control method, and as such, it is possible to adapt to the specific dynamic properties of the environment [8]. This capability makes RL particularly interesting for the control of UFH or Heating, Ventilation, and Air-conditioning (HVAC) systems in general [9]. In particular, the commissioning phase is automated with RL. Moreover, the user behavior has a large effect on the comfort level of the temperature zones. Here, RL is able to take this behavior into account in the control as well [10]. In this paper, user behavior is however not included, because the goal is to investigate how the RL algorithms will adapt to the building environments. The user behavior will just complicate this analysis.

The RL algorithm studied in this article is an online learning method. Therefore, the agent/agents of the RL will not perform optimally during training. The training time is highly correlated with the complexity of the state-action space [8]. This correlation is an issue in the RL control design for UFH, as it makes it difficult to scale the RL algorithm to houses with multiple temperature zones [11]. To illustrate the scaling problem of having a single agent to control the entire system of a one, two, three, and four-zone UFH system, the calculations of the action-state space have been made for the four cases, see Table 1.

Table 1. Size of action-state space for a one, two, three and four-zone UFH system. The action-state space grows exponentially with the number of temperature zones. Assumptions about the number of discrete values each action or state has: T_{supply}; 15, Valve per zone; 2, T_{room} per zone; 12, T_{amb}; 30, heat consumption; 10, sun; 10.

	Action-State Space Size
One zone	1×10^6
Two zones	26×10^6
Three zones	622×10^6
Four zones	15×10^9

Table 1 shows that the action-state space grows exponentially with the number of temperature zones in the system, which means that RL control does not scale well in the UFH control task or many other control problems [8]. To deal with this scaling problem, we propose to incorporate Multi-Agent Reinforcement Learning (MARL). Instead of formulating the problem as a *Markov Decision Process* (*MDP*) and use Single Agent Reinforcement Learning (SARL), the problem is formulated as a *Markov Game* (*MG*), and MARL can be used, see Figure 1 for an illustration of an MDP and an MG.

From Figure 1, it is seen that the interaction with the environment changes, but not the environment itself. Whereas the environment in a single agent system receives one action, in a multi-agent system it receives an action vector with the same size as the number of agents in the system. The states and rewards received by the agents from the environment are distributed, such that it is possible to only pass relevant state information to a given agent. Formulating an environment as an MG to use MARL has been used in other applications as well. In [12], a voltage control for a power grid [12] is designed, and MARL is applied with success for route planning in road network environments in [13]. MARL is also used for training unmanned fighter aircrafts in air-to-air combat in [14].

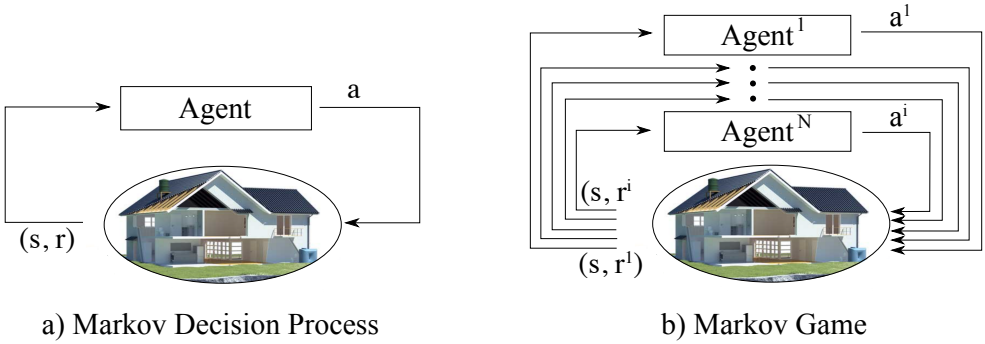


Figure 1. Illustration of the difference between a Markov Decision Process and a Markov Game. In (**a**) one agent and one environment are seen. The agent interacts with the environment by sending a tubule of actions and receiving one reward and the states of the environment. In (**b**) multiple agents and one environment are seen. The action space is split into *i* number of actions. Each agent receives a reward and the states of the environment.

RL for HVAC systems has previously been studied. In [15], Q-learning is used for a single thermostat. Here 10% energy saving is achieved by scheduling the reference temperature, such that comfort is only considered relevant if there are occupants in the room. RL is used in [16] for controlling the supply temperature and the pressure in a mixing loop. Though the mixing loop only represents a small part of the entire HVAC system, it shows that RL can outperform state-of-the-art industrial controllers. In [17], RL is used to control airflow rates for up to five zones, where each zone has an individual actuator. It is found that it is possible to reduce energy cost, but training times increased drastically when going from one to four zones. This result supports the need for an MG formulation, which makes it possible to use MARL in an HVAC system.

To the authors' knowledge, the results presented in this paper are the first to introduce MARL for the control of UFH systems. Prior papers on MARL in other parts of HVAC systems do exist, though primarily for HVAC systems for commercial buildings. In [10], an air handling unit controlled by a MARL algorithm formulated as a Markov Game in a cooperative setting is presented. This formulation means that each agent is aware of the state of the other agents, but not the current action taken by other agents. The algorithm is based on an actor-critic model. In that paper, a 5.6% energy saving is achieved. Looking slightly beyond HVAC systems, MARL is used in hot water production for domestic houses in [18]. Here, MARL is used in a distributed setting where agents are not aware of other agents' actions.

The paper is organized as follows. First, Section 2 presents the background and contributions of the work. The general SARL and MARL theory is explained, and which methods inspired the present work, finally elaborating on the contributions to the research field. Then, Section 3 presents the system design and an evaluation of the designed Dymola multi-physics simulation. This simulation serves as the training and test environment for the designed SARL and MARL algorithms. Next, Section 4 presents the underlying math and design of the RL systems, along with hyperparameters, input states, and reward functions. Finally, Section 5 presents the experiments and the analysis of experimental results, and Section 7 concludes the paper.

2. Background and Contributions

This section introduces the theory behind SARL and MARL, and argues why MARL is relevant in the control of UFH. Furthermore, the contributions of this paper are explained.

2.1. Reinforcement Learning

RL can be divided into three categories: Value-based learning, policy-based learning, and actor-critic-based learning, where actor-critic is a combination of value-based and policy-based learning. In this paper, we focus on value-based learning, specifically Q-

learning as a backbone technology, but the benefits of using MARL transfer across all three types [19]. The central idea of value-based learning is to find the optimal action-value function, which needs to satisfy the Bellman optimality equation (Equation (1)) [8]. Let $Q^*(s,a)$ be the optimal value-action function, then $Q^*(s,a)$ is given as follows:

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(s',a') \mid s,a]. \quad (1)$$

The bellman equation states that if the future state s' for all actions is known, then the optimal policy is to choose the next action a' that results in the highest Q value ($Q^*(s,a)$). This approach for choosing a is referred to as a greedy policy.

An RL algorithm learns about the environment it interacts with by iteratively updating its estimate of the action-value function, such that the action-value function converges towards $Q_i \to Q^*$ for $i \to \infty$.

Choosing the correct action if the policy is greedy is simple. Updating the Q-function so that Q_i converges towards Q_* is, however, difficult, at least within a reasonable number of iterations. The update strategy for Q-learning without so-called function approximators is given by Equation (2).

For the update strategy to converge to Q^*, it is necessary for the environment to satisfy the conditions of a Markov Decision Process (MDP).

Definition 1. *A MDP is defined by a tuple $\{S, A, P, R\}$; S is the finite number of states, A is a finite number of actions, and P is the transition probability for s_t to transition to s_{t+1} under a given action a. R is the immediate reward for the expected transition from s_t to s_{t+1}.*

$$Q^{update}(s,a) = Q^{current}(s,a) + \alpha \cdot (r_t + \gamma \max_{a'} Q(s',a) - Q^{current}(s,a)). \quad (2)$$

Equation (2) shows that for the Q-function to converge to Q^*, it is necessary to visit all state-action pairs. This is simply not feasible in large systems. Therefore, function approximation is used to approximate the Q function $Q(s,a;\theta) \approx Q(s,a)$. By using an Artificial Neural Network (ANN) as a function approximator, it is shown that a Q-function can converge [20]. Additionally, for function approximation with ANNs, a range of methods have been developed to reduce convergence time or improving convergence in value-based RL, for example double Q-learning [21], experience replay, prioritized experience replay [22], and several other methods.

Even though the above-mentioned methods do improve generalization and reduce training, RL, and machine learning, in general, do suffer from what Richard E. Bellman refers to as *"the curse of dimensionality"* [23]. For this reason, MARL is explored for RL control of HVAC systems.

2.2. Multi-Agent Reinforcement Learning

MARL is like SARL, concerned with solving decision-making problems. However, instead of one agent deciding all actions in a system based on one reward, multiple agents decide the actions and receive individual rewards or a joint reward dependent on the type of MARL setting.

This paper focuses on MARL systems formulated as an MG. The formal definition of an MG is as follows:

Definition 2. *A Markov Game is defined as a discrete time-stochastic process, a tuple $\langle N, S, {A^i}_{i \epsilon N}, {R^i}_{i \epsilon N}, P \rangle$ where N is the number of agents, S is the state space observed by all agents, and ${A^i}_{i \epsilon N}$ is the joint action space of all N agents. $R^i : S \times A \times S \to \mathbb{R}$ is the immediate reward received by agent i for transitioning from (s, A^i) to s', and P is the transition probability [24]. The definition of an MG can be interpreted as the following: At time T every i'th agent $i = [1 \dots N]$ determines an action A^i according to the current state s. The system changes to state s' with probability P and each agent receives an individual reward R^i based on the new state s'.*

When going from SARL to MARL, an entirely new dimension is added to the problem. It is, therefore, necessary to define what type of MG the system is. The type describes how the agents are formulated and affect each other. A MARL problem can be formulated in three ways: (1) A cooperative setting, (2) a competitive setting, and (3) a mixed setting [25].

Cooperative, Competitive, and Mixed Setting

In general, a MARL algorithm in a cooperative setting is formulated with a common reward function so $R^1(s, a, s') = R^2(s, a_2, s') = R^n(s, a_n, s')$ and referred to as a Markov team game. A number of different algorithms exist for solving a Markov team game, these include *team-Q* [26] and *Distributed-Q* [27]. Distributed-Q is a MARL algorithm framework, which is proven to work for deterministic environments. It has been shown that all agents will converge to an optimal policy without sharing data between agents. This is very appealing. However, because this article is concerned with a general sum problem, the approach will not work.

A MARL algorithm in a competitive setting is formulated as a zero-sum game where one agent's win over the other agents. Such a setting can be formulated as $\sum_i R^i(s, a_i, s') = 0$. There exists a number of zero-sum game algorithms, see for example *minimax-Q* [28]. This MARL algorithm setup is however of little interest for this paper, as rewards for a UFH system cannot be formulated as a zero-sum game.

Finally, in the mixed setting each agent has its own reward function, and therefore its own objective. This mixed setting is also referred to as a general sum game.

Game theory and the Nash equilibrium play an essential role in the analysis of these systems. In contrast to a cooperative setting, where it is assumed that the system's overall best reward can be found by having all agents maximize their own reward, this is not possible in a general sum setting. A number of different algorithms for general sum games have been designed for static tasks [29]. However, UHF is a dynamics system meaning that static tasks must be adapted to dynamic systems to be interesting for our work.

In addition, single-agent algorithms are used in a mixed setting [30,31]. That is, even though there is no grantee of conversion if applying SARL to a multi-agent system [24]. Algorithms, which are designed for dynamic tasks, include "*Nash-Q*" [32] and "*PD-WoLF*" [33], and will form the starting point for our work.

The idea of a Q-learning algorithm that finds a Nash equilibrium is compelling. Succeeding papers have however argued that the application of Nash-Q is limited to environments that have a unique Nash equilibrium for each iteration [25]. More recent work on fully decentralized MARL has been proven to converge under the assumption of using linear function approximators for the value function [34]. Even though this algorithm is distributed, a joint Q function is incorporated, which makes all agents aware of each other. This is necessary to prove general convergence, but it also increases complexity as the number of agents grows and therefore makes the approach less scalable. In a more recent work, agents are distributed in a similar manner to our work [35]. The main difference between their architecture and our framework is that our agents only observe parts of the state space. Moreover, different methods are utilized for updating the Q-function.

2.3. Contributions

This paper extends the current state-of-the-art for model-free control of UFH systems, including testing SARL on UFH systems and presenting a novel MARL approach to HVAC systems. The novelty lies in the interaction between agents in the MARL algorithm. In distributed Q and Nash-Q, agents are either not aware or completely aware of each other. In this paper, each agent acts according to a well-defined structure as described in Section 4. Furthermore, the comparison between the SARL simulation and MARL simulation validates the hypothesis that MARL can reduce training times in HVAC systems. Lastly, we present a novel method to ensure robustness for controlling the supply temperature in HVAC systems.

3. System Design and Evaluation

To test hyperparameters, input states, and algorithms, it is necessary to have a simulation environment. A simulation environment is never a 1:1 representation of the real world, but for a simulation environment to be applicable in this study, it is necessary that the simulation, to a large extent, has the same dynamic behavior. To accomplish this, Dymola has been used as the simulation tool.

3.1. Dymola Multi-Physics Environment

Dymola is a Modelica-based multi-physics simulation software and, as such, is suitable for simulating complex systems and processes. Several libraries have been developed for Dymola. For the simulations in this paper, the standard Modelica library and the Modelica Buildings library are used.

The simulation can be split into two parts: (1) The hydraulic part and (2) the thermodynamic part. The hydraulic part of the simulation can be described by a mixing loop, a pump, one valve per temperature zone, and some length of pipe per temperature zone.

The thermodynamic side of the simulation is constructed using the base element "*ReducedOrder.RC.TwoElements*". This element includes heat transfer from exterior walls, windows, and interior walls to the room. It furthermore includes radiation from the outside temperature and radiation from the sun. This means that wind and rain do not affect the simulation, as they are assessed to be smaller disturbances. These disturbances are not negligible, but for the purpose of this paper, the simulation results will still indicate the saving potential that can be expected in real-life installation. The element is made in accordance with "VDI 6007 Part 1", which is the European standard for calculating the transient thermal response of rooms and buildings [36].

The length of pipe used in each zone and parameters for the windows, walls, zone area, and volume are shown in Table 2.

Table 2. Parameters used for each temperature zone. A and B refer to if it is the one-zone simulation or the four-zone simulation.

Parameter	Zone1A	Zone1B	Zone2B	Zone3B	Zone4B
Length of pipe	105 m	56 m	105 m	42 m	70 m
Window area	20 m^2	12 m^2	25 m^2	12 m^2	24 m^2
Wall area	39 m^2	36 m^2	40 m^2	12 m^2	30 m^2
Zone area	30 m^2	16 m^2	30 m^2	12 m^2	20 m^2
Zone volume	80 m^3	48 m^3	90 m^3	36 m^3	60 m^3

To simulate how the room receives heat from the floor, a floor element has been constructed. The floor element incorporates the pipe length, pipe diameter, floor thickness, floor area, and construction material. These parameters enable the floor element to simulate how the heat from the water running in the pipes will transfer through the concrete and into the room. The heat in the room is assumed to be uniformly distributed. This means that the temperature at the floor, at walls and at the sealing of the room is the same. Modeling the temperature distribution uniformly is also in accordance with "VDI 6007part1".

3.2. Evaluation of Simulation

It is not possible to validate the simulation environment with data from a real-world system. We can, however, evaluate step responses to evaluate the dynamic convection of heat from the water in the pipes to the air in the room. Additionally, we can evaluate the amount of power the rooms require and compare it to a real-world house. Lastly, the daily and seasonal power consumption can be evaluated.

To evaluate the simulation environment, a run of the simulation with hysteresis control on the valves and an outdoor compensated supply temperature is executed. Note, that hysteresis control is the control method traditionally used in the UFH system. For the

validation, a simulation with a one-temperature zone system is used. However, a similar simulation has been made for a four-temperature zone system with similar results.

All simulations are made with a hysteresis control with reference point 22 °C and a dead band of ±0.1 °C. The outside compensated supply temperature follows a linear model, see Equation (3).

$$T_{supply} = -0.6 \cdot T_{ambient} + 42. \quad (3)$$

Firstly, the room temperature of an entire heat season is plotted in Figure 2.

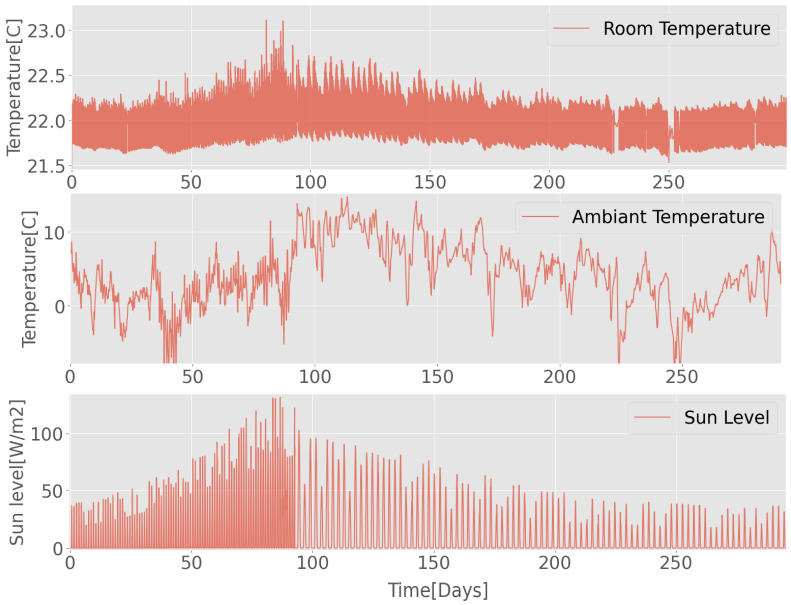


Figure 2. Simulation results of one heating season, with a traditional controller and outside compensated supply temperature.

From Figure 2, it can be seen that a heating season is approximately 280 days. The heating season is defined here as the period of the year where the building needs energy to sustain a zone temperature of 22 °C. The simulation starts 1 March, and the period from 1 June to 1 September has been removed from the weather file as no heat is needed in this period. The seasonal effect can be seen in the figure, where occasional overshoots happen in the period from day 70 to day 140. Hence, in the fall/spring period, where heat is needed during the night and morning but not during the daytime, overshoots happen. The temperature is otherwise oscillating between 21.7 °C and 22.2 °C.

To investigate the response more closely, the room temperature and the associated valve position is plotted over a period of 2 h and 30 min, see Figure 3.

Figure 3 shows that when a valve is opened, 1700 s (0.02 days) will pass before the temperature gradient in the room becomes positive. Additionally, it can be seen that when the valve is closed, the temperature will continue to rise an additional 0.1 °C and another about 1700 s will pass before the gradient becomes negative. This behavior is due to the slow dynamic properties that are expected of a UFH system, and therefore it also validates that this simulation resembles the typical dynamics of a UFH system.

The price of heating over one heating season is plotted in Figure 5. Before reviewing the plot, it is necessary to explain how this price is calculated. The price will also serve as a benchmark to prove that significant cost savings are possible by utilizing reinforcement learning in UFH. To that end, in this article, it is assumed that the heat supply is an air-to-water heat pump.

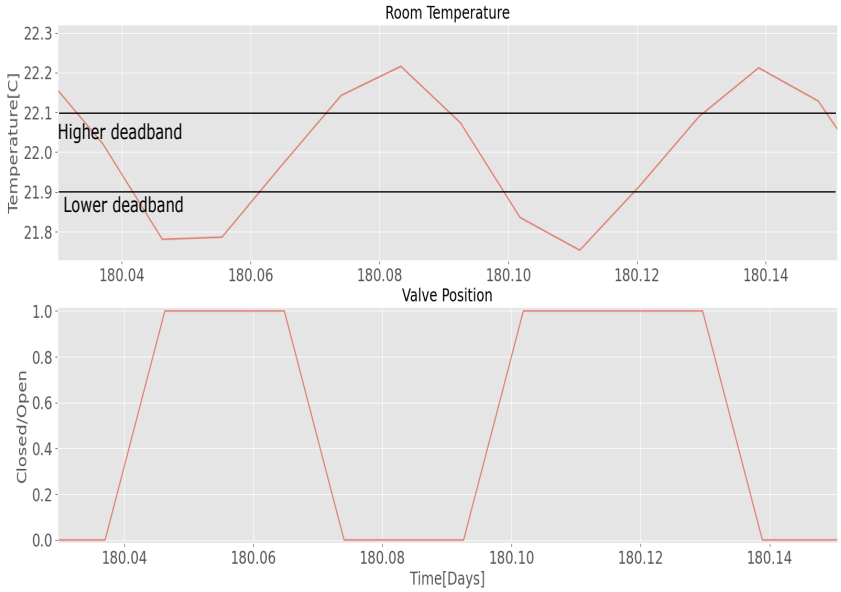
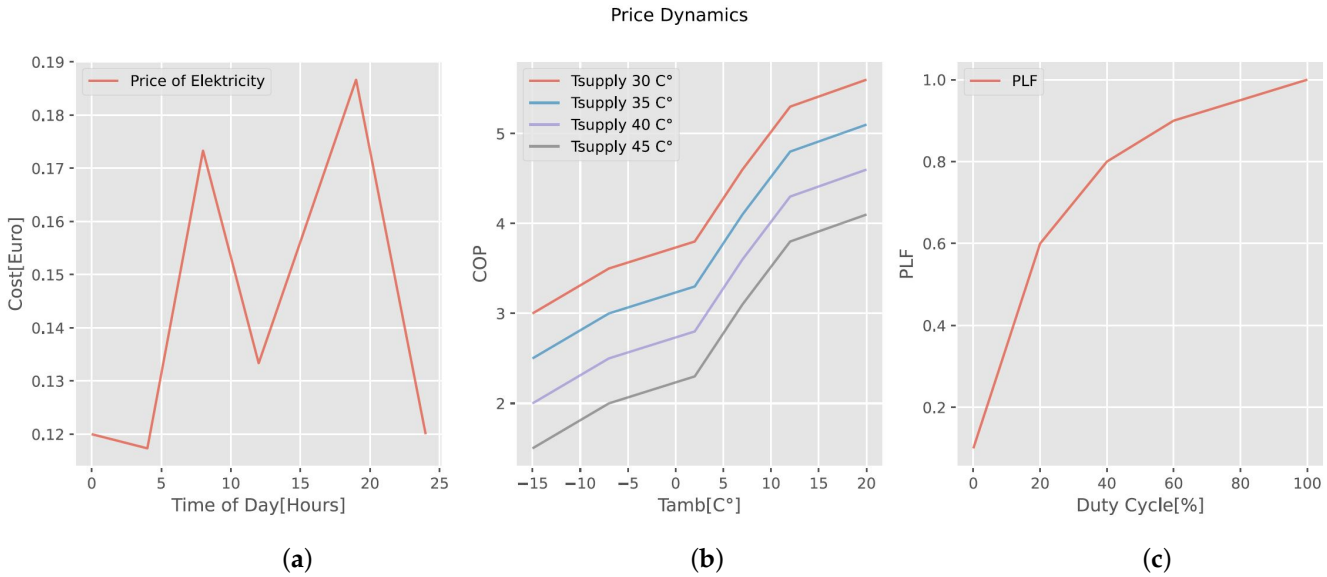


Figure 3. Room temperature and associated valve position over a period of 8640 s (0.1 days). By investigating the graph, the dynamic response can be analyzed.

The price of heating with a heat pump can be simulated by knowing the cost of electricity, the dynamics of a heat pump, and the power consumption of the system. The cost of electricity is assumed to fluctuate during the day. The average Danish price of electricity during a day can be seen in Figure 4a [37]. The dynamics of a heat pump can be described with the Coefficient of Performance (COP), which is a function of the ambient temperature and the supply temperature. This COP can be seen in Figure 4b [38]. Additionally, it is necessary to describe the Part Load Factor (PLF), which describes how the efficiency of the heat pump depends on the duty cycle. This PLF is shown in Figure 4c [39].



(**a**) (**b**) (**c**)

Figure 4. Dynamics of a heat pump: (**a**) Shows the average electricity prices, including taxes in Denmark as a function of the time of day (tod). (**b**) Shows the Coefficient of Performance(COP) as a function of the ambient temperature, for four different supply temperatures. (**c**) Shows the Partial Load Factor(PLF) as a function of the duty cycle (D).

With the Cost of Electricity (CE), the COP and the PLF described and the power consumption of the system (ΔE) available from the simulation, the cost of heating with a heat pump can be simulated with Equation (4):

$$cost = \frac{\Delta E}{COP(T_{amb}, T_{supply}) \cdot PLF(D)} \cdot CE(tod). \quad (4)$$

From Figure 5, it is evident that the price of heating over one heating season varies. The lowest cost is found in the spring/autumn period and highest during winter. Though there is a yearly trend, it is also evident that the price during a 14-day period can vary 30%, as seen from day 200 to 210.

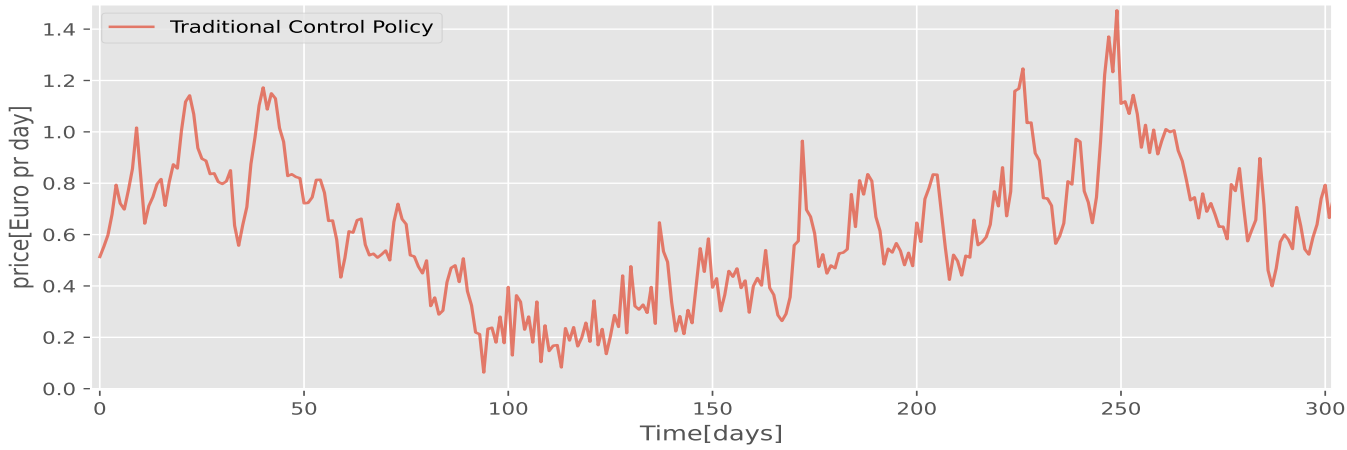


Figure 5. Price of heating over one heating season for a one-temperature zone system, of 30 m^2. The price is price per day in Euro.

Finally, the power consumption is reviewed. The temperature zone is 30 m^2 and consumes 3561 kWh over a heating season at a reference temperature of 22 °C, meaning an average of 118 kWh per m^2. An average Danish house uses 115 to 130 kWh per m^2 [40], which shows that the simulation is within what is considered average in a Danish climate.

The use case is now described, the simulation has been evaluated, and it has been shown that it resembles a traditional Danish house. The RL algorithm can therefore be designed and tested on this simulation.

4. RL Algorithm Design

Figure 6 illustrates the hydraulic system for a four-zone UFH system. The system in the figure consists of a heat pump with a supply temperature and four on/off valves controlling the temperature of each of the four zones. The MARL algorithm is designed as an MG in a general sum setting as explained in Section 2.2. By reviewing Section 3, it can be seen that the natural way of dividing the UFH environment into multiple agents is achieved by having one agent control the supply temperature and one agent for each of the temperature zones. Each of the temperature zone agents will, in this setting, control the on/off valves supplying the zones with hot water. This setup means that for a four-temperature zone UFH system, there are five agents, as illustrated in Figure 6.

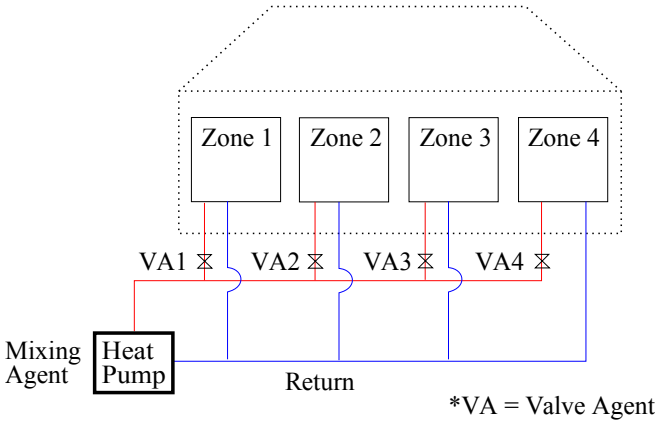


Figure 6. Four-zone temperature system, with a air to water heat pump. From the illustration, it can be seen how the agents are divided into one mixing agent and four valve agents. The mixing agent controls the supply temperature and the valve agents control the flow to each zone.

By splitting the environment into five agents instead of one, the action space is changed. The result of this change is shown in Table 3. From the table, it is seen that the actions can be formulated as one action with 240 discrete values or as 5 separate actions, which are either 15 or 2 discrete values. With this formulation of the action space, $A = [a_1, a_2, a_3, a_4, a_5]$. A distributed RL formulation of the problem can be written as in Equation (5).

$$Q^i_{t+1}(s, a^1, ., a^n) = \mathbb{E}_{s,a,r,s'}\Big[(1-\alpha)Q^i_t(s, a^1.a^n) + \alpha[r^i_t + \beta \max_{a^i} Q^i_t(s', a^1, ., a^n)]\Big]. \tag{5}$$

Table 3. Action space for Single Agent RL ann Multi Agent RL.

	SARL	**MARL**
Single Agent/Mixing Agent	240	15
Valve Agent 1	0	2
Valve Agent 2	0	2
Valve Agent 3	0	2
Valve Agent 4	0	2

From Equation (5), it can be seen that all agents have a Q-function and have full observability of the actions of all other agents. However, if the UFH system is investigated, it can be argued that each valve agent has little to no effect on each other. For this reason, it can be argued that the connections between the valve agents are unnecessary and therefore should be removed. Removing these connections will give the following formulation of the Q-functions, see Equations (6) and (7). Here, $1, \cdots, m$ refers to valve 1 to valve m and st refers to the supply temperature:

$$Q^{st}_{t+1}(s, a^{st}, a^{v_{1.m}}) = \mathbb{E}_{s,a,r,s'}\Big[(1-\alpha)Q^i_t((a^{st}, a^{v_{1.m}}) + \alpha[r^i_t + \beta \max_{a^{st}} Q^{st}_t(s', a^{st}, a^{v_{1.m}}))]\Big], \tag{6}$$

$$Q^{v_m}_{t+1}(s, a^{st}, a^{v_m}) = \mathbb{E}_{s,a,r,s'}\Big[(1-\alpha)Q^i_t((s, a^{st}, a^{v_m}) + \alpha[r^i_t + \beta \max_{v_m} Q^{v_m}_t(s', a^{st}, a^{v_m})]\Big]. \tag{7}$$

Since it is argued that zones have little to no effect on each other, it can also be argued that parts of the state space should only be locally observed. For this reason, the local state space $[s_{st}, s_{v_1}, s_{v_2}, s_{v_3}, s_{v_4}]$ are defined, and the Q-functions can be rewritten to Equations (8) and (9). Elaboration on which states are relevant for which agents are given in Section 4.3.

$$Q^{st}_{t+1}(s_{st}, a^{st}, a^{v_1..m}) = \mathbb{E}_{s,a,r,s'} \left[(1-\alpha) Q^i_t((a^{st}, a^{v_1..m}) + \alpha[r^i_t + \beta \max_{a^{st}} Q^{st}_t(s'^{st}, a^{st}, a^{v_1..m})] \right] \tag{8}$$

$$Q^{v_m}_{t+1}(s^{v_m}, a^{st}, a^{v_m}) = \mathbb{E}_{s,a,r,s'} \left[(1-\alpha) Q^i_t((s^{v_m}, a^{st}, a^{v_m}) + \alpha[r^i_t + \beta \max_{v_m} Q^{v_m}_t(s'^{v_m}, a^{st}, a^{v_m})] \right]. \tag{9}$$

With the Q-functions formulated, the foundation for the MARL algorithm is established. An illustration of the structure of how the agents are interacting with the environment can be seen in Figure 7.

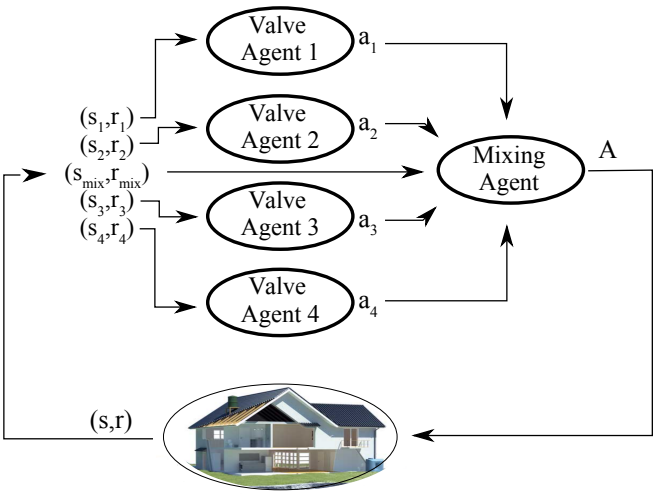


Figure 7. Illustration of how the agents interact with each other and the environment. In the figure, four valve agents, one mixing agent, and a four-zone Underfloor Heating System can be seen. The sequence of interactions is as follows; all valve agents choose an action based on the state of the environment. These actions are passed to the mixing agent, the mixing agent chooses an action based on the state of the environment and the actions of the valve agents. All actions are passed to the environment and the environment returns states and rewards for the agents.

The following explains (1) which RL methods are used, (2) the pseudo-code for the algorithm, (3) how the reward functions are formulated, (4) how the state-action space is distributed, and (5) which hyper-parameters are used.

4.1. Reinforcement Learning Methods

The backbone RL algorithm used in this paper is the deep double Q-network algorithm with experience replay, N-step learning, and epsilon greedy decay exploration.

The N-step eligibility trace used in the algorithm is also used in [41,42]. This approach is chosen due to the slow dynamic of the UFH system, making eligibility traces desirable. Experience replay is also used to enhance data efficiency. The implementation of experience replay is customized to N-step eligibility trace and MARL. This is done by maintaining the experience in mini-batches with the same length as the N-step eligibility trace. Additionally, the experience replay is synchronized between the agents, so the experience for $agent_1$ at time t has the same timestamp as the experience for $agent_i$ at time t. The pseudo-code for the MARL algorithm can be seen in Algorithm 1.

Algorithm 1 MARL Deep Q-Learning.

for each iteration k **do**
  for each environment step **do**
    Observe state S_t and distribute local states $s_t^1, s_t^2 ... s_t^n$ to respective agents.
    Valve agents select action $a_{v(n)}^k = \max_{a_{v(n)}} Q(s, a_{mix}^{k-1}; \theta)$ or pick random action with probability epsilon.
    Mixing agent select action $a_{mix}^k = \max_{a_{mix}} Q(s, a_{v(n)}^k; \theta)$ or pick random action with probability epsilon.
    Collect action $a_t^1, a_t^2 ... a_t^n$ to A execute A_t and observe next state S_{t+1} and reward.
    Store $(s_t^n, a_t^n, r_t^n, s_{t+1}^n)$ in replay buffers $\mathcal{D}^n$.
    Decay epsilon
  end for
  for each update step **do**
    Agent n sample experience of size B each with length n-step from $\mathcal{D}^n$.
    Compute Q-values as described in Equations (7) and (9).
    Calculate losses for all agents.
    Calculate gradients with respect to the network weights θ and perform gradient step.
    Every C environment step, update target networks.
  end for
end for

4.2. Reward Functions

To gain intuition about the reward functions, two base functions are defined—one for the valve system and one for the supply temperature.

4.2.1. Valve Reward

The valve reward is shown in Equation (10). The reward function depends on two sub-functions shown in Equations (11) and (12).

$$\mathrm{R}(T_z, V, H_c) = \begin{cases} 2 - (T_z - T_{ref}) \cdot & \text{if } 21.6 < T_z < 22 \\ -(T_z - T_{ref}) & \text{if } 21.6 > T_z \text{ or } T_z > 22 \\ -H_c & \text{if } SC = active \end{cases}, \tag{10}$$

$$\mathrm{SC}(T_z, V) = \begin{cases} \text{not active} & \text{if } 21 < T_z > 23 \\ \text{active} & \text{if } T_z < 21 \text{ and } VP = 0 \\ \text{active} & \text{if } T_z < 23 \text{ and } VP = 1 \end{cases}, \tag{11}$$

$$H_c\ (\mathrm{SC}) = \begin{cases} 1 + H_C & \text{if SC = active} \\ 5 & \text{if SC = not active} \end{cases} \tag{12}$$

The abbreviations in the equations above are the following: R = Reward SC = Safety controller, T_z = Zone temperature, V = Valve position, and H_c = Hard constraint.

The two sub-functions Equations (11) and (12) are a part of a safety framework for ensuring robust behavior in RL [43]. In [43], it is demonstrated that by implementing a safety controller, in this case a tolerance controller, on top of the RL algorithm, robust behavior can be ensured. The safety controller is activated when the RL controller is trying to explore the action-state spaces that are known to be outside safety boundaries. The H_C variable is the soft constraint that iteratively increases linearly as the agent continuously tries to explore parts of the action-state spaces, which is known to have negative comfort characteristics. The immediate +2 reward, which is received when the agent is less than

−0.4 °C from the reference point, enforces that the comfort is highest when the temperature is in this range. When the reward function is used in the MARL setting, the reward is simply distributed so that each agent receives the reward for the zone it is controlling. When the rewards are used in the SARL setting, all rewards are summed into one reward.

4.2.2. Supply Temperature Reward

The reward function for the supply temperature is shown in Equation (13), and the associated sub-functions are shown in Equations (14) and (15).

$$R(T_z, V, H_c) = \begin{cases} 2 - (T_z - T_{ref}) - P & \text{if } 21.6 < T_z < 22 \text{ and } VP = 1 \\ -(T_z - T_{ref}) - P & \text{if } 21.6 > T_z \text{ or } T_z > 22 \end{cases} \quad (13)$$

$$SC(T_z, V) = \begin{cases} \text{not active} & \text{if } T_z, > 20.5 \\ \text{active} & \text{if } T_z, < 20.5 \text{ and VP = 1} \end{cases} \quad (14)$$

$$H_c(SC) = \begin{cases} 1 + H_C & \text{if SC = active} \\ 5 & \text{if SC = not active} \end{cases} \quad (15)$$

The abbreviations in the equations above are the following: R = Reward, SC = Safety controller, T_z = Zone temperature, V = Valve position, H_c = Hard constraint, and P = Price.

From Equation (13), it is seen that the reward is much like the reward from the valve agent, with the difference that the +2 reward requires that the valve is open. When heating with a heat pump, it is optimal to have as much water circulation as possible. Adding that the reward is highest when the valve is open enforces this behavior. The price is a scalar between 0 and 1, and the lower the price of heating, the better.

Like in the valve reward function (Equation (10)), a safety controller is put on top of the RL algorithm. Simulation results show that it has some of the same effects as in the case of the valve agent reward. The safety controller is the *outside compensate supply temperature* used in the simulations in Section 3.2. The safety controller is activated whenever the temperature in a given zone is 1.5 °C lower than the reference temperature, and the associated valve is open.

4.3. Input States

The input states to the RL agent are a mix of actual states of the system and parameters that are functions of the system's states. The input states are divided into valve input states and supply input states. The input state space is explained from a MARL point of view. From a SARL point of view, the same states are used. These are combined in a single tuple and sent to the single agent.

4.3.1. Valve Input States

Seven states are used in the valve agents. All states are normalized so they assume a value from 0 to 3. These states can be seen in the list below.

- Valve agent input:
  - Room Temp $i \in \{1, \cdots, n\}$, [°C];
  - Δ Room Temp $i \in \{1, \cdots, n\}$, [°C];
  - Hard constraint Valve $i \in \{1, \cdots, n\}$;
  - Supply temperature, [°C];
  - Ambient temperature, [°C];
  - Sun, [w/m^2];
  - Time of day, [hour and minutes].

From the list above, seven input states can be seen, the "Δ Room Temp" is the gradient of the room temperature, and the "Sun" is the strength of the Sun. In the weather file, this strength is measured in [W/m^2].

4.3.2. Supply Temperature Input States

The input states for the supply temperature can be seen in the list below.

- Supply agent input:
  - Room Temp$i \in \{1, \cdots, n\}$, [°C];
  - Δ Room Temp$i \in \{1, \cdots, n\}$, [°C];
  - Hard constraint Supply;
  - Supply Temperature, [°C];
  - Ambient Temperature, [°C];
  - Sun, [w/m^2];
  - Time of day, [hour and minutes];
  - Price, [Euro].

From the above, it can be seen that many of the states are the same as in the valve agent, only the price and the hard constraint states are different. An overview of how the number of states increases as the number of temperature zones also increases is given in Table 4.

Table 4. Overview of how the number of input states is increasing when more zones are added to the system for both MARL and SARL.

	Supply Agent	**Valve Agent**	**Single Agent**
One zone	8	7	10
Four zones	17	7	22

4.3.3. Action Space

As explained in Section 3, the action space for the SARL formulation for a four-zone UFH system is a discrete value from 0 to 239 and the action space for a MARL system is a vector as follows: $A = [a_1, a_2, a_3, a_4, a_5]$, see Table 3.

Tests in the simulation have shown that when doing simulations with SARL, it is necessary to manipulate the action state space, so that there are 31 actions instead of 240 for a four-zone system. This reduction is done by separating the control of the valves and the control of the supply temperature. That is, the agent can either control the valves or the supply temperature at a given step. This reduction results in 16 actions for the valves, and 15 for the supply temperature, hence the 31 actions all in all. The reduction of the action space is also done in SARL for the simulation of the one-zone system resulting in the 16 actions. That is, 15 mixing actions and one action for closing the valve, hence the action state space for the one-zone SARL and MARL algorithms are similar and therefore, it is expected that the convergence time will be similar. An overview of the action space in the different settings can be seen in Table 5.

Table 5. Overview of how the input states are increasing when more zones are added to the system and how MARL and SARL are affected by this.

	Supply Agent	**Valve Agent**	**Single Agent**
One zone	15	2	16
Four zones	15	2	31

4.3.4. Hyperparameters

The hyperparameters used for training and testing the algorithms are displayed in Table 6. The values seen below are found from empirical tests of the algorithms.

Table 6. Hyperparameters used for training the agents.

	Supply Agent	Valve Agent	Single Agent
Learning rate	0.01	0.01	0.01
Epsilon decay	0.0005	0.0005	0.0005
Epsilon max	1	1	1
Epsilon min	0.1	0.1	0.1
batch size	432	432	432
N_steps	45	45	45
gamma	0.9	0.9	0.9
ANN	$60 \times 60 \times 60$	$60 \times 60 \times 60$	$90 \times 90 \times 90$
Target update rate	540	540	540

The following section will present a test plan explaining which experiments are required to prove scalability in MARL and increased performance when compared to a traditional controller. Furthermore, the results of the experiments will be presented and analyzed. All raw data obtained from these simulations can be found in (https://github.com/ChrBlad/MARL_data, accessed on 30 March 2021).

5. Experiments and Results

A test plan is established to validate that the MARL formulation reduces training time and hence improves scaling capabilities compared to a SARL formulation. The test plan includes two test levels, consisting of environments with one and four temperature zones, respectively. By introducing these test levels, it is verified that the scaling problem stated in the introduction is solved using MARL. The test plan is outlined in Table 7 and consists of six tests and three comparisons. Both test levels consist of simulations of 1000 days. To review how MARL performs in comparison to SARL, the reward of the MARL is compared to the reward of SARL.

Table 7. This test plan elaborates on which experiments are necessary to prove scalability in MARL and better performance in Reinforcement Learning when comparing with traditional control (TC).

	SARL	MARL	TC
Test Level 1	1	1	0
Test Level 2	1	1	1

5.1. Test Level 1

In Test level 1, a single zone UFH system is simulated with a MARL and a SARL controller, respectively.

Figure 8 shows that the MARL and SARL algorithms converge in approximately the same time, about 180 days. This is in accordance with the assumption that the convergence time should correlate with the size of the action state space. Since the distribution of agents in a one-zone system almost results in the same action state space as if it was one agent, the convergence time is more or less the same. The performance of the RL algorithms, compared to a traditional controller, is better after 40 days of training. However, a drop in performance is seen after 80 days. This drop is due to change of seasons and therefore load conditions that are unknown to the RL. After 120 days, the RL algorithms are shown to perform better than the traditional controller. There are a few days during a heating season where the MARL and SARL controllers are not performing better than the traditional controller. This can be found around day 220 and 240. These days are exceptionally cold and therefore the system is in saturation and therefore the RL-controller cannot improve the performance.

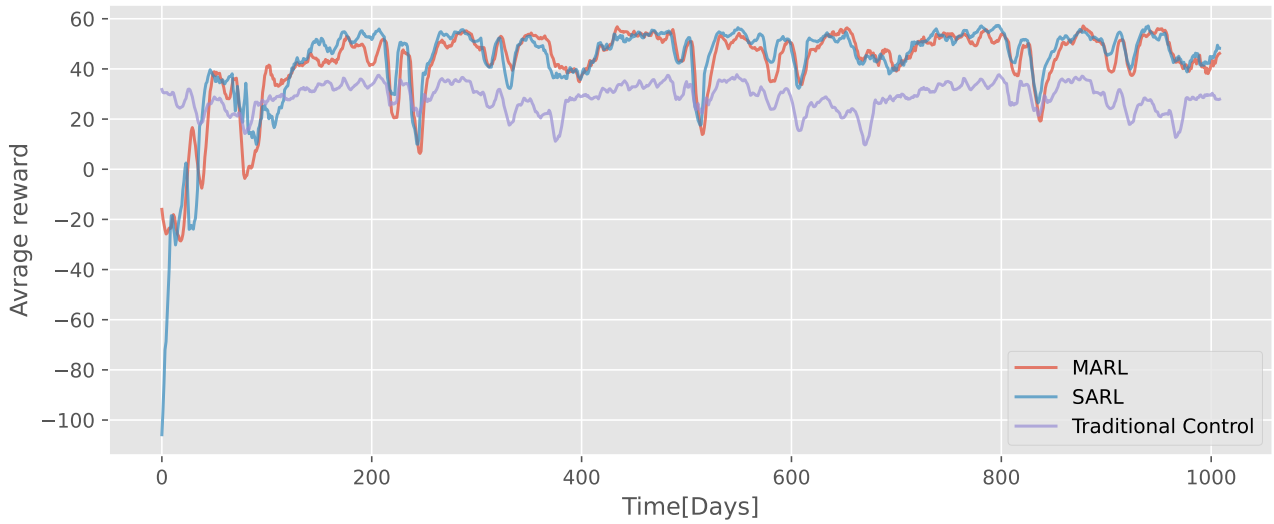


Figure 8. Reward signal of the one-zone UFH system. Simulation time is 1000 days.

Some variation in the reward signal is observed after the algorithms have converged. This is due to the seasonal effect on the comfort and prices of heating. This is to be expected and may be different for the reward plots seen in other articles, where the reward converges to a constant value.

5.2. Test Level 2

In test level 2, the SARL and MARL performance in a four-zone UFH system is compared. The result of test level 2 is shown in Figure 9.

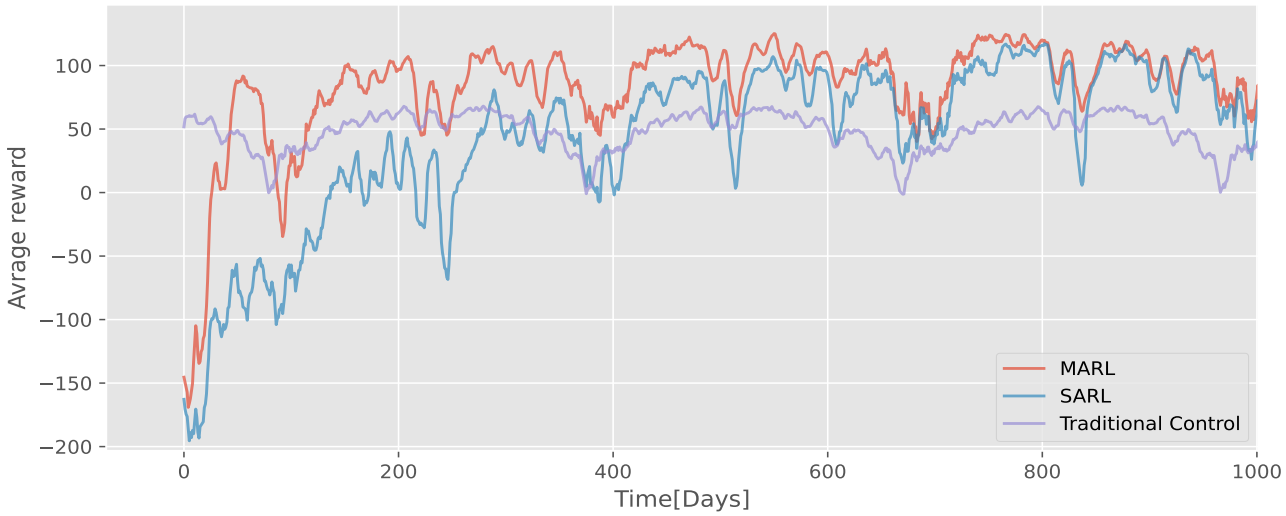


Figure 9. Reward signal of the four-zone UFH system in, simulation time is 1000 days (working on 2000-day simulation results).

From the reward plot of test level 2, some of the same behavior as in test level 1 can be observed. The MARL agents converge after 180 days but are performing well after 40 days. The SARL agent converges to approximately the same as the MARL but after 600 days. This difference in convergence speed confirms the assumption on the relation between convergence speed and size of the action space. Whereas SARL and MARL both works for a one-zone UFH system, the advantages with MARL are clear in the four-zone simulation. MARL results in faster convergence and marginal better convergence over a time period of 1000 days.

As the SARL and MARL converge to almost the same policy with identical performance, there is no reason to compare the performance in terms of comfort and price. However, it is interesting to compare the RL performance to the traditional controller. For this comparison, MARL is used.

Traditional Control vs. MARL

The reward signal is a good measurement of performance. It does, however, not explain how much better a MARL algorithm performs in terms of comfort and price compared to a traditional controller. The performance in terms of comfort and price is therefore evaluated.

Firstly, the room temperature data for zone #1 is evaluated from a histogram to determine the distribution of the temperature data.

As shown in Figure 10, the temperature distribution for a traditional controller is far from normally distributed. For this reason, standard deviations or variance cannot be used to calculate performance. A box plot for each temperature zone is therefore used. From this plot, it is possible to conclude on the variation in the room temperature for each zone.

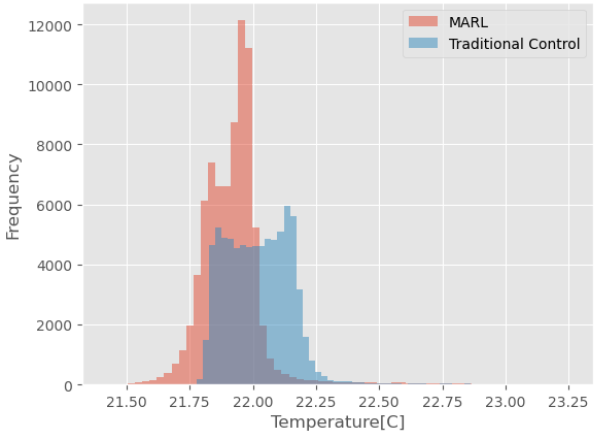


Figure 10. Histogram of room temperature in zone #1 from simulation with four temperature zones.

Figure 11 contains four box plots showing the temperature variations in each of the four temperature zones. From these plots, it can be deduced that the variation is about 40% less with MARL compared to a traditional controller. Note, outliers have been removed from the data before being used for the box plots. Smaller variations in the temperature give both better comfort and reduced price. Hence the MARL agents behave like this.

The cost of heating with the MARL algorithm, the traditional controller, and the savings as a function of time can be seen in Figure 12. The cost is calculated based on the description in Section 3.2 and Equation (4).

From this plot, it can be concluded that the savings vary over the season, but the MARL controller performs better than the traditional controller at any point in time. The average savings are 19% when using MARL compared to a traditional controller. During the first 20 weeks, it can be seen that the savings oscillate more then the remaining 120 weeks, the reason for this is naturally that the control policy of the MARL agent has not jet converged.

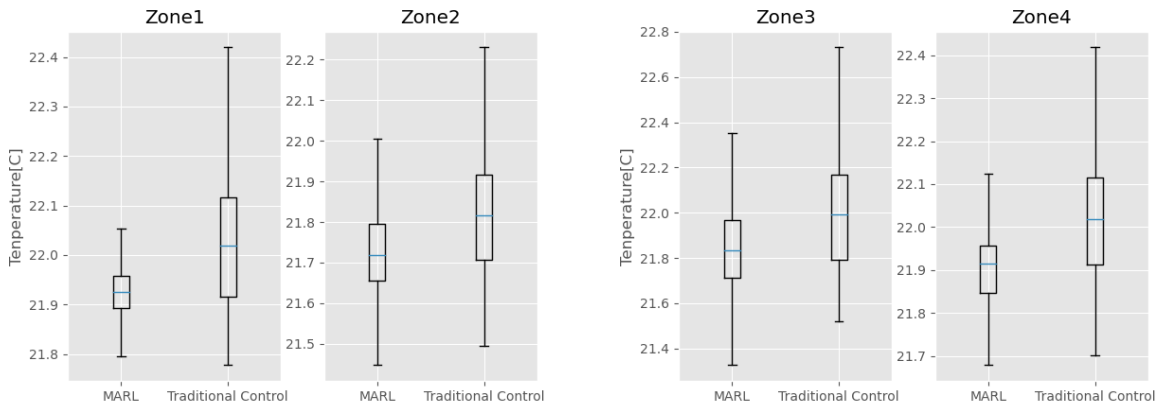


Figure 11. Boxplots of temperature distribution in (**a**) zone 1, zone 2, (**b**) zone 3, and zone 4 in the four-zone UFH system for the MARL simulation and the simulation with a traditional control policy.

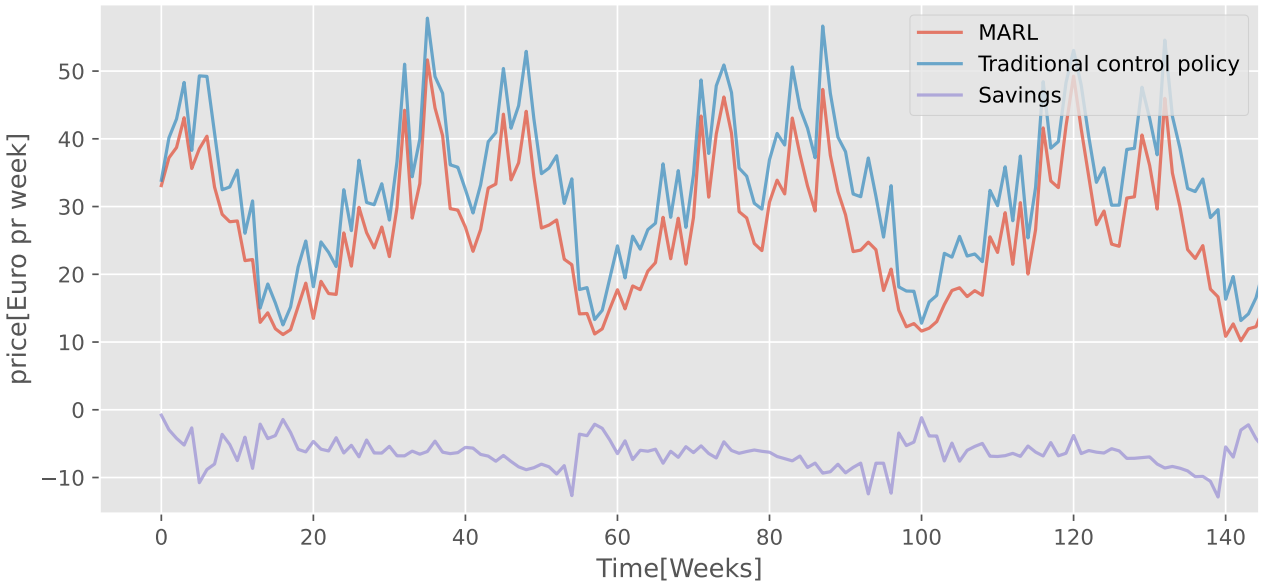


Figure 12. Price of heating per week for MARL and traditional control for a four-zone UFH system over a 1000-day period.

6. Discussion

The results of this paper is based on a unique simulation environment, for this reason general savings of 19% can not be proven, and there is no standard model to benchmark against. However when reviewing Sections 3 and 3.2, it is shown that both the benchmark controller and the simulation environment gives a valid view of a real world test. In the Git repository, the simulation environments and control code are available. The authors do encourage researchers to benchmark further algorithms against this code. Since this is a simulation environment and not a real world test, the algorithm has not been tested for robustness towards sensor faults, sliding errors in sensor measurement, or stochastic behavior. The main purpose of this paper is to prove that this algorithm can converge fast to the thermal properties of a house. Further research will prove if this algorithm is resilient towards stochastic behavior in the forms mentioned above.

7. Conclusions

In this paper, we explore a novel approach for building control strategies with Multi-Agent Reinforcement Learning for underfloor heating systems. Formulating the Underfloor Heating (UFH) system as a Markov Game (MG) instead of a Markov Decision Process (MDP) makes it possible to distribute the action space locally between agents. Moreover, it is argued that it is possible to have some states of the system observed locally.

By using a multi-agent structure and observe states locally, it is demonstrated in simulation that convergence time can be reduced by more than 70% when compared with a single agent approach. Furthermore, it is shown that as the complexity of the state-action space increases, the convergence time of the MARL agent will remain acceptable. In comparison, a SARL agent becomes almost unfeasible. Additionally, the simulation experiments show that MARL is a better alternative to traditional control methods when comparing heating costs. The simulation shows a 19% reduction of the heating cost. If assuming a Seasonal Coefficient of Performance (SCOP) of 4, the average size of a house is 140 m^2 [44], and the average heat consumption is 115 kWh per m^2, these savings sum to approximately 750 kWh of electric energy per year in an average Danish household.

Author Contributions: All authors has contributed equally to this study. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Innovation Fund Denmark.

Data Availability Statement: The data supporting this study can be found in https://github.com/ChrBlad/MARL_data, accessed on 30 March 2021.

Acknowledgments: Supported by Innovation Fund Denmark and Grundfos A/S. A special thanks to Erik B. Sørensen, Control Specialist, Grundfos A/S for supporting the work on the simulation model.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pérez-Lombard, L.; Ortiz, J.; Pout, C. A review on buildings energy consumption information. *Energy Build.* **2008**, *40*, 394–398. [CrossRef]
2. Gaglia, A.G.; Tsikaloudaki, A.G.; Laskos, C.M.; Dialynas, E.N.; Argiriou, A.A. The impact of the energy performance regulations' updated on the construction technology, economics and energy aspects of new residential buildings: The case of Greece. *Energy Build.* **2017**, *155*, 225–237. [CrossRef]
3. Privara, S.; Širokỳ, J.; Ferkl, L.; Cigler, J. Model predictive control of a building heating system: The first experience. *Energy Build.* **2011**, *43*, 564–572. [CrossRef]
4. Huang, H.; Chen, L.; Hu, E. A new model predictive control scheme for energy and cost savings in commercial buildings: An airport terminal building case study. *Build. Environ.* **2015**, *89*, 203–216. [CrossRef]
5. Yu, L.; Jiang, T.; Zou, Y. Online energy management for a sustainable smart home with an HVAC load and random occupancy. *IEEE Trans. Smart Grid* **2017**, *10*, 1646–1659. [CrossRef]
6. Tsui, K.M.; Chan, S.C. Demand response optimization for smart home scheduling under real-time pricing. *IEEE Trans. Smart Grid* **2012**, *3*, 1812–1821. [CrossRef]
7. Kull, T.M.; Simson, R.; Thalfeldt, M.; Kurnitski, J. Influence of time constants on low energy buildings' heating control. *Energy Procedia* **2017**, *132*, 75–80. [CrossRef]
8. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
9. Vázquez-Canteli, J.R.; Nagy, Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Appl. Energy* **2019**, *235*, 1072–1089. [CrossRef]
10. Yu, L.; Sun, Y.; Xu, Z.; Shen, C.; Yue, D.; Jiang, T.; Guan, X. Multi-agent deep reinforcement learning for HVAC control in commercial buildings. *IEEE Trans. Smart Grid* **2020**, *12*, 407–419. [CrossRef]
11. Ruelens, F.; Claessens, B.J.; Vandael, S.; De Schutter, B.; Babuška, R.; Belmans, R. Residential demand response of thermostatically controlled loads using batch reinforcement learning. *IEEE Trans. Smart Grid* **2016**, *8*, 2149–2159. [CrossRef]
12. Wang, S.; Duan, J.; Shi, D.; Xu, C.; Li, H.; Diao, R.; Wang, Z. A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning. *IEEE Trans. Power Syst.* **2020**, *35*, 4644–4654. [CrossRef]
13. Zolfpour-Arokhlo, M.; Selamat, A.; Hashim, S.Z.M.; Afkhami, H. Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms. *Eng. Appl. Artif. Intell.* **2014**, *29*, 163–177. [CrossRef]
14. Sun, Z.; Piao, H.; Yang, Z.; Zhao, Y.; Zhan, G.; Zhou, D.; Meng, G.; Chen, H.; Chen, X.; Qu, B.; et al. Multi-agent hierarchical policy gradient for Air Combat Tactics emergence via self-play. *Eng. Appl. Artif. Intell.* **2021**, *98*, 104112. [CrossRef]

15. Barrett, E.; Linder, S. Autonomous hvac control, a reinforcement learning approach. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Porto, Portugal, 7–11 September 2015; pp. 3–19.
16. Overgaard, A.; Kallesøe, C.S.; Bendtsen, J.D.; Nielsen, B.K. Mixing Loop Control using Reinforcement Learning. In *E3S Web of Conferences*; EDP Sciences: Ulis, France, 2019; Volume 111, p. 05013.
17. Wei, T.; Wang, Y.; Zhu, Q. Deep reinforcement learning for building HVAC control. In Proceedings of the 54th Annual Design Automation Conference 2017, Austin, TX, USA, 18–22 June 2017; pp. 1–6.
18. Kazmi, H.; Suykens, J.; Balint, A.; Driesen, J. Multi-agent reinforcement learning for modeling and control of thermostatically controlled loads. *Appl. Energy* **2019**, *238*, 1022–1035. [CrossRef]
19. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017; pp. 66–83.
20. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
21. Van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-learning. *arXiv* **2015**, arXiv:1509.06461.
22. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2015**, arXiv:1511.05952.
23. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37. [CrossRef] [PubMed]
24. Buşoniu, L.; Babuška, R.; De Schutter, B. Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications-1*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–221.
25. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv* **2019**, arXiv:1911.10635.
26. Littman, M.L. Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2001**, *2*, 55–66. [CrossRef]
27. Lauer, M.; Riedmiller, M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of the Seventeenth International Conference on Machine Learning, Standord, CA, USA, 29 June–2 July 2000.
28. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 157–163.
29. Bowling, M.; Veloso, M. Multiagent learning using a variable learning rate. *Artif. Intell.* **2002**, *136*, 215–250. [CrossRef]
30. Crites, R.H.; Barto, A.G. Improving elevator performance using reinforcement learning. *Adv. Neural Inf. Process. Syst.* **1996**, 1017–1023.
31. Matarić, M.J. Reinforcement learning in the multi-robot domain. In *Robot Colonies*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 73–83.
32. Hu, J.; Wellman, M.P. Multiagent reinforcement learning: Theoretical framework and an algorithm. In Proceedings of the International Conference on Machine Learning (ICML), Madison, WI, USA, 24–27 July 1998; Volume 98, pp. 242–250.
33. Banerjee, B.; Peng, J. Adaptive policy gradient in multiagent learning. In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne, Australia, 14–18 July 2003; pp. 686–692.
34. Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; Basar, T. Fully decentralized multi-agent reinforcement learning with networked agents. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5872–5881.
35. Bertsekas, D. Multiagent reinforcement learning: Rollout and policy iteration. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 249–272. [CrossRef]
36. Wetter, M.; Zuo, W.; Nouidui, T.S.; Pang, X. Modelica buildings library. *J. Build. Perform. Simul.* **2014**, *7*, 253–270. [CrossRef]
37. El-Priser og Afgifter. Available online: https://www.vivaenergi.dk/el-priser-og-afgifter (accessed on 30 March 2021).
38. Nie, J.; Li, Z.; Kong, X.; Li, D. Analysis and Comparison Study on Different HFC Refrigerants for Space Heating Air Source Heat Pump in Rural Residential Buildings of North. *Procedia Eng.* **2017**, *205*, 1201–1206. [CrossRef]
39. Piechurski, K.; Szulgowska-Zgrzywa, M.; Danielewicz, J. The impact of the work under partial load on the energy efficiency of an air-to-water heat pump. *E3S Web Conf.* **2017**, *17*, 00072. [CrossRef]
40. Se Det gns. Varmeforbrug I husstande der Ligner Din. Available online: https://seas-nve.dk/kundeservice/forbrug/gennemsnitsforbrug/varmeforbrug/ (accessed on 30 March 2021).
41. Blad, C.; Koch, S.; Ganeswarathas, S.; Kallesøe, C.; Bøgh, S. Control of hvac-systems with slow thermodynamic using reinforcement learning. *Procedia Manuf.* **2019**, *38*, 1308–1315. [CrossRef]
42. Overgaard, A.; Nielsen, B.K.; Kallesøe, C.S.; Bendtsen, J.D. Reinforcement Learning for Mixing Loop Control with Flow Variable Eligibility Trace. In Proceedings of the IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, China, 19–21 August 2019; pp. 1043–1048.
43. Blad, C.; Kallesøe, C.S.; Bøgh, S. Control of HVAC-Systems Using Reinforcement Learning With Hysteresis and Tolerance Control. In Proceedings of the IEEE/SICE International Symposium on System Integration (SII), Honolulu, HI, USA, 12–15 January 2020; pp. 938–942.
44. Danmarks Statestik. Available online: https://www.statistikbanken.dk/bygv06/ (accessed on 30 March 2021).