# Multi-Agent Reinforcement Learning project: Talking Buildings

s3806715, s4013484, s3978389, s3799174

October 2024

## 1 Situation

Efficient energy management is increasingly critical in a world where electricity demand continues to rise alongside the need for sustainable consumption practices. Buildings, particularly in winter, require significant electricity for heating, which not only raises operational costs for owners but also places considerable strain on the power grid. With the ongoing construction of new buildings, the grid struggles to meet rising demand, leading to potential overloads and higher energy prices during peak periods.

This report explores a multi-agent reinforcement learning (MARL) approach to optimize heating systems in a shared power grid environment. We examine two different scenarios with two buildings each. The buildings require heating to maintain comfortable indoor temperatures during office hours. In the first scenario, one building follows the heating behavior of the building from our dataset while the other is managed by an autonomous agent responsible for its heating energy consumption, this agents only goal is managing temperature. In the second scenario, both buildings are controlled by agents and they have an additional goal. This goal is for the agents to learn to coordinate to prevent grid overloads and ensure efficient energy use, all without communicating. While communication could be helpful, it was mentioned explicitly that buildings will not communicate their data with other buildings. Instead, we expect buildings to adjust their behavior by including the grid load in the rewards through a load-adjusting energy price. So, "talking" in this context, refers to coordinating through the shared grid price.

## 2 Problem Description

Multiple buildings are consuming electricity in order to heat themselves. The goal for each building is to be at a comfortable temperature during opening hours, as well as minimize money spent on energy. Moreover, if multiple buildings are heating at the same time, the grid load will rise. A high grid load results in a high energy price, which is undesirable for the consuming buildings. Each building has to deal with temperature loss to the outside environment, the effect of heating on the inside temperature, and the energy consumption of turning on the heating and the price thereof. Other influences are office hours, hours where the building should be comfortably heated. In conclusion, the problem is that the buildings temperatures must be managed correctly, while reducing load on the grid. The data on which our simulation is based is supplied by the company AIMZ. It contains the energy consumption, inside temperature, and outside temperature, recorded every hour for the duration of three months (=92 days, = 2208 hours) in the winter.

### 2.1 Environment

The environment in this Multi-Agent Reinforcement Learning (MARL) project "Talking Buildings" simulates a shared power grid scenario with multiple buildings, each with its own heating demands and constraints. Here, we explore a world where two buildings must strategically manage their heating systems to stay warm during office hours, all while preventing an overloaded grid. The environment consists of a regulated outside temperature based on the data, as well as providing a way for the agents to heat the buildings. How the heating is modelled, is explained in Section 2.3. Furthermore, the environment loops through time and outside temperature, while keeping a global grid for each agent to use to decide when to consume energy or not. A formal description of the environment can be observed in Section 3.1.

### 2.2 Agents

There are three different kinds of agents in our simulation. Each agent acts as the internal heating manager for a single building. One agent, the *Data Agent*, is passive as it follows the heating behavior of the building in the dataset. The second agent is an active decision-making agent that learns solely based on building-temperature requirements. The third agent is also an active decision-making agent, though this kind of agent acts based on the energy price and the building temperature requirements. The active decision makers have a discrete action space of 15, the action is multiplied by 4 to end up as energy in kWh resulting in an action between 0 and 56 kWh. For a more clear explanation of the set-up of the experiment and the agents, see Section 5.2.

### 2.3 Sensors and sensor properties

Each agent has access to different pieces of data. Some of this data is local, describing the behavior of an agent's building, which is private to the agent. Other data is available to all agents, namely the energy price, as well as the outside temperature, and the time. These parameters can be considered public as they are known to every agent and can be seen as world parameters. The sensors of each agent record:

- The outside temperature in degrees Celsius (float) - In our simulation, this sensor is perfect in the sense that each building measures the exact same outside temperature at all times.
- The inside temperature in degrees Celsius (float).
- The power consumption in kWh (integer).
- The current time of the environment (24-hour cycle).
- The power grid of energy consumption with the corresponding price per kWh (float).

This results in an observation space of size 5, being a tuple defined according to Equation 1 for each individual agent $i$.

$$\mathcal{Z}^i = (temp_{out},\ temp^i_{in},\ kWh^i,\ time_{env},\ GridPrice_{kWh}) \tag{1}$$

## 3 Mathematical framework

To get a solution for the given problem of heat regulation of multiple buildings, we will consider the use of a partial observable stochastic game (POSG). This framework works with multiple

agents, multi-stage, multi-state, simultaneous moves, and most importantly for choosing the POSG: partially observable. However there are some subclasses of POSG which can be used to capture the environment more precisely (Section 3.1).

## 3.1 Subclass of POSG

All Multi-Agent Reinforcement Learning scenarios can be modelled as a Partially Observable Game. However, there exist subclasses of this, which can be exploited resulting in better performance of the agents.

The subclass of POSG relevant to our problem is a Mixed-Motive POSG. Other subclasses of POSG do not adequately represent the partially observable and mixed-motive environment in which our case occurs. We can formalize our problem by fitting it in the POSG tuple (n, $\mathcal{S}$, $\mathcal{A}$, $\mathcal{Z}$, p, r, o).

- **n** , the number of agents. In our game, n would equal 2 as we start with two buildings in the environment each regulated by their own agent. Two different experiments are done were each environment has 2 agents.
- $\mathcal{S}$ , the state set contains the hour of the day, the energy consumption per building, the inside and outside temperature per building, as well as the current energy price.
- $\mathcal{A}$ , the action set. For each agent, an action set is available as: $A^i$ of 15 actions. Resulting in a total action set of $A \equiv A^1 \times A^2$. The action itself is multiplied by 4 to end up as an energy consumption amount in kWh, which is fed into the environment.
- $\mathcal{Z}$ , the observation set. For each agent, an observation set is available as $\mathcal{Z}^i$, where the observation is of size 5 as explained in Section 2.3. Resulting in a $\mathcal{Z} \equiv \mathcal{Z}^1 \times \mathcal{Z}^2$. Where three values are captured from the global environment, and the other two from the agents' sensors.
- **p** , the transition function. Given the state of the whole environment $\mathcal{S}$ and the joint action $\mathcal{A}$ the transition function $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$. In our model, this transition function is being modeled by different formulas (building heating/cooling) and the grid model as explained in Section 2.1. These model equations can be observed in Equations 11, 12.
- **r** , the payoff/reward function. Can be formulated as $r^i = \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Resulting in a reward tuple: $r \equiv (r_1, r_2)$. This reward is based on the reached goal of temperature as well as the combination of the current energy price and the energy consumption, the design is explained in Section 5.1.2.
- **o** , the observation function. Given the state of the whole environment $\mathcal{S}$ and the joint action set $\mathcal{A}$ can be given as: $\mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{Z})$. This function retrieves the observation set from the state and action set. Within our framework, the observation function returns the tuple shown in Equation 1 by the environment for each agent.

## 3.2 Solution Concept

In this problem, we used Proximal Policy Optimization (PPO) to individually maximize each agent's reward consisting of an individual element (comfortable temperature), as well as a collective element (price). A detailed description of PPO can be found in Section 1. Because we used PPO, we cannot guarantee an equilibrium. Rather, we compute locally stable minima. We cannot ensure a Nash equilibrium as we do not know if each agent has converged to its global minimum

considering the other agents' policies. It is also difficult to converge towards a Nash equilibrium using decentralized training, as the policies of other agents are unknown to each learner and change throughout training without the coordination of a central learner. Similarly, a coordinated equilibrium is unlikely (and cannot be guaranteed) due to a missing coordinating device. As a result, it is likely that our approach approximates to find a local minimum than an equilibrium.

From the perspective of the problem description, it would be optimal to achieve a Nash equilibrium. This way, the agent could be ensured that they are always behaving optimally in their situation. This is relevant when applying the simulation to the real world, here we want to assure a client of using this strategy. Of course, this assumes that all other agents are also following the advice given by our Nash equilibrium. However, this would require each client to be transparent regarding their data and policies. It would also require us to implement a system such as regret matching. Sadly, we did not have sufficient time to implement such a system, which is also discussed in Section 6.2. For these reasons, we decided to approach this problem using PPO, aiming to produce locally minimized policies in a Distributed Training and Distributed Execution setting (DTDE) a more thorough explanation can be observed in Section 4.1.

### 3.3 Aimed Class of Policies

As the project is defined, the policies will be of a Mixed-motive nature. The policies cannot be cooperative, as each agent is rewarded individually for keeping its building's temperature within a target range. This means that agents have personal incentives to consume energy when needed, which may conflict with the collective goal of keeping the grid-load low. The policies cannot be competitive because of the aforementioned goal of keeping the grid load low. This is required to optimize each agent's energy consumption costs. Furthermore, as will be described in Section 4, the agents are implemented using PPO, resulting in stochastic policies.

## 4 Algorithm design

### 4.1 Paradigm

To elaborate on which algorithmic paradigm is most suited for our problem, each paradigm will be discussed.

In Central Training and Central Execution (CTCE), the agents would receive an action to perform from a central learner. It would know all temperatures of buildings, so it could define a planning strategy that would lower grid pressure and agent energy prices paid as much as possible. For these reasons, the central learner is a suitable option to the problem itself, as it could give the agents an optimal action to take in a certain point in time. However given the problem stated in Sections 1 and 2, the clients are interested in keeping as much data private as possible and retaining their autonomy. This would mean a central learner would go against the wishes of the client, as information has to be shared for it to work. As our goal is to provide a suitable solution for the client, the central learner and thus CTCE is not an option for our problem.
We find the same problem in Central Training and Distributed Execution (CTDE), as a central learner is involved in the training process. CTDE would thus also not be a suitable option.

In Decentralized Training and Centralized Execution (DTCE), no central learner is present. The agents learn themselves when to heat the building to get the highest reward, as they are trained on their own local observations and experiences. Furthermore, there is no sharing of parameters, gradients, or experiences between agents during the training phase. This decentralized training is

in line with our problem requirements, as there is no explicit information sharing necessary. For the execution phase, the execution of all actions will be coordinated through a central mechanism. This means that one central coordinator will collect information from all agents to make joint decisions, which allows for coordination strategies. We encounter a situation where information has to be shared, and the agents lose their autonomy. This means that DTCE is not a suitable paradigm, as the goal is to keep the agents autonomous and share as little information as possible.

The final option to be discussed is Decentralized Training and Distributed Execution (DTDE). In this paradigm, the distributed training is again suitable for our problem. Unlike DTCE, DTDE has distributed execution. This means that agents operate independently during execution, using only their local observations, without a central coordinator. This property fits well to the requirements of our problem, as it results in agents being in control of their own actions.

For these reasons, the algorithmic paradigm we chose the DTDE paradigm in our experiment. Each agent is working on his own local environment, which would be the building. Furthermore, there will be a global environment, the environment that regulates the buildings and the energy grid, as well as global variables. The agents do not have to share their information with others and are able to keep their autonomy. The paradigm thus closely fits the wishes of the clients.

## 4.2 Reinforcement Learning Algorithm: Proximal Policy Optimization

For this reinforcement learning problem, the Proximal Policy Optimization (PPO) (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) algorithm was used, which is an algorithm that optimizes a "surrogate" objective function using stochastic gradient descent. The key idea of PPO is to improve the stability of the training by avoiding large updates to the policy by using a clipped objective function. This ensures that the new policy does not differ too much from the old policy, which is done by taking the ratio between old and new probabilities, and clipping it to ensure that the update will not be too large.

In Algorithm 1 can be seen how a PPO agent was trained.

**Algorithm 1** Proximal Policy Optimization (PPO)

Initialize policy parameters $\theta$ and value function parameters $\phi$
**for** iteration $= 1, 2, \ldots$ **do**
  Run policy $\pi_{\theta_{\text{old}}}$ in the environment for T timesteps, collecting a set of trajectories $\mathcal{D}$.
  **for** each epoch **do**
    Randomly split $\mathcal{D}$ into mini-batches of size $M$
    **for** each mini-batch of trajectories $\mathcal{D}$ **do**
      Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
      Compute the policy ratio $r_t(\theta)$.
      Perform a gradient ascent step on $L(\theta)$ to update policy parameters $\theta$
      Update the value function by minimizing the mean-squared error.
      Perform a gradient descent step on $L^{VF}(\phi)$ to update value function parameters $\phi$
    **end for**
  **end for**
  Update old policy parameters: $\theta_{\text{old}} \leftarrow \theta$
**end for**

### 4.2.1 Clipped Surrogate Objective Function

The surrogate objective function $L(\theta)$ used in PPO ensures that the new policy $\pi_\theta$ does not deviate too much from the old policy $\pi_{\theta_{\text{old}}}$. The clipping term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ limits the change in policy to the range $[1-\epsilon, 1+\epsilon]$. For this implementation, $\epsilon$ was set to 0.2.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big], \tag{2}$$

where $r_t(\theta)$ is the ratio between the two policies, calculated as

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{3}$$

### 4.2.2 Advantage Estimation

The advantage function $\hat{A}(s,a)$ is used to quantify how much better or worse an action is compared to the average action taken in a given state. PPO uses estimates of this advantage function, denoted as $\hat{A}_t$, which are computed using Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{4}$$

where the TD residual $\delta_t$ is given by:

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t). \tag{5}$$

### 4.2.3 Value Function Update

The value function, which is modelled by the Critic, is updated by minimizing the mean-squared error between the predicted value and the empirical return ($V_t^{targ}$):

$$L^{VF}(\phi) = \mathbb{E}_t\left[\left(V_\phi(s_t) - V_t^{targ}\right)^2\right]. \tag{6}$$

### 4.2.4 Entropy Regularization Term

Entropy regularization is used to incentivize the agent to explore the action space more thoroughly. The entropy $H(\pi_\theta)$ measures the uncertainty of the policy $\pi_\theta(a|s)$. A higher entropy value indicates a more uncertain policy, which is coupled to more exploration, while a lower entropy value indicates a more certain (and possibly deterministic) policy.

By adding the entropy term to the policy objective, a certain level of randomness is maintained, which can help in discovering better policies and avoiding local optima. The coefficient $c_2$ controls the trade-off between the original objective and the exploration encouragement provided by the entropy term. A larger $c_2$ encourages more exploration, while a smaller $c_2$ allows the policy to focus more on exploiting known good actions.

The entropy of a policy is given by:

$$H(\pi_\theta)(s) = -\sum_a \pi_\theta(a|s)\log\pi_\theta(a|s) \tag{7}$$

The entropy regularization term is added to the actor policy objective, which is minimized during training through gradient descent, and can be written as:

$$L_t^{\text{Actor}}(\theta) = -L_t^{CLIP}(\theta) - c_2 H[\pi_\theta](s_t) \tag{8}$$

# 5 Implementation

## 5.1 Simulation

### 5.1.1 The Environment

To perform our experiments, a suitable environment for a multi-agent setting had to be designed. This required a model for the heating and cooling of a building based on energy consumption (kWh), as well as a method to model the dynamic price of electrical energy as a function of capacity.

**Building** The buildings need a method to heat and cool. To roughly achieve this goal we combined the specific heat capacity (see Equation 9) with generalized Newton's law of cooling (see Equation 10). We combined these two formulas into a rough model to regulate the buildings' temperature based on consumption as well as the environmental temperatures (see Equation 11).

$$Q = cm\ \Delta T \tag{9}$$

$$\Delta T = -k\ (T_{inside} - T_{outside}(t)) \tag{10}$$

$$\Delta T = \frac{Q}{cm} - k(T_{inside} - T_{outside}) \tag{11}$$

This formula contains two constants; $cm$, which decides how much the building will heat based on $Q$, which is used as a kWh, and $k$, which is a constant that regulates the rate of cooling down based on the difference between inside and outside temperatures. $cm$ is actually a combination of two constants, but it was treated as a single value since they are multiplied with each other in the equations. $c$ represents the specific heat constant and $m$ represents the mass.
These two constants are fit to the data, which was done by using SciPy curvefit function (Virtanen et al., 2020). With the use of this method in combination with the data that provides outside temperature, inside temperature, as well as the energy consumption (kWh), an estimate of the appropriate model for the building heat transfer could be made. In Figure 1 the curve shows a comparison between the inside temperature from the dataset and the inside temperature calculated by changing the temperature according to Equation 11, the outside temperature is also given for reference. The values for the constants that were found in order to obtain this graph were $cm = 1.08 \times 10^2$ and $k = 8.35 \times 10^{-3}$. The predicted temperature does not perfectly match the temperature from the dataset, this is most likely because the real building also consumes energy with other processes, such as lighting and cooling. The proposed equation only takes heating into account. Although it is not perfect, we think it is good enough in order to be able to say something meaningful about the agents that perform in this environment. There are still some complications regarding this, which will be discussed in Section 6.2.

**Power Grid**
The power grid determines the price of one kWh of energy consumption. The price is determined based on how much energy is consumed by all agents active in the shared environment. How the grid price is calculated, is shown in Equation 12. Here, *t_cons* represents total consumption, $cons_i$ represents consumption per agent, *cap* represents the grid capacity and *n_agents* represents the number of agents. This manner of grid price calculation results in behavior where the price changes slightly when total consumption is under the capacity, and it increases exponentially when total consumption is above the capacity. We do limit the price to 100, as it could otherwise keep growing to infinity, resulting in unexpected simulation behavior. The goal for the design of

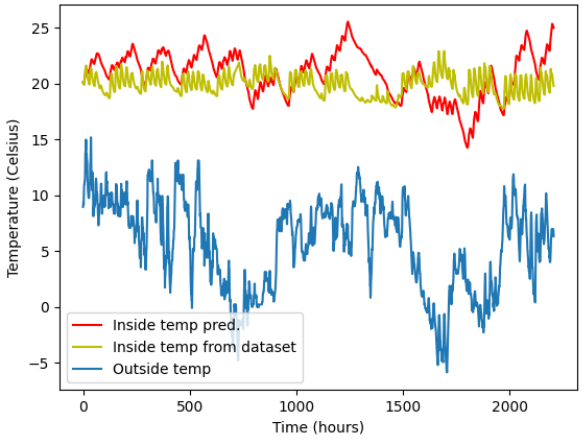


Figure 1: Curve fitting results. $cm = 1.08 \times 10^2$, $k = 8.35 \times 10^{-3}$. Where the outside temperature is plotted from the dataset.

this formula was to punish the agents more heavily when the grid capacity is exceeded, where the max capacity is equal to 25 kWh times the number of consuming agents. This value is chosen as the data showed that 25 kWh is plentiful to reach a comfortable temperature. For clarity, the grid price is plotted over total consumption in Figure 2.

$$\begin{aligned} \text{GridPrice}(t) &= \min(0.8 \times 10^{\text{t\_cons}-\text{cap}} + 0.05 \times \text{t\_cons} + 0.001, 100) \\ \text{where} \quad \text{t\_cons} &= \sum_{i=0}^{\text{n\_agents}} \text{cons}_i \\ \text{and} \quad \text{cap} &= \text{n\_agents} * 25 \end{aligned} \tag{12}$$

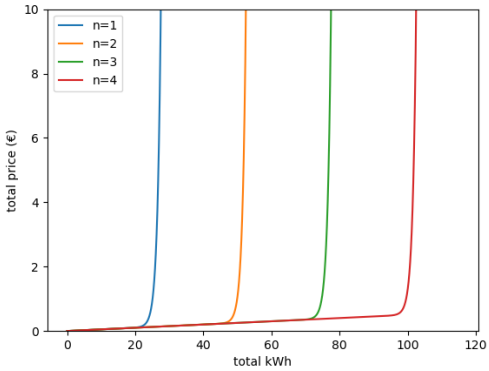


Figure 2: Grid price plotted over total consumption. Where n is the number of consuming agents in the environment.

### 5.1.2 Agents

**Reward:**
The rewards were designed to complete the two main goals. All rewards are chosen to be negative, as the two goals are both clearly defined boundaries that are more easily captured with negative rewards. The first, keeping the temperature near the comfortable temperature (the desired temperature) when employees can be present, should have a punishment for the agent when it deviates from the desired temperature. The second goal, minimizing money spent on energy, should have a reward that punishes the agent the more money is spent due to energy consumption. Our designed reward model is created with the inspiration of a similar reward structure from the research of Nguyen et al., 2024, where the problem also is described as the temperature control of a building.

The reward to accomplish the first goal was implemented as the *ComfortPenalty*, where temperatures deviating from the desired temperature of 20 degrees Celsius are punished when employees are present. Office hours were introduced, which go from 08:00 until 17:00, to define the times when employees can be present. As mentioned, a negative reward is given to the agent if the temperature of the building deviates within office hours, which increases exponentially the more it deviates from it to more heavily punish bigger deviations. This is modeled through a negative mean square error between the current temperature in timestep $t$ and the desired temperature. No punishment is given when it is outside the office hours, as then there is no reason to be within comfortable temperatures. The reward for timestep $t$ can be seen in Equation 13.

$$\text{ComfortPenalty}_i(t) = \begin{cases} -\left(\text{Temp}_{\text{in}}(t) - \text{Temp}_{\text{desired}}\right)^2, & \text{if OfficeHours}(t) = 1 \\ 0, & \text{if OfficeHours}(t) = 0. \end{cases} \tag{13}$$

The second goal was accomplished with a reward called *PricePenalty*. This reward is a punishment based on the amount of money the agent has to pay, where the amount of money is the consumption for the building in that timestep times the price of energy given by the grid at that timestep (see Equation 12). The parameter -0.5 was set to provide the negative reward and balance this parameter against the *ComfortPenalty*. The balancing is important, as otherwise, the agent could prioritize either reward above the other, which could produce unwanted agent behavior. Furthermore, the *PricePenalty* was capped at -100, to make sure that the reward for price would not overrule the first goal. The reward function can be seen in Equation 14.

$$\text{PricePenalty}_i(t) = max(-0.5 * \text{GridPrice}(t) * \text{consumption}_i(t), -50). \tag{14}$$

Combining both reward functions leads to the total reward function *TotalReward* given in timestep $t$, which is

$$\text{TotalReward}_i(t) = \text{ComfortPenalty}_i(t) + \text{PricePenalty}_i(t). \tag{15}$$

### 5.1.3 Reinforcement Learning Algorithms

In order to set up a multi-agent reinforcement learning environment, our agents which regulate the building temperatures have to learn a policy using a reinforcement learning algorithm. In this section, the specific implementation of the PPO agent will be explained and more detail on the hyperparameters and the design is given.

**PPO** Within our framework, we used PPO to update the agents' policies. We chose PPO as it leads to stable policy updates, which is a property that is useful within our problem, furthermore,

PPO is an algorithm that is easy to tune with its parameters as it does not have that many. Table 1 shows the used parameters for our 2 MARL agents.

| Parameter | value | description |
|---|---|---|
| $\epsilon$-clip | 0.2 | the new policy will stay within a range of 0.2 of the old policy. |
| $\gamma$ | 0.99 | for discounted reward |
| $\lambda$ | 0.95 | GAE parameter |
| *batchsize* | 256 | number of items to sample from experience |
| *nsteps* | 2024 | number of timesteps before updating the policy |
| *nminibatches* | 4 | number of policy updates after n steps |

Table 1: PPO specifics and hyperparameters

**Network** The network that PPO uses is an Actor-Critic network consisting of an actor output layer whose size equals the number of actions from which the log probabilities are calculated, as well as a critic output layer that predicts the state value. The actor and critic heads share two fully connected layers, the full architecture can be observed in table 2. Furthermore, the Adam optimizer is used to optimize the network, with a learning rate of 0.0001.

| Layer | Size |
|---|---|
| input size | 5 |
| fully connected 1 | (5, 128) |
| fully connected 2 | (128, 128) |
| actor output | (128, 15) *15 discrete actions* |
| critic output | (128, 1) *state value* |

Table 2: Actor-Critic architecture

**Training** During the simulation, we trained our agents by running through the environment. Every 2024 timesteps (so 2024 hours) the agent updates their policy by mini-batch training. This means that in total 4 policy updates will happen after 2024 timesteps. As PPO-clip is an on-policy RL algorithm, after every learn update the experiences are thrown away, and new experiences are gathered using the new policy.

## 5.2 Experiments

To validate our multi-agent reinforcement learning (MARL) implementation, we will compare it against two baseline agents (Section 5.2.1) to assess its effectiveness. Therefore a total of two different simulations will be held each containing two agents.

### 5.2.1 Baseline Agents

**Data Agent**
The first baseline is the Data Agent, which only consumes the kWh values provided by a dataset. This outputs both a price and a consumption graph, allowing us to compare the resulting costs to those of the MARL agents to determine if our method positively impacts the building's financial outcome.
**Temperature Agent**
The second baseline, the Temperature Agent, resembles the MARL agents but omits the price

penalty as seen in Equation 14 from its reward function, focusing instead solely on building comfort during office hours. Thus, only the comfort-related component of the reward, as shown in Equation 13, is used. This comparison will reveal whether including the power grid in the reward function reduces the building's electricity costs.

#### 5.2.2 Experimental Setup

There were two experiments. In the first experiment, two MARL agents were deployed in the same environment. In the second experiment, two baseline agents, one Data Agent and one Temperature Agent, were deployed in the same environment. Both experiments were run for 1000 epochs, enough to make the PPO algorithm converge to stable behavior, with each epoch involving the agents interacting with the environment and updating their kWh consumption's effect on grid load. During the experiment, the rewards, price spent on energy, and kWh consumption of each agent were captured for comparative analysis. To gain a more robust understanding of system performance, each experiment will be repeated five times, such that the mean and standard deviation of the results can be gathered.

# 6 Validation

In this Section, the results of the two experiments are displayed. Although the baseline experiment and the one with the MARL agents were performed in different environments, they are displayed together whenever possible, so that they can be compared easily.

Before the graphs with results were produced, first the produced data was checked for outliers. Two runs were removed from the results, one for both experiments (one for ppo_baseline and one for ppo_0), as they included models that produced suboptimal behaviour, with constant energy consumption leading to temperatures above 40 degrees Celsius.

## 6.1 Experiment Results

In Figure 3 the mean rewards of the three Reinforcement Learning agents are shown. The ppo_baseline agent is shown in a separate plot, because it uses a different reward function, which does not take the grid price into consideration. The reward function shows proper learning behaviour, where the rewards converge around the 400th epoch for the ppo_baseline agent and around the 800th epoch for ppo_0 and ppo_1. ppo_baseline's mean reward over the last 100 epochs is $-0.404$, ppp_0's is $-1.602$ and ppo_1's $-1.719$. Please note that the y-axis is a logarithmic scale.

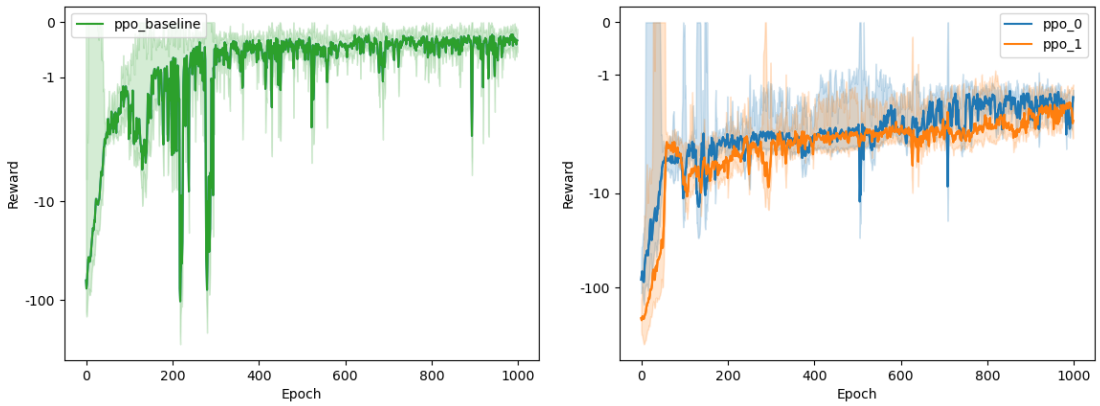


Figure 3: The mean rewards for the PPO agents on a logarithmic scale over the entire experiment. On the left the ppo_baseline's rewards are shown and on the right the PPO agents with price included in the reward. The lines show the mean over 4 runs per PPO agent, with the shaded area being the standard deviation.

Figure 4 shows the mean consumption for all four agents per epoch. The consumption of the agents changes a lot at the beginning of the runs, after which they stabilize when the agents are learning. The Data agent has a constant consumption for each epoch, because it is not learning, so its behavior does not change. ppo_baseline's mean consumption over the last 100 epochs is 13.186 and the Data agent's is 13.169. The PPO agents with reward functions that include the price penalty consume less, as ppp_0's mean consumption over the last 100 epochs is 12.810 and ppo_1's 12.721.

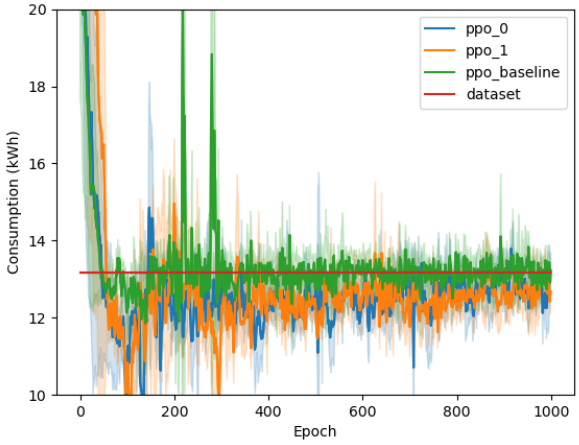


Figure 4: Mean consumption for all agents of the entire experiment. The blue, yellow, and green colored lines show the mean consumption over 4 runs per PPO agent, with the shaded area being the standard deviation. The red line shows the mean consumption for the dataset, which is stable.

Figure 5 shows the mean total energy cost for all the agents, so how much money they spend on average during a full epoch. ppo_baseline's mean total energy cost over the last 100 epochs is 1.765 and the Data agent's is 2.811. The PPO agents with reward functions that include the price penalty spend slightly less than ppo_baseline, as ppp_0's mean consumption over the last 100 epochs is 1.666 and ppo_1's 1.509.

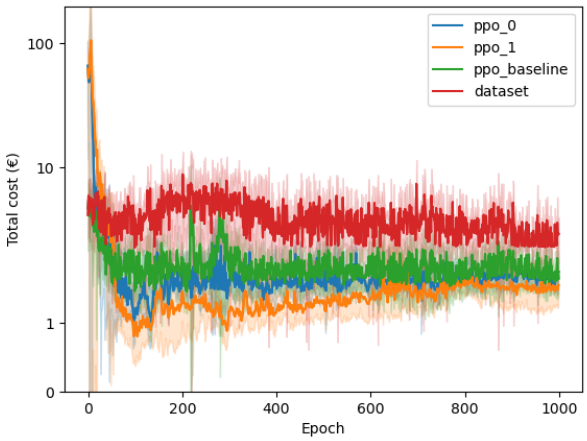


Figure 5: Total energy cost for all agents on a logarithmic scale of the entire experiment. The lines show the total energy cost per epoch over 4 runs per agent, with the shaded area being the standard deviation.

Figure 5 shows the mean money spent per kWh consumed for all the agents. The Data agent paid the most per kWh by far, with a mean over the last 100 epochs of 0.213. ppo_baseline's ending mean is 0.134, ppo_0's mean is 0.130 and ppo_1's ending mean is 0.119. So, the PPO agents with a price penalty in their reward function tend to pay slightly less per kWh.

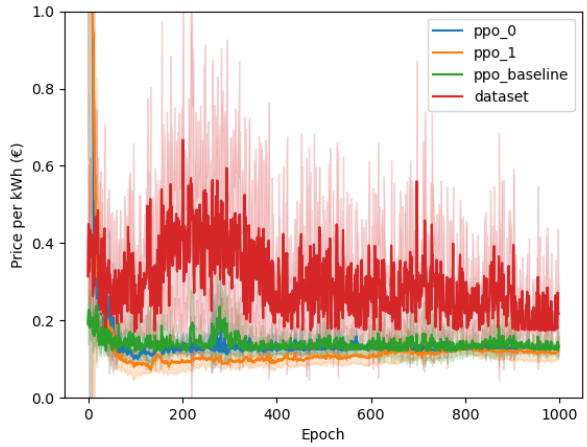


Figure 6: Price per kWh for each agent.

Figure 7 shows the mean inside temperatures of each agent during their final epochs. The agents aim to keep their temperature close to 20 degrees Celsius during office hours, two lines at 18 and 22 degrees are shown as a reference for desirable inside temperatures during office times. All agents generally stay within the comfort range through the full epochs. The Data agent and the ppo_1 agent do get outside the range briefly at times.

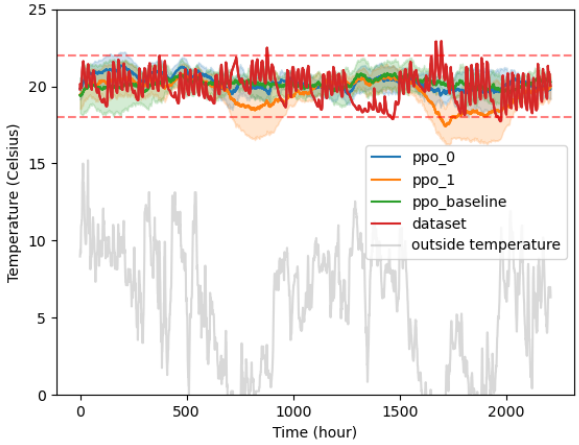


Figure 7: Mean temperature for all agents over the final epoch. The colored lines show the mean temperature per time-step over 4 runs per agent, with the shaded area being the standard deviation. The gray line shows the outside temperature.

Figure 8 is a zoomed-in version of Figure 7, here the mean temperatures over the first week of the last epoch is shown. The grey areas indicate the office hours. The Data agent shows a rapid decrease in temperature outside of office hours, while all three PPO agents tend to keep their temperatures more stable.

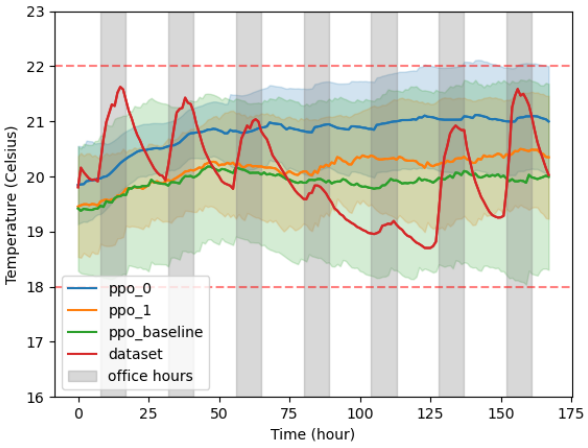


Figure 8: Mean temperature for all agents over a week of their final epochs. The coloured lines show the mean over 4 runs per agent, with the shaded area being the standard deviation. The gray line shows the outside temperature.

## 6.2 Discussion

The results show that the PPO agents were able to learn to stabilize the building temperature around the desired temperature. Furthermore, they also indicate that the PPO agents paid less for the energy than the dataset baseline did. However, there are multiple implications based on these results, which will be discussed in this Section.

The grid price as shown in Equation 12 was created with a certain idea of how much energy a building would need to consume to remain at a certain temperature. This is based only on the building in our dataset, as no parameter for the maximum energy capacity of the grid was available. If this is present, it means that the grid price equation should be changed. Furthermore, the prices that were aimed for in this project are not representative of real prices. When implementing this method in a real-world environment, a careful re-evaluation of this formula should be performed with available data of grid capacity and energy prices.

There is no separation of different days in the environment, as only hours are counted. This means that the model does not fit for weekends (which can be seen in Figure 8) or holidays (see dip in Figure 7 between hours 1300 and 1500) as the dataset model does. On the one hand, this is a flaw of our environment and especially the comparison with the dataset, as there is a clear difference in goals. On the other hand, it shows that our model has even better performance when looking at mean consumption (Figure 4) and total energy cost (Figure 5). The MARL agents show a decrease in both compared to the baseline, which is an indication that the agents found a more efficient heating strategy. However, this indication should be taken with a grain of salt. The total consumption of the dataset most likely also includes the energy needed for the lighting and cooling of the building, although the latter is minimal in the cold winter months. So even though there is a reduction in consumption of the PPO agents, it might be only due to these other costs. Furthermore, using the dataset as a baseline has complications on its own, as the buildings that the PPO agents control have different dynamics than the one in the dataset. As shown in Figure 1, the cooling and heating of the two types of buildings are different. This means that

comparisons are harder and have less strength. However, it provides a certain amount of realism to the simulation. If this method is used in real life, different buildings would also have different dynamics. It shows that the simulation is also able to work on more realistic problems.

It could be that the method implemented might not be optimal to solve our problem. If time allowed it, we were interested in adding a regret-based algorithm to solve the problem, as there are indications in the problem that this would have better outcomes. In a regret-based approach, agents would be able to communicate in the past tense, where they look at what other agents have already communicated (the amount of energy they consumed) and adjust their learning on this hindsight accordingly. In further research, we highly recommend looking into this approach. Another thing we were aiming to introduce at the beginning of the project, was a dynamic price for energy during the day. This would be a representation of the general energy price dictated by a bigger source of consumers than just two buildings. In this way, we could provide a way for the buildings to keep the grid pressure low together (which is what we have implemented) as well as look at the energy price fluctuations themselves. The idea was that it would stimulate the agents more to consume energy when prices are low. However, it proved very difficult to incorporate this with online available data as well as functions we created. For this reason, we have chosen to let the energy price be only decided by the grid price formula (Equation 12). This does mean that the prices seen in Figures 5 and 6 are only an indication of what the price could be and not a very accurate simulation of the real-world dynamic grid prices. The function of these figures is to show the difference in cost between the baselines and MARL agents, which is only influenced by the total energy consumption of the agents in the experiments at a point in time.

As clearly shown in Figure 8, all the PPO agents do not show a decrease in temperature outside office hours, as the building of the dataset shows. For the PPO_baseline, this is expected behavior because it is only focused on getting the temperature near the desired temperature of 20 degrees Celsius. For the other two PPO agents, it is unexpected behavior. These agents appear to prioritize reducing the *ComfortPenalty* over the *PricePenalty*, as they show no different behavior than the PPO_baseline. The comparisons between the MARL agents and the baseline show that either the balancing between the rewards is not done correctly, as the punishment for consuming energy appears to be negligible compared to deviating from the desired temperature, or the building environment is not correctly modeled, as the agent is able to keep the temperature near desired values with minimal energy consumption.

As can be seen in the results, the baseline where the impact of the grid was measured, the PPO_baseline agent, has similar outcomes to the MARL agents. At first glance, this is an indication that in the reward function, the *ComfortPenalty* gets a higher priority than the *PricePenalty*, as excluding the *PricePenalty* did not lead to different results. However, it is also likely the case that the PPO baseline did not have the intended effects when simulated together with the dataset agent. It was likely able to learn the other agent his consumption, as that is a stable factor, and plan accordingly.

A key factor that may have impacted our ability to reach the expected temperature fluctuations during office hours is that our simulation doesn't account for other heat sources. During office hours, the building likely benefits from occupant-generated heat, which could help maintain warmth and reduce electricity costs. As well as other environmental changes that could help the building cool/heat, think of the wind or sun hours. Although this was intendedly left out of the simulation, as it was aimed to be as simple as possible, when making a more robust model these factors should be taken into account.

Outside of office hours, the buildings didn't cool down as much as we anticipated. We initially thought the price penalty would have a stronger impact here (since the *ComfortPenalty* is always

0 outside office hours). Additionally, the buildings didn't have any cooling mechanisms, making it challenging to lower the indoor temperature. This shows signs that if our building can cool down better, a reduction in energy price might be expected as currently, the building is keeping its temperature quite steady at 20 °C which seems like unnecessary energy consumption for heating.

The problem of optimization is also something to be taken into account when evaluating the current results of the MARL implementation. The PPO agents haven't been optimized due to time and resource constraints. PPO has numerous hyperparameters, shown in Section 5.1.3. When further implementing and optimizing this approach of MARL for the energy consumption regulation of buildings a grid search should be held to optimize all these hyperparameters, this could lead to better results as all current parameters are either standard parameters or accepted whenever we encounter a learning PPO implementation.

Besides the PPO algorithm, the DQN algorithm was also implemented. However, this algorithm has been shown to produce worse outcomes in earlier stages of the project, which is why it was chosen not to pursue this direction.

# 7 Conclusion

In this report, we have explored a multi-agent reinforcement learning (MARL) approach to optimize energy consumption for heating in two buildings while maintaining comfortable indoor temperatures during office hours. The aim was to provide a 'proof-of-concept' for a MARL environment. The results show that agents using the PPO algorithm in a DTDE paradigm are able to learn the modeled environment and keep building temperatures near the desired temperature while minimizing cost.

In the experiments, two buildings were simulated in the same environment. The choice for only two buildings was based on the fact that we wanted to show the buildings are able to cooperate in the environment, resulting in a lower load on the electrical grid. This was a success, as our modeled agent paid a lower price per kWh compared to the dataset agent. While this is promising, it should be taken into account that our simulation is a strong simplification and that the dataset agent had different environmental circumstances than the modeled agent. A following step could be to expand the environment to incorporate more (diverse) buildings. If the rewards are balanced correctly, we expect a bigger impact from the *PricePenalty* when more buildings are incorporated. With more buildings, the impact one building has on the entire *GridPrice* diminishes, meaning it becomes more stable. As the environment becomes more stable and accurate—through refined reward balancing, grid price calculations, and improved modeling of building heating and cooling dynamics—it allows agents to learn more effectively, resulting in more stable policies. This, in turn, provides a clearer basis for drawing conclusions about desired behavior. We can, however, conclude that incorporating grid price into the environment does have an impact, as evidenced by the observed decrease in price per kWh over time. This supports the proof of concept, which can be further explored in future research.

# References

Nguyen, A. T., Pham, D. H., Oo, B. L., Santamouris, M., Ahn, Y., & Lim, B. T. (2024). Modelling building hvac control strategies using a deep reinforcement learning approach. *Energy and Buildings*, *310*, 114065. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0378778824001816` doi: https://doi.org/10.1016/j.enbuild.2024.114065

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

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2