Athens University of Economics and Business

Master in Computer Science
Department of Informatics
Athens, Greece

Master Thesis
in
Computer Science

Residential Energy Consumption Optimization through Reinforcement Learning

Odyssefs Diamantopoulos-Pantaleon

Co-Supervisors: Professor George C. Polyzos
Professor Iordanis Koutsopoulos

Examiner: Professor Vasilios A. Syris

November 2023

Odyssefs Diamantopoulos-Pantaleon
Residential Energy Consumption Optimization through Reinforcement Learning

November 2023

Athens University of Economics and Business
Master in Computer Science
Department of Informatics
Mobile and Multimedia Laboratory
Athens, Greece

I would like to dedicate this thesis to my loving parents who have always supported me and guided me through the highs and lows.

Abstract

We propose a novel Residential Energy Manager (REM) that suggests actions to the user about the entire household. To be more precise, the REM views the status of every electrical device within the residence along with many specific attributes for each of the devices, like its energy cost and the duration of its function, and then makes a suggestion based on the current electricity price and the habits of the user for all the devices. In the literature, there have been several approaches that differ in the way they define the environment. Some approaches allowed the REM to initiate actions on its own; however, this is often not feasible due to infrastructure constraints and can also frustrate the user, since positive or negative feedback stems from the resident's decision to turn off or ignore the REM initiated appliance. Other researchers preferred to assign REMs to clusters of devices or to each individual device separately, instead of deploying a single REM for the entire household. Although these approaches make the problem less complex and thus easier to solve, we consider that the supervision of the household by a single REM creates possibilities for greater performance since it will be able to form spatial and temporal correlations within the household and better understand the uniqueness of the resident's behavior. To create the REM we use Reinforcement learning (RL), which seeks to maximize the notion of cumulative reward and to explain the way in which intelligent agents ought to perform actions in an environment. RL is especially useful when applied to complex and dynamic environments, such as households, because of the various types and numbers of devices encountered. Some need time to finish their function, while others can be turned on or off at any time. Additionally, each resident has a unique personality and behavioral habits, which further increases the complexity. RL has a lot of potential in this area because it can explore the environment of the household and exploit it by adapting to user feedback, resulting in better appliance management that also takes into consideration the resident's character and needs. We further propose and open source, unlike other works in the literature, a generalized environment that utilizes RL and maintains core parameters that can be applied in almost all real-case scenarios, which can act as the foundation on which future researchers can build their own scenarios. Moreover, we experiment with many state-of-the-art algorithms. More specifically, we conducted experiments using the Advanced Actor Critic (A2C), the Proximal Policy Optimization (PPO), and the Monotonic Advantage Reweighted Imitation Learning (MARWIL) RL

algorithms in four different households with different complexities, created based on our approach. The aim was to verify the effectiveness of the proposed REM in shifting energy consumption away from electricity price peaks and to understand the limitations of the RL algorithms in our proposed complex solution. Through our extensive experiments with various decision-making intervals, episode lengths, and different RL algorithms, we conclude that the MARWIL RL agent shows promising performance in the most complex household, successfully shifts power consumption when needed, understands how to use different types of devices, and manages to adapt to the resident.

Περίληψη

Προτείνουμε έναν νέο Residential Energy Managers (REMs) που προτείνει στον χρήστη ενέργειες για ολόκληρο το νοικοκυριό. Για να είμαστε πιο ακριβείς, ο REM βλέπει την κατάσταση κάθε ηλεκτρικής συσκευής εντός της κατοικίας μαζί με πολλά συγκεκριμένα χαρακτηριστικά για κάθε μία από τις συσκευές, όπως το ενεργειακό κόστος και η διάρκεια λειτουργίας της, και στη συνέχεια κάνει μια πρόταση με βάση την τρέχουσα τιμή της ηλεκτρικής ενέργειας και τις συνήθειες του χρήστη για όλες τις συσκευές. Στη βιβλιογραφία έχουν υπάρξει διάφορες προσεγγίσεις που διαφέρουν ως προς τον τρόπο που ορίζουν το περιβάλλον. Ορισμένες προσεγγίσεις επέτρεπαν στο REM να ενεργοποιεί από μόνο του ηλεκτρικές συσκευές, ωστόσο, αυτό συχνά δεν είναι εφικτό λόγω περιορισμών που επιβαλλουν οι οικιακές υποδομές. Επίσης μπορεί να εκνευρίσει τον χρήστη, δεδομένου ότι η θετική ή αρνητική ανατροφοδότηση προέρχεται από την απόφαση του κατοίκου να απενεργοποιήσει ή να αγνοήσει τη συσκευή που ξεκίνησε το REM. Άλλοι ερευνητές προτίμησαν να αναθέσουν REM σε ομάδες συσκευών ή σε κάθε μεμονωμένη συσκευή ξεχωριστά, αντί να αναπτύξουν ένα ενιαίο REM για ολόκληρο το νοικοκυριό. Παρόλο που οι προσεγγίσεις αυτές καθιστούν το πρόβλημα λιγότερο πολύπλοκο και συνεπώς ευκολότερο στην επίλυσή του, θεωρούμε ότι η εποπτεία του νοικοκυριού από ένα μόνο REM δημιουργεί δυνατότητες για μεγαλύτερες επιδόσεις, καθώς θα είναι σε θέση να διαμορφώνει χωρικές και χρονικές συσχετίσεις εντός του νοικοκυριού και να κατανοεί καλύτερα τη μοναδικότητα της συμπεριφοράς του κατοίκου. Για τη δημιουργία του REM χρησιμοποιούμε το Reinforcement learning (RL), το οποίο επιδιώκει να μεγιστοποιήσει την έννοια της σωρευτικής ανταμοιβής και να εξηγήσει τον τρόπο με τον οποίο οι ευφυείς πράκτορες οφείλουν να εκτελούν ενέργειες σε ένα περιβάλλον. Το RL είναι ιδιαίτερα χρήσιμο όταν εφαρμόζεται σε πολύπλοκα και δυναμικά περιβάλλοντα, όπως τα νοικοκυριά, λόγω των διαφόρων τύπων και αριθμών συσκευών που συναντώνται. Ορισμένες χρειάζονται χρόνο για να ολοκληρώσουν τη λειτουργία τους, ενώ άλλες μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν ανά πάσα στιγμή. Επιπλέον, κάθε κάτοικος έχει μοναδική προσωπικότητα και συνήθειες συμπεριφοράς, γεγονός που αυξάνει περαιτέρω την πολυπλοκότητα. Το RL έχει πολλές δυνατότητες σε αυτόν τον τομέα, επειδή μπορεί να εξερευνήσει το περιβάλλον του νοικοκυριού και να το αξιοποιήσει προσαρμόζοντας την συμπεριφορά του με βάση την

ανατροφοδότηση του χρήστη, με αποτέλεσμα την καλύτερη διαχείριση των συσκευών που λαμβάνει επίσης υπόψη τον χαρακτήρα και τις ανάγκες του κατοίκου. Επιπλέον, προτείνουμε και διαθέτουμε ελεύθερο κώδικα, σε αντίθεση με άλλους στη βιβλιογραφία, ένα γενικευμένο περιβάλλον που χρησιμοποιεί το RL και διατηρεί βασικές παραμέτρους που μπορούν να εφαρμοστούν σχεδόν σε όλα τα σενάρια πραγματικών περιπτώσεων, το οποίο μπορεί να λειτουργήσει ως βάση πάνω στην οποία οι μελλοντικοί ερευνητές μπορούν να δημιουργήσουν τα δικά τους σενάρια. Επιπλέον, πειραματιζόμαστε με πολλούς αλγορίθμους τελευταίας τεχνολογίας. Πιο συγκεκριμένα, πραγματοποιήσαμε πειράματα χρησιμοποιώντας τους αλγορίθμους RL Advanced Actor Critic (A2C), Proximal Policy Optimization (PPO) και Monotonic Advantage Reweighted Imitation Learning (MARWIL) σε τέσσερα διαφορετικά νοικοκυριά με διαφορετική πολυπλοκότητα, που δημιουργήθηκαν με βάση την προσέγγισή μας. Στόχος ήταν να επαληθευτεί η αποτελεσματικότητα του προτεινόμενου REM στη μετατόπιση της κατανάλωσης ενέργειας μακριά από τις κορυφές της τιμής της ηλεκτρικής ενέργειας και να κατανοηθούν οι περιορισμοί των αλγορίθμων RL στην προτεινόμενη σύνθετη λύση μας. Μέσω των εκτεταμένων πειραμάτων μας με διάφορα διαστήματα λήψης αποφάσεων, μήκη επεισοδίων και διαφορετικούς αλγορίθμους RL, καταλήγουμε στο συμπέρασμα ότι ο πράκτορας MARWIL RL παρουσιάζει υποσχόμενες επιδόσεις στο πιο πολύπλοκο νοικοκυριό, μετατοπίζει με επιτυχία την κατανάλωση ενέργειας όταν χρειάζεται, κατανοεί πώς να χρησιμοποιεί διαφορετικούς τύπους συσκευών και καταφέρνει να προσαρμόζεται στον κάτοικο.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Professor George C. Polyzos, for his continuous and valuable guidance and support during the preparation of this thesis and in my studies in general, and to Professor Iordanis Koutsopoulos for his helpful comments and constructive criticism. I am also very grateful to PhD candidate Spiros Chadoulos for the technical guidance and knowledge he provided me and the countless conversations we had, helping me to complete this thesis successfully. I would also like to thank all the members of the Mobile and Multimedia Laboratory (MMlab) at the Athens University of Economics and Business, as their research activities pave the way for newcomers, like myself, to study in this amazing field. I also want to thank Nikos Ipiotis, founder and CEO of Plegma Labs, for his continuous support throughout my thesis. Finally, I would like to express my gratitude to my university colleagues and friends Platonas Karageorgis, Panos Kaliosis, and Dimitris Mamakas, as well as my dear friend Ilias Stogiannidis, for the support they provided me over the last year.

Contents

1 Introduction

1.1 Motivation and Problem Statement

Throughout modern human history, one of the most precious and sought-after resources has been energy. Energy resources are essential to our society, and without them, the world as we know it would not be able to function. An energy resource can be non-renewable or renewable. Non-renewable resources are those that are going to eventually run out or will not be replenished in our lifetimes. Currently, almost 80% of our energy production is produced by burning fossil fuels, because they are extremely energy rich and relatively cheap to extract and process [7]. On the other hand, renewable energy resources are abundant, but they are harder and more expensive to harvest [14]. The excessive use of non-renewable resources has severely damaged the local environments and directly contributed to the emission of greenhouse gases that damage the atmosphere of the planet.

The last decade societies have made a huge shift toward renewable energy, dedicating a lot of resources into a new electricity grid that will be able to smoothly integrate many sources of renewable energy and other advanced technologies. The new grid is called "Smart Grid" and aims to solve many of the problems of the old grid by combining information gathering and communication technologies with power system engineering. One of the core aspects of the Smart Grid is the Internet of Things (IoT). Within the grid, there are many information collecting devices (smart meters) that provide real-time monitoring and management of the grid nodes. Smart meters help the grid avoid critical failures by detecting them in an early stage, and also enable decentralized energy generation, which opens up a lot of new possibilities.

The introduction of smart meters can greatly influence current societies because it can help one of the largest and most important energy consumers, the residential sector, which according to the Eurostat service, in 2020 represented 27% of the final energy consumption [6]. More specifically, smart meter data gathering paves the way for the integration of artificial intelligence, which, thanks to the recent technological progress, can now be applied to large-scale scenarios. Machine learning is rapidly growing and its finding more and more applications in various fields. It provides researchers and engineers with tools that help them solve complex problems more

efficiently both time and budget wise. These models have shown their capabilities in popular fields such as natural language processing , computer vision and forecasting systems and are becoming more popular in Smart Grid related applications.

Reinforcement learning is a subcategory of Machine learning that has been really popular among researchers in the Smart Grid, because it specializes in solving complex problems that exist in dynamic environments. One of those problems that directly influences the residential sector is the appliance management problem in households. The appliance management problem refers to the scheduling of devices in order to take advantage of low electricity prices that might arise at certain hours of the day due to the abundance of renewable energy or due to the low electricity demand. The complexity of this problem is high because each household is different and contains many devices with different attributes. Some require a lot of time to function, while others consume a lot of energy and need to be carefully scheduled. There are also many types of devices. Priority devices, such as the air conditioning or the lights, can be turned on or off at any time, while deferrable devices, such as the dishwasher and the washing machine, need to operate continuously to finish their functions. In addition, devices can have temporal or spatial relationships that should be taken into account. For example, some kitchen devices are likely to operate at the same time when the resident wants to prepare a meal. Finally, the resident is another variable that adds even more complexity to the household environment. The unique behavior and habits of the resident influence the appliance management schedule and add the additional challenge of adaptation. The explorative nature of Reinforcement learning along with its ability to adapt based on feedback to its environment, make it an ideal Machine learning technique to use to solve this problem. Therefore, researchers have employed Reinforcement learning to create Residential Energy Managers that can be installed in households and help residents lower their electricity bills by shifting their electricity usage to off-peak hours where electricity prices are lower.

In our work, we introduce a Residential Energy Manager (REM) that utilizes Reinforcement learning to adapt to each individual household and help reduce its electricity costs. More specifically, the REM acts as a consultant that continually suggests actions for every device in the house to the resident and uses the responses as feedback to improve itself. Additionally, to make informed decisions, we give it access to information on the status of all devices in the house and the price of electricity. Moreover, we present a flexible gym simulation environment that can effectively represent most types of households and other larger buildings, such as malls. This environment contains two of the most popular types of devices, priority (lights, air condition, etc.) and deferrable devices (dishwasher, laundry machine, etc.) and allows the user to add specific details associated with each machine. Furthermore, it is easily customizable and will allow future researchers to edit it

according to their needs. Finally, we test the Advanced Actor Critic (A2C), the Proximal Policy Optimization (PPO), and the Monotonic Advantage Reweighted Imitation learning (MARWIL) Reinforcement learning algorithms in four different environments that represent households with different complexities and conclude that the MARWIL algorithm is the best of the 3 and can successfully adapt to users and helps them optimize their appliance management.

1.2 Novelties

Our approach goes beyond the current state-of-the-art. In our work, we use the agent as a suggestive REM. In other words, we do not let it initiate jobs on its own, but only suggest them to the user, who then makes the final decision. By doing this, we make sure that the agent is trained not only on reducing costs, but also on making the user feel as comfortable as possible and adapting to each individual's personal needs. The first novelty that we propose is that the supervision of the entire household and all its devices by the RL agent allows the REM to make new types of connections between devices that might also be unique to each user. To be more specific, the agent can formulate spatial and temporal correlations. For example, it is possible to understand that opening the lights in the kitchen can lead to another device being turned on in the same area (spatial correlation), or it can derive that the user likes opening the Air Conditioner after returning from work (temporal correlation). The second novelty of our work is the creation of an open-source [1] generalized environment that maintains core parameters that can be applied in almost all real-case scenarios. We envision this environment as a foundation on which future researchers can build their own scenarios, making it an important academic tool. The last novelty that we propose is the incorporation of user transition probabilities into the environment, which helps model different resident recommendation adoption patterns. User transition probabilities greatly assist the training process and enhance the agent's capability to adjust to diverse profiles and situations, which is a fundamental attribute of RL.

1.3 Thesis Structure

In **Chapter 2** we provide all the background information necessary to understand the content of the thesis, review the literature on the research area of data analytics and Machine learning in the smart grid sector, and present some of the most important research articles. In **Chapter 3** we present our approach and showcase the results of the experiments we carried out.

[1] project repository

2 Background and Related Work

2.1 Background

2.1.1 Artificial Intelligence

Artificial intelligence (AI) has been in the spotlight over the last few years and has infiltrated many different fields. Its applications are becoming more and more ambitious, surpassing the expectations experts had decades ago. For example, there are algorithms that examine medical records or can drive cars! There are many definitions for AI. According to IBM, "In its simplest form, artificial intelligence is a field that combines computer science and robust data sets to enable problem solving" [9]. However, AI itself is a huge and chaotic field that is constantly evolving. It includes categories such as Automation, Computer Vision, Robotics, and Machine learning [3]. In this report we review papers that utilize certain Machine learning techniques. Therefore, we are going to present background information for the three subcategories of Machine learning, which are called Supervised, Unsupervised, and Reinforcement learning.

Supervised learning:

In Supervised learning, we use training data that have been labeled by a human or some other technique. These algorithms undergo a self-improvement process within a data set, where they receive accurate feedback following their predictions, enabling them to adjust their internal parameters for improved correctness in subsequent predictions. The fundamental concept here is that by engaging in this cycle of predictions and corrections, the model can grasp the inherent patterns and connections within the input data and the desired output. With this acquired knowledge, it can effectively make accurate predictions for unseen data. Supervised machine learning models are usually preferred for classification and regression problems [16].

Unsupervised learning:

In Unsupervised learning, we do not have labeled data. Data sets contain structured data, and Unsupervised learning is tasked with noticing patterns and grouping these data points without any external guidance. Although these models are generally more unpredictable than Supervised learning (2.1.1), they can detect patterns that were previously unknown. Additionally, they require less time to train since there is no need for someone to label all the data that the model needs for training. Data clustering belongs to this family of learning methods.

Reinforcement learning:

Reinforcement learning (RL) is an autonomous, self-teaching system that essentially learns through trial and error. It is based on the problem of learning from an interaction to achieve a goal [20]. In other words, the *agent* interacts with the *environment* and the *environment* responds with new situations that test the agent. Furthermore, the *environment* gives rewards to the agent depending on the action taken. More specifically, according to Sutton [20]: "the agent and environment interact at each of a sequence of discrete time steps, $t \in \{0, 1, 2, 3, \ldots\}$. At each time step t, the agent receives some representation of the environment's state, $S_t \in S$, where S is the set of possible states, and on that basis selects an action, $A_t \in A(S_t)$, where $A(S_t)$ is the set of actions available in state S_t. One time step later, in part as consequence of its action, the agent receives a *numerical* reward, $R_{t+1} \in R \subseteq \mathbb{R}$, and finds itself in a new state, S_{t+1}. At each time step, the agent implements a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's *policy* and is denoted π_t where $\pi_t(a|s)$ is the probability that $A_t = \alpha$ if $S_t = s$. Reinforcement learning methods specify how the agent changes its policy as a result of its experience. The agent's goal, roughly speaking, is to maximize the total amount of reward it receives over the long run". In other words, Reinforcement learning is based on giving rewards when the model acts according to our wishes and handing out penalties when its actions are unwanted. As we have previously mentioned, the goal of RL is to train an RL agent to complete tasks within an environment. That environment can be known or unknown. There are algorithms that are efficient when the agent knows beforehand the available actions and their rewards / penalties, and others that are used when an agent is required to also explore the surroundings [4]. Today, the applications that use RL are limited compared to applications that utilize Supervised and Unsupervised learning. However, there is a lot of untapped potential, especially in the field that concerns us, where the model aims to learn each individual and adapt to their preferences and lifestyle, while at the same time trying to reduce electricity prices. This is because by the constant interaction of the *agent* with the *environment* the agent can learn more

complex tasks. In the next subsections we will discuss five essential RL concepts and give more information about the most used Python RL libraries. To be more precise, we cover the State, Action, Reward, Policy, Episode, and present the Gymnasium and Ray-RLlib libraries, which we used to shape our project.

State:

The State is a fundamental concept that encapsulates the current situation or condition of an agent within an environment. It contains information about the environment and the agent. For example, in the case of a robot trying to navigate itself out of a room, the State could contain the robot's coordinates, its distance from the walls, its battery percentage, etc. The part of the State that the agent perceives is called Observation. In some scenarios, we want the agent to know everything that happens in the environment. For example, in applications where an agent is designed to perform actions in a specific environment that does not change dramatically, it makes sense for the agent to have perfect information. In these situations the State equals the Observation. However, in some applications where we want the agent to explore new environments, we prefer to leave it with imperfect information. In other words, we hide information from the agent and the Observation of this agent is a subset of the State.

Action:

The Action refers to the decision made within the environment. In other words, the RL agent decides to perform an action based on the Observation. After the Action is taken, a new Observation is created. In the robot example, the robot takes a step and new coordinates or battery percentage is calculated, which are shown in the updated State and Observation.

Reward:

The Reward dictates the policy that the RL agent follows. The Reward is calculated based on the Reward formula, which encapsulates the goal of the RL agent. It connects the different States and Actions and helps the RL agent to determine the most profitable Actions to take. In the robot example, an example Reward formula could be the Euclidean distance between robot and the door which we want it to exit. In a video game setting, the RL agent could be trying to finish the game without losing. We can safely say that choosing the Reward Formula is not an easy process and should be carefully thought out since its vital for the success of the RL algorithm.

Policy:

The Policy represents the strategy of the RL agent. In other words, its the action plan that its using to interact with its environment. Typically, RL agents want to maximize the Reward they get from their Actions, but at the same time they want to try new State-Action combinations. This is called the exploration versus exploitation dilemma. The agents usually try to find a healthy balance, exploring the environment in the first steps and then exploiting it when they have gathered more information. There are many different policies with different strengths and weaknesses. We will present some of them in the following sections.

Episode:

An episode refers to a single sequence of interactions (Actions) between an RL agent and its environment from a starting State to a terminal State. It represents one complete run or trial of the RL algorithm, during which the agent takes actions based on its policy, receives feedback in the form of Rewards, and observes the resulting State transitions. At the end of an Episode, the agent updates its policy based on the experiences and the Rewards he received and starts a new one.

Gymnasium:

Gymnasium[1] is a Simulation Environment library designed to facilitate the development and testing of Reinforcement learning (RL) algorithms. Developed by OpenAI, it provides a standardized framework for creating and benchmarking various RL agents across a wide range of tasks. With its user-friendly API and extensive documentation, Gymnasium has become a popular choice in the Artificial Intelligence community to accelerate the progress of RL research and applications. Gymnasium has been extremely useful in our work, helping us implement our approach and create the proposed household simulation environment. The most fundamental concept in the Gymnasium library is the Gymnasium Space [2]. Spaces are used to place boundaries on the variables used for Action and Observation spaces. In other words, depending on the Space type, an Action can have e.g. four distinct values, a numerical upper and lower bound or a binary vector. More specifically, there are four types of Spaces:

1. **Discrete Space:** A Discrete Space represents a finite set of discrete elements. The Discrete Space is commonly used in problems with a finite number of possible actions, such as playing games with discrete moves.

[1]Gymnasium website : https://gymnasium.farama.org/
[2]Gymnasium Spaces website : https://gymnasium.farama.org/api/spaces/

2. **Box Space:** A Box Space represents a continuous n-dimensional space. It is defined by specifying a range of possible values for each dimension. These spaces are used to represent continuous states such as the speed of a car.

3. **MultiDiscrete Space:** A MultiDiscrete Space is a combination of multiple Discrete Spaces. It represents a collection of discrete elements, each belonging to a separate Discrete Space.

4. **MultiBinary Space:** A MultiBinary Space represents multiple binary elements. It is used when dealing with a set of binary values, where each element can take on two possible states.

Gymnasium spaces are a very useful tool and are essential for RL Simulation Environments.

Ray-RLlib:

Ray-RLlib[3] is a scalable, state-of-the-art Reinforcement learning (RL) Python framework. It includes the latest and most known RL algorithms, which are used to learn and solve problems within custom-made Gymnasium (2.1.1) environments. These algorithms use policies to select actions, and given a policy, rollouts throughout an environment produce sample batches of experiences. Training steps can also be customized for each individual RL experiment. RLlib supports both PyTorch and TensorFlow, which are the most popular Deep learning frameworks. It also allows highly distributed learning, batched and parallel environments, and independent learning of neutral or coexisting agents. Another key feature of RLlib is its ability to save and load RL algorithms and policies. Checkpoint objects can be used to store and load the current state of an Algorithm or Policy and the neural networks weights within these structures. In terms of hyperparameter tuning, RLlib integrates with Ray Tune[4], a distributed hyperparameter search framework for deep learning and RL. It is based on grid search and uses early stopping techniques, including the Median Stopping Rule and HyperBand. RLlib also provides examples for various use cases and features, such as multi-agent and hierarchical training, weight sharing between policies, and GPU examples. In general, RLlib is a powerful tool for RL, offering a wide range of features and capabilities that make it suitable for academic research and industrial applications.

[3]Ray-RLlib : https://www.ray.io/rllib
[4]Ray Tune : https://docs.ray.io/en/latest/tune/index.html

2.1.2 Reinforcement learning Algorithms

In this section, we will analyze the Reinforcement learning algorithms that we experimented with. These algorithms include the policy-based algorithm called Proximal Policy Optimization (PPO) (2.1.2), which is one of the most basic RL algorithms in the literature, Advanced Actor Critic (A2C) (2.1.2) which combines both value-based and policy-based methods and is also one of the most well known RL algorithms, and the Monotonic Advantage Reweighted Imitation Learning (MARWIL) (2.1.2) which, like A2C, combines value-based and policy-based methods, but performs better at more complex tasks.

Proximal Policy Optimization:

Proximal Policy Optimization (PPO) is a state-of-the-art algorithm in the field of Reinforcement learning, specifically designed to address stability and sample efficiency in training deep neural network policies. PPO builds upon the Actor-Critic framework but focuses primarily on improving the policy (actor) while using the value function (critic) to estimate advantages and evaluate actions. The key idea behind PPO lies in its careful handling of policy updates to prevent large policy changes that could lead to instability during training. The actor in PPO represents the policy, which is typically parameterized as a deep neural network. It learns to map states to actions and aims to maximize the expected cumulative rewards over time. Instead of making significant policy updates, PPO employs a surrogate objective function that measures how much the updated policy deviates from the previous while still ensuring performance improvement. The critic in PPO is responsible for estimating the advantages of different state-action pairs. It helps the actor guide policy updates by providing valuable feedback on expected future rewards. The critic utilizes the value function approximation, often implemented using a neural network, to estimate the state's value. The advantages derived from these value estimates allow PPO to emphasize actions that are more likely to yield higher rewards, leading to better learning efficiency.

Advanced Actor Critic:

Actor-Critic Reinforcement learning (RL) is a powerful and widely used approach that solves complex sequential decision-making tasks. It combines the strengths of both policy-based methods (the actor) and value-based methods (the critic) to efficiently learn and optimize the agent's behavior in uncertain environments. The actor represents the policy function, which maps states to actions and is typically implemented as a parametric function approximator, such as a neural network.

The critic plays the role of an evaluator or an approximator of value functions [5]. Its purpose is to estimate the expected future rewards that the agent can obtain from each state. By providing feedback on the value of different states, the critic helps the actor make more informed decisions. Advantage Actor-Critic (A2C) is an implementation of the previous explanation. The difference between A2C and Asynchronous Advantage Actor-Critic (A3C) is that A3C released by Deep-Mind in 2016 was praised for its simplicity and speed. It uses multiple independent networks, each with its own weights, that interact with a different copy of the environment in parallel. This architecture helps A3C explore a wider part of the state action space in much less time. In general, A3C performs much better than all other RL algorithms on standard RL tasks.

Monotonic Advantage Reweighted Imitation learning:

The Monotonic Advantage Reweighted Imitation Learning (MARWIL) is a Reinforcement learning (RL) agent that combines value- and policy-based methods, proposed by Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang in their 2018 paper titled "Exponentially Weighted Imitation Learning for Batched Historical Data" [21]. In most RL settings, the learner interacts with the environment through a simulator, which acts as the environment oracle. The simulator provides feedback to the learner, allowing it to adjust its policy based on the rewards it receives and the state transitions it observes. However, the MARWIL RL agent is designed for deep policy learning with only batched historical trajectories. More specifically, the learner does not have access to an environment oracle. Instead, it learns from a batch of historical data, which includes past states, actions, and rewards. MARWIL is applicable to problems with a complex nonlinear function approximation and works well with hybrid (discrete and continuous) action spaces. The method does not rely on the knowledge of the behavior policy, which means that it can be used to learn from data generated by an unknown policy. Under mild conditions, the MARWIL algorithm, despite its surprising simplicity, has a policy improvement bound and empirically outperforms most competing methods.

2.2 Related Work

In this section, we conduct an initial review of the literature on the existing work on data analytics and Machine learning in the smart grid sector. More specifically, we present research papers that cover consumer profiling, consumer engagement, and appliance scheduling.

[5] Article url : https://theaisummer.com/Actor_critics/

The first research papers we present showcase an area of the literature that focuses on the consumer. Some of them devise models that provide an accurate and efficient way to categorize consumers based on certain characteristics, while others focus on improving consumer engagement through various incentives. The paper named Household Energy Consumption Segmentation Using Hourly Data published in 2014 by Stanford members has made an impact in this field [11]. This research paper tried to make good use of smart meter data, suggesting a unique clustering process that would help improve the targeting of the Energy Efficiency (EE) and Demand Response (DR) programs (depicted in 2.1). More specifically, they used data provided by Pacific Gas and Electric Company, and applied an adaptive clustering process of K means[2], followed by hierarchical clustering to correctly summarize the results and merge small clusters that may correlate[6]. Then they performed an entropy analysis to find how random and varied are the decisions of certain customers and create specific consumer profiles. The research paper reaches the conclusion that consumer profiles that use entropy can offer many benefits when it comes to proposing programs that could suit each customer's lifestyle and help reduce or shift energy consumption.

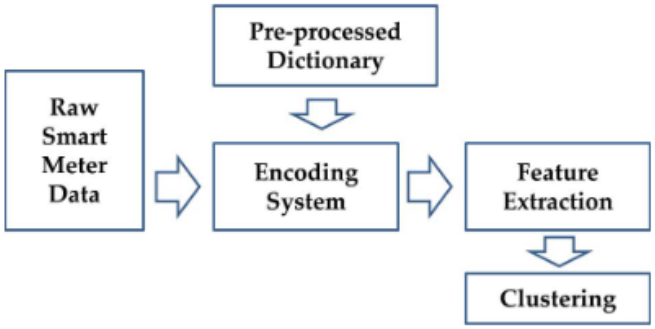

Fig. 2.1: High level overview of the system presented by the [11] research paper

In 2015 McLoughlin et al. published a research paper in the Applied Energy journal called A Clustering Approach to Domestic Electricity Load Profile Characterization using smart metering data[12]. This paper criticized previous Profile Creation (PC) models (i.e., models that were used to create Consumer Profiles out of certain data) that lumped together households, which used electricity in different ways. Their study was conducted on the Irish data set and proposed a new way of creating profiles, filling the literature gap by clustering based on the Time of Use (ToU) for a large sample of residential customers over a period of six months. More specifically, among the most popular clustering algorithms (k-means,k-medoid and self-organizing maps), with the use of the Davies-Bouldin (DB) index, they identified that the self-organizing maps suited the data more and then followed a three-step process. Firstly, they clustered each day of the 6-month period separately, which

[6]https://www.displayr.com/what-is-hierarchical-clustering/

made sure that the seasonal components of the electricity use could be captured. Then, for a particular day, each cluster's electricity demand was averaged in order to create a daily electricity load profile for a cluster. Small clusters were merged, which led to the creation of certain PCs. Furthermore, they assumed that each customer could be using a different PC each day, and thus they recorded every day the PC that each customer used in a separate index called the Customer Class Index (CCI). Finally, they used a multinominal logistic regression that determined the likelihood of a certain household with specific characteristics to act in a similar way to one of the PCs they had created. This process can be seen graphically in 2.2. They concluded that it is possible to classify most customers according to their characteristics, without any prior knowledge of the consumption of household electricity.

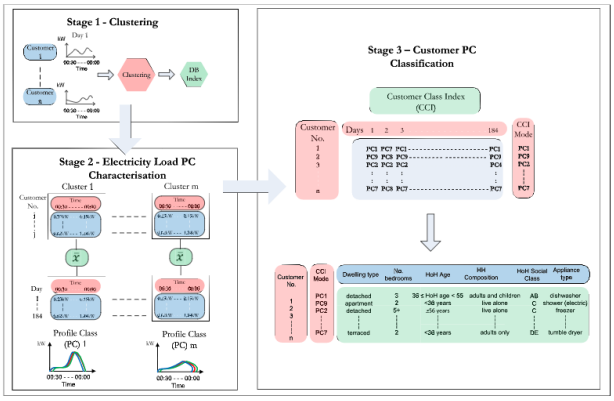

Fig. 2.2: Proposed method of the [12] research paper

Another paper that is worth looking at is titled Analysis and Clustering of Residential Customers energy Behavioral Demand using smart meter data written in 2016[8]. To begin with, the paper had two clear objectives. The first goal was to cluster consumer profiles in a way that would help distribution network operators (DNOs) to correctly identify customers who could be suitable for energy management solutions, and the second goal was to find some links between publicly available information, such as the location of the household and other sociodemographics, and energy behavioral usage. One of the important contributions of the paper is the notion that not all data collected by smart meters[7] is valuable. In order to reach these goals, the researchers conducted a detailed analysis on the Irish data set and then used Finite Mixture Models[8] for clustering. Additionally, they decided to check the output reliability by using Bootstrapping, which was something new in the power systems literature at the time. They concluded that their final clusters could identify many of the important behaviors that customers exhibited and could also locate consumers with high variability, all of which could significantly help DNOs create a better long-term plan for their networks.

[7]Smart meters are electronic devices that record information, like electric energy and more
[8]https://www.stata.com/features/overview/finite-mixture-models/

A recent paper that was published in 2023 by Spyros Chadoulos, Iordanis Koutsopoulos and George Polyzos is named Deep4Ener[19]. The research paper proposed an approach where a single Deep learning (DL) model was trained on multiple consumers. They claimed that this model could capture shared patterns and make accurate predictions even in the case of erratic consumption characteristics. The high-level approach can be seen in the 2.3 figure. Initially, they used a double clustering process to group consumers based on their energy profiles. In the next step, a Deep Neural Network (DNN), which contains an Recurrent Neural Network (RNN) encoder[9] and an Multi Layered Perceptron (MLP)[10], took as input the results of the double clustering technique and the consumption of the last 24 hours or one week to produce the predicted consumption. The RNN has the ability to maintain an internal memory that can capture temporal characteristics, which increased the performance of the MLP, which was tasked with making the final prediction. It is also important to note that this approach can tackle the cold-start problem, i.e., when a model is trained on a user that does not have a lot history, the model tends to have really poor performance. The paper finally states that this approach has managed to mitigate the cold start problem, outperform most other state-of-the-art approaches, and achieve scalability and generalization.

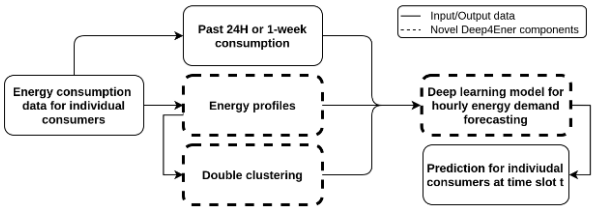

Fig. 2.3: High level architecture of the Deep4Ener approach of the [19] research paper

The next paper we present emphasizes on consumer engagement. The research paper is titled Mobile Apps Meet the Smart Energy Grid: A Survey on Consumer Engagement and Machine Learning Applications[5]. According to researchers, consumers are the main actors in the electricity grid and are extremely important, as residential buildings represent 22% of the total energy demand. Therefore, they considered it essential to engage and educate them through the creation of an application for mobile devices. More specifically, in their paper, they presented the architecture of their system, which consists of the Internet of Things (IoT) infrastructure (smart meters, etc.), the mobile application, and the back-end server, which received information from the IoT infrastructure, processed it and sent it to the mobile app (2.4). Additionally, they argued that there were some key design features that made consumers care. These features included information provision, a reward system, social networks, a user performance status with leader boards

[9]RNN : https://www.ibm.com/topics/recurrent-neural-networks
[10]MLP : https://www.simplilearn.com/tutorials/deep-learning-tutorial/multilayer-perceptron

and badges, and, most importantly, a friendly user interface that was easy to learn. Furthermore, they described and critically evaluated previous mobile applications that were designed with the same goal. They concluded that if the Machine learning and recommender system components were integrated into those apps, they could reach much higher levels of engagement and energy savings. They also believed that creating accurate profiles for each user / household would further increase energy savings because the mobile app could make even more personalized offers to consumers.

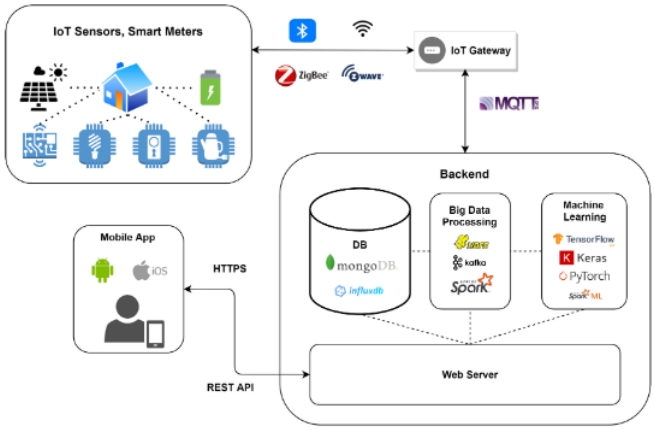

Fig. 2.4: Example of a smart grid system architecture taken from [5]

Our thesis is very close to the topic of appliance scheduling. Appliance scheduling focuses on creating scheduling managers that help consumers, either by reducing electricity bills or improving their quality of life, as more actions are automated.

The first paper that we present is called Optimal Control Policies for Power Demand Scheduling in the Smart Grid, published in 2012 [10]. The authors presented a demand load management system that would move non-emergency power demands at off-peak times. Their aim was to reduce the operating costs of the grid and the electricity bills of consumers. More specifically, their contribution lies in the two on-demand scheduling policies that they proposed for their model: Threshold Postponement (TP) and Controlled Release (CR). TP is the policy with which the controller decides to serve a new demand request either immediately or with a certain delay that depends on the current power consumption. On the other hand, CR serves requests immediately if their power consumption is lower than a certain threshold set by the user or the electricity company; otherwise, it places them in a queue that will make them wait until their power consumption drops. Based on their evaluation, they showed that these policies helped make their model better than other solutions that were developed at the time. The high-level architecture of the system is shown in 2.5.

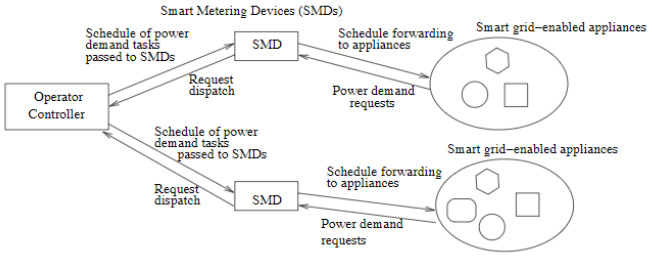

Fig. 2.5: High level overview of the system in [10]

Another interesting approach was proposed by a group of researchers from the National Technical University of Athens who published the Non-Intrusive Load Monitoring for Personalized Strategies Towards Energy Consumption Reduction[17]. Their goal was to provide personalized energy saving tips for user's schedules. To do that, they proposed a series of steps that would also aim at respecting the privacy of households. Firstly, they fed the data gathered from smart meters to an Non-Intrusive Load Monitoring (NILM) algorithm that tried to analyze the appliance usage. The external weather data and the results of the NILM algorithm were then passed to a data analytics engine that produced user-specific energy consumption patterns. Finally, they created personalized messages based on the information produced to help consumers improve their energy behavior (2.6).

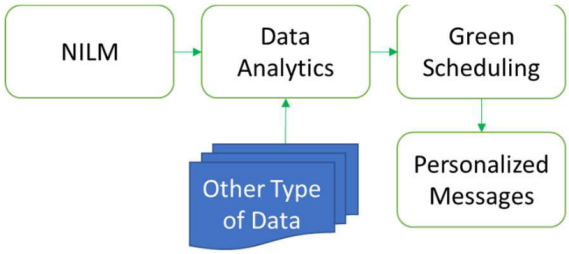

Fig. 2.6: System pipeline of [17]

Reinforcement learning has been popular among researchers in the Smart Grid, because it specializes in solving complex problems that exist in dynamic environments, such as the appliance management problem. Consequently, researchers conducted research that utilized Reinforcement learning to tackle it. The next research papers that we present are closely related to the subject of our thesis and use Reinforcement learning for appliance scheduling.

The first article we present is called "Residential Demand Response using Reinforcement learning" and was published more than 13 years ago in 2010[15]. ONeill et al. proposed an EMS named Consumer Automated Energy Management System (CAES) that took into account both electricity pricing and user demands, as seen in Figure 2.7. The Reinforcement learning model gets as input the State, which

includes the consumer reservation and the energy pricing sequence, and outputs an action that allocates energy to devices. CAES was considered innovative at the time because it first introduced the idea of learning the resident and shifting the energy peaks. Furthermore, according to their evaluation, CAES reduced the financial cost of the end user by a $16 - 40\%$ margin. However, the State that it presented was relatively simple, it made many assumptions for the consumer that were later deemed questionable by other researchers, and it only experimented with Q-learning. In our thesis, we present a more complex State, a more sophisticated modeling of the user, and experiment with a lot of state-of-the-art Reinforcement learning algorithms in many different household scenarios. Furthermore, we accept direct feedback from the user to boost the model's performance.

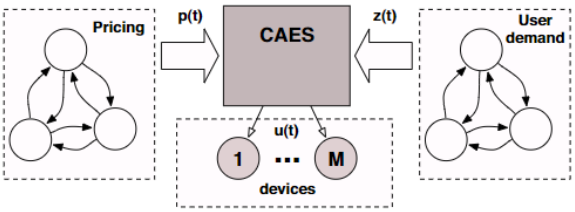

Fig. 2.7: CAES system from [15]

Zheng Wen et al. published a research paper [22] that outlined the problems of state-of-the-art research on Demand Response (DR) and Residential Energy Manager systems (REMs) and proposed their own solution to these challenges. The authors criticized the work of O'Neil et al.[15] by questioning the feasibility of discovering disutility functions for each residence or device and argued that a good REM would self-initiate appliances, since most energy-consuming activities occur without the user's input. The proposed approach aimed to learn user and device disutility functions and allowed for user and REM-initiated jobs. More specifically, the State (2.1.1) of the RL agent included grid signals (such as the energy price), requests, cancellations, and feedback from the resident (2.8). The user could request appliances to start in specific time frames or cancel existing device functions that were unwanted. Additionally, systematic evaluation of these actions was expected in order to further boost the model's learning. The objective of the agent was to minimize the reward function that combined user dissatisfaction with the electricity cost of scheduled appliances. Furthermore, the authors used the divide-and-conquer tactic and classified the devices into clusters, which helped the algorithm to converge faster. Lastly, they conducted experiments with the Q-learning algorithm, which showed promising results. Our approach differs because we offer a more complex State, we use a model that suggests actions for the entirety of the house and learns directly through user feedback, which includes accepting and rejecting appliances, and we test more than one household with state-of-the-art Reinforcement learning algorithms.

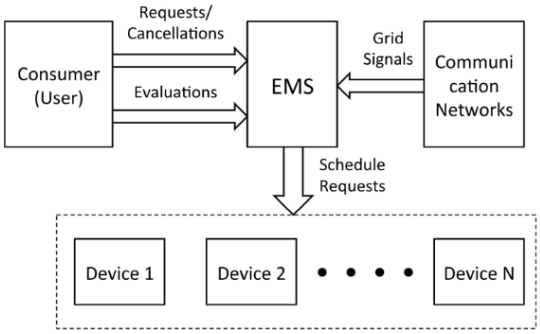

Fig. 2.8: Environment of [22]

Another EMS research paper was featured in IEEE Transactions on Smart Grid at 2019 titled On-Line Building Energy Optimization using Deep RL[13]. The authors made an interesting introduction to Deep Reinforcement learning (DRL), which is essentially the combination of Deep learning and Reinforcement learning. Their contribution was that they found a way to adapt DRL algorithms to the smart grid context and investigated some of the most popular DRL algorithms, that is, Deep Q-learning (DQN) and Deep Policy Gradient (DPG). The authors wanted to reduce load peaks and minimize energy costs in several buildings. They also assumed that end users could produce energy and sell it back to energy providers at a certain price. Another hypothesis they have is the existence of three types of device consumption profiles. The time-scaling load assumes that the device can be turned on and off a certain number of times during the optimization period (e.g., air conditioner). The time-shifting load devices had to consume a certain amount of power over a period of time (e.g. dish washer). The last profile is the time-scaling and time-shifting load that combines both and refers to devices such as the electric vehicle charger. The State (2.1.1) that they employ is simple. It contains only the energy consumption of the building and the price of electricity at any given moment. Figure 2.9 shows the proposed approach architecture, which also contains a simple Deep Neural Network. The State that they proposed included the prices that the user could sell and buy energy, and the state of the device. They also proposed an interesting Reward formula where the model tries to learn to optimize its performance on multiple tasks. They tested their architecture on the Pecan Street data set and evaluated both DPG and DQN. The research paper concluded that both DRL methods were much better than traditional RL algorithms. Lastly, they provided evidence (2.10) that showed how much better the DPG algorithm performs compared to current systems. In this thesis, we present a more complex State, we do not use the third type of devices that they introduced, we use a different Reward formula that aims to push the Reinforcement learning agent to learn to correctly use the different types of devices in the household and satisfy the user's unique demands, and we perform a thorough experimentation with different Reinforcement learning algorithms. Furthermore, we

take direct feedback from the user and let the model adapt to the unique habits of each resident.

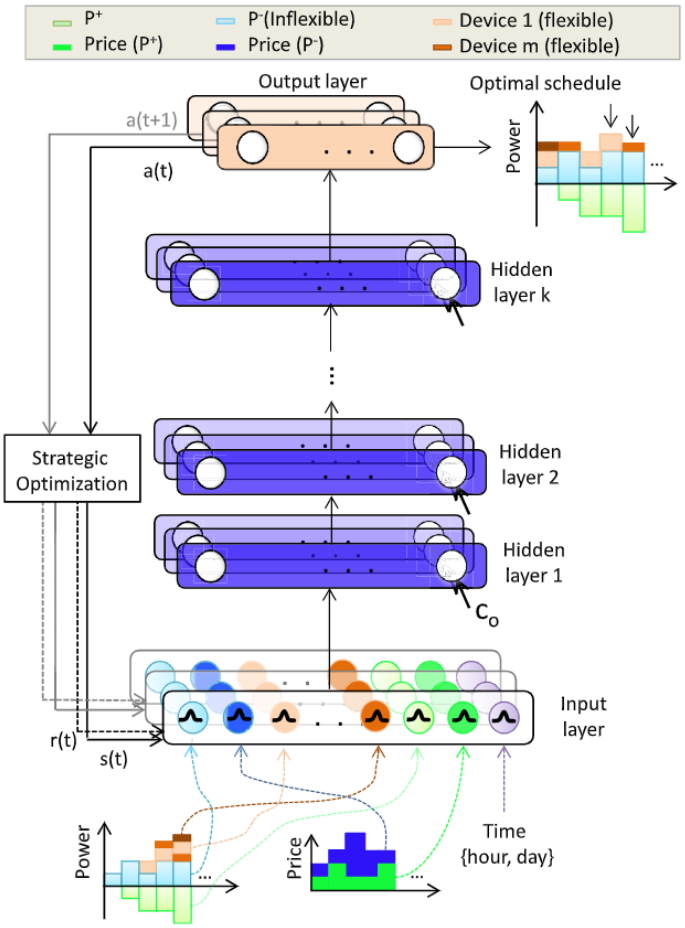

Fig. 2.9: Proposed Approach Architecture from [22]

All of the research papers so far have focused on finding the optimal solution to a single agent problem. A multi-agent approach was presented by Ahrarinouri et al. on the IEEE Transactions on Industrial Informatics journal at 2021[1] that tried to present efficient ways for multiple agents to work toward the same goal. This makes the agent learning process more difficult and unique because, apart from their environments, agents must also take each other into account. This leads to extremely unstable environments that make convergence extremely difficult. The household in which they experimented had solar panels, access to the electrical grid, and was able to use natural gas. Additionally, this home had four agents who worked together to reduce energy costs as much as possible by changing between the available energy from renewable sources and the two energy grids they were connected to. The four agents were a combined heat and power unit, a plug in an electric vehicle, several solar panels, and a water heater (2.11). The reward function varied between agents, but the objective function encouraged smart energy

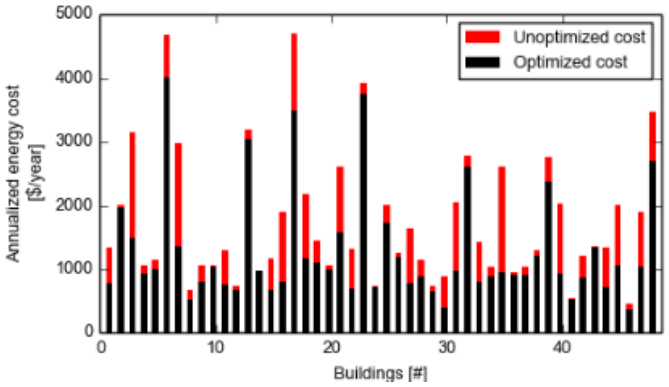

Fig. 2.10: Yearly savings per buildings when cost optimization is performed at the aggregated level on 48 buildings using Deep Policy Gradient (DPG) method [13].

consumption. The authors also tested the decision-making process of agents in shorter time intervals than most articles. They experimented with 10-60 minute time intervals, whereas most papers only tried 1-hour intervals and used Q-learning for the training process. In figure 2.12 we can see that the more often the agents made decisions, the better the results. They concluded that their model successfully shifted electricity consumption to hours when electricity was cheaper and used the combined heat and power unit when the price of natural gas was lower. In our thesis, we consider that the model suggests actions for the entire household. Furthermore, we take direct feedback from the resident and improve the model based on it. Lastly, we experiment with many state-of-the-art Reinforcement learning algorithms in different household scenarios with different hyperparameters, including different time intervals and episode lengths.

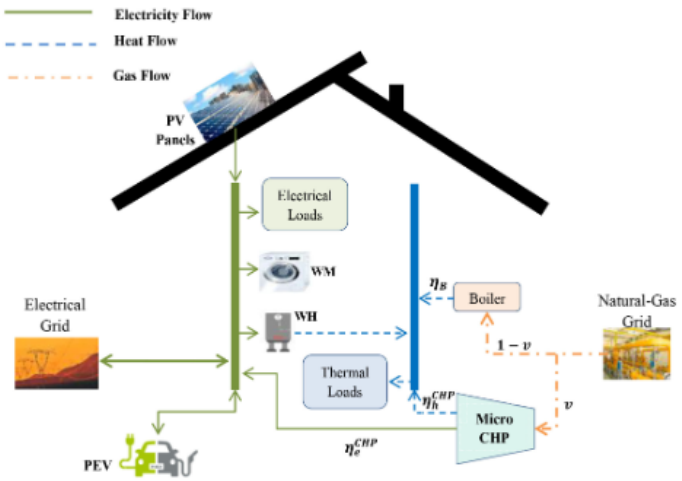

Fig. 2.11: Residential energy system schematic [1]

The last article we examined was published in 2022 and is considered state-of-the-art. The research paper has the title Home Energy Recommendation System

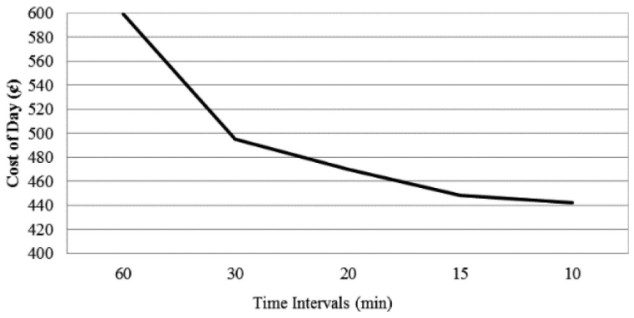

Fig. 2.12: Time Interval - Day Cost from [1]

(HERS)[18]. Shuvo et al. proposed a fully fledged home REM that added a very interesting interaction with the users. The model was given the freedom to initiate jobs on its own, as in the [22] paper, however, the user could manually override the actions that the system took. In the paper, they divided household devices into 3 categories: priority, deferrable, and flexible devices. Priority devices can be switched on and off at any time, deferrable devices need to operate for a certain time to complete their function, and flexible devices can change their energy consumption. Additionally, they decided that they would have multiple RL agents at home, where each of them would be responsible for a specific device. They also highlighted the importance that the State (2.1.1) selection plays in the success of an RL project. More specifically, 2.1 shows the State that they chose for each device. The Reward formula focused on minimizing the total energy cost for all devices, but also took into consideration user dissatisfaction when the agents initiated jobs that were not desired. The authors tried to convert user dissatisfaction into US dollars through a 20-person survey where each person had to set a discomfort cost for the 4 house devices. The proposed approach of the article was tested on the ARAS data set and used the A2C algorithm (2.1.2) to train agents. They concluded that they show better results than the other state-of-the-art approaches based on 2.13. They also noted, based on 2.14 that their agents understood how to efficiently manage appliances to reduce the household energy bill cost. One questionable decision was the survey they conducted that translated user discomfort into USD, which many could argue should be left up to the model to decide the importance of each device for the user, based on the feedback it receives. Our work proposes an entirely different problem formulation. The most noticeable difference is that the Reinforcement learning agent supervises all of the devices collectively, while Shuvo et al. assign a different agent for each device. Additionally, we include only priority and deferrable devices in the households that we test, but we add more details about deferrable devices in the State. Moreover, our Reward formula is different, because we penalize the agent with the goal of learning to use devices correctly and adapt to the user's feedback. Furthermore, we experiment with more household types, more state-of-the-art Reinforcement learning algorithms, and train our agents with many different

hyperparameter combinations, like different time intervals for decisions and episodes with different lengths (1-day,7-days,30-days). Finally, our project is open source and fully customizable.

Input	AC	DW	WD	EV
Activity	✓	✓	✓	✓
Clock Time	✓	✓	✓	✓
Electricity price	✓	✓	✓	✓
Device status	✓	✓	✓	✓
Device activation duration	✓	✓	✓	✓
Battery charge level	X	X	X	✓
Travel upcoming	X	X	X	✓

Tab. 2.1: State for each Device from [18]

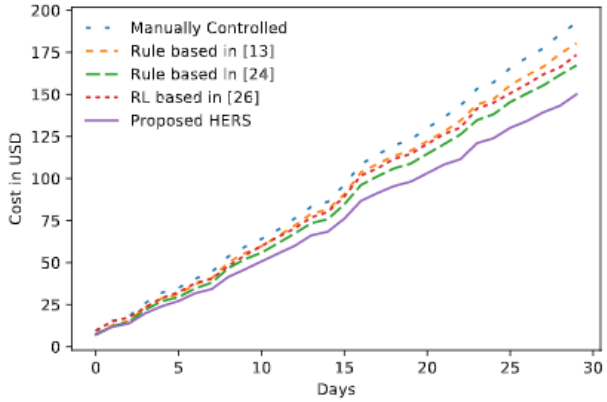

Fig. 2.13: Daily cumulative cost comparison for all devices among the considered policies [18]

All in all, in this thesis we train a single Reinforcement learning agent for multiple devices to capture complex user behavior patterns, while most existing work trains a single agent per device. More specifically, we introduce a suggestive Residential Energy Manager that supervises the entire household, with a State, which is more complex than other States used in existing work, that includes the energy price, the status of all household devices, including priority and deferrable devices, the energy consumption of the devices, and some specific attributes of each device, such as their functionality time. In addition, we propose a novel Reward formula that takes into consideration multiple factors, unlike most related work where it focuses only on shifting the power consumption, such as: the correct use of each device (e.g. not breaking the function of a device unless the resident asks it), reducing the household costs by correctly shifting power consumption, and adapting to the resident's unique habits. Furthermore, we perform more thorough experimentation than other existing work, testing our proposal on three state-of-the-art Reinforcement learning algorithms (A2C, PPO, MARWIL) with multiple hyperparameters and household scenarios. Finally, we are the first to offer a fully

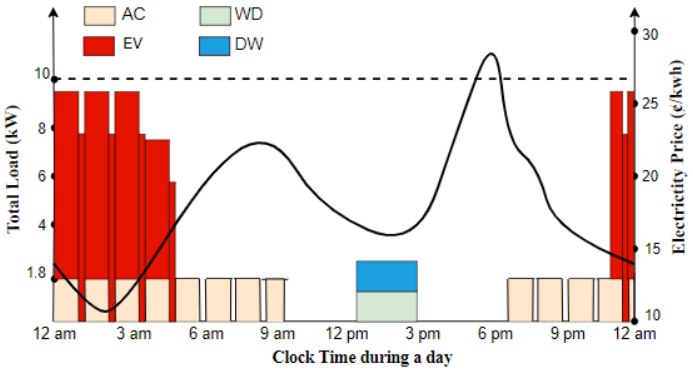

Fig. 2.14: HERS scheduling results for a day [18]

customizable household gym simulation environment to the research community. We hope that this initiative will help increase research in the field.

3 System Model

3.1 Model

In this section, we present our work. In the Reinforcement learning (RL) background (2.1.1) we mention that the State, the Action, and the Reward are integral parts of any RL project. In the following sections, we provide an extensive analysis of the proposed approach.

3.1.1 State

Our State contains all the basic components of a household and some specifically useful attributes of deferrable devices, such as their time of function. This generic nature of it allows others to build scenarios on top of it, since most buildings can be expressed in this State form. More specifically, a household has D devices where X are priority devices and Y are deferrable devices. Each time slot is defined as t with $t \in \{0, \dots, T\}$ where T is the time horizon.

The state (2.1.1) S_t in the time slot t includes:

- Time (Hour) at time slot t
- The status of priority device $x \in \{1, \dots, X\}$ for time slot t : F_t^x, where F_t^x is the status of device x for time slot t (1 for on, and 0 for off).
- The status of deferrable device $y \in \{1, \dots, Y\}$ for time slot t : F_t^y
- The energy consumption of priority device $X \in \{1, \dots, X\}$ measured in KWh : C_x
- The energy consumption of deferrable device $Y \in \{1, \dots, Y\}$ measured in KWh : C_y
- The energy tariff (price) from the retailer for the time slot t: p_t

- The on duration of deferrable device $y \in \{1, \ldots, Y\}$ in hours : J_y
- The remaining hours for deferrable device $y \in \{1, \ldots, Y\}$ to finish its function : U_y

3.1.2 Action

Our RL agent suggests an action for all electrical devices in the household. More specifically, at time slot t suggests action (P_t, D_t) to the resident defined as $P_t = \{P_t^1, \ldots, P_t^x, \ldots, P_t^X\}$ and $D_t = \{D_t^1, \ldots, D_t^y, \ldots, D_t^Y\}$, where P_t^x is the recommended action regarding priority devices $x \in \{1, \ldots, X\}$ and D_t^y is the recommended action regarding deferrable devices $y \in \{1, \ldots, Y\}$ for time slot t. Recommended actions P_t^x and D_t^y can be **ON** or **OFF** (1 or 0).

3.1.3 Observation

The observation contains a subset of information available in the state. In other words, the Reinforcement learning (RL) agent does not have absolute information about the environment. To be more precise, when deciding its action the agent has knowledge of:

- Time (Hour) at time slot t
- The status of priority device $x \in \{1, \ldots, X\}$ for time slot t : F_t^x
- The status of deferrable device $y \in \{1, \ldots, Y\}$ for time slot t : F_t^y
- The remaining hours for deferrable device $y \in \{1, \ldots, Y\}$ to finish its function : U_y
- The energy tariff (price) from the retailer for the time slot t: p_t

3.1.4 User Modeling

One of the main goals of the Reinforcement learning (RL) agent is to adapt to the resident of the household. Therefore, it is really important to model the resident as correctly and as accurately as possible inside the simulation environment. Other works in the literature do not define the user and only task the agent to shift energy consumption when the price of electricity is high. In our work, we assign the probabilities p1, p2, q1 and q2 for each electrical device in the household as shown

in Figure 3.1. To be more precise, p1 is the probability of the resident to accept the suggestion of the RL agent when the device is on and the agent suggests keeping it on. Furthermore, p2 is the probability that the resident rejects the suggestion of the RL agent when the device is on and the agent suggests that the device be turned off. Similarly, q1 and q2 are the probabilities that the resident accepts keeping the device closed when the device state is off and rejecting a turn on request when the device is off, respectively.

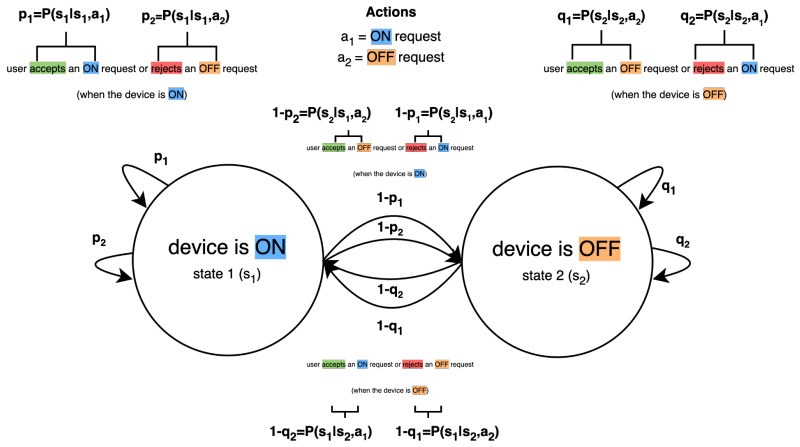

Fig. 3.1: User Transition Probabilities

3.1.5 Reward

In this section, we will analyze the Reward Formula (2.1.1) which we used to train our RL agent. This formula has as a goal to take into account the reduction of electricity costs, the correct use of deferrable devices without interrupting their function and the adaptation to the habits of the residents, which are modeled on the basis of 3.1.4.

Priority Devices Reward:

The priority energy cost for time slot t is defined as:

$$PC_t = p_t \sum_{x=1}^{X} C_x \tag{3.1}$$

The priority reward for time slot t can be defined as:

$$R_t = -\sum_{x=1}^{X}(1 - f_{x_t}) \cdot PC_t + f_{x_t} \cdot \delta_x \tag{3.2}$$

where f_{x_t} is the user feedback and $\delta_x \in R^+$ is the cost coefficient that represents the user's discomfort for device x (δ_x is an input parameter).

$$f_{x_t} = \begin{cases} 1 & \text{if } F_t^x \neq P_t^x \\ 0 & \text{if } F_t^x = P_t^x \end{cases}$$

Deferrable Devices Reward:

Deferrable devices should remain on for a specific set of time slots when they are activated (e.g. washing machine has programs that last specific minutes). The duration of usage (in hours) of a deferrable device y is defined as $J_y \in R^+$. Additionally, the usage countdown U_y represents the time left for a deferrable device to finish its function after it is activated.

The deferrable energy cost for time slot t is defined as:

$$DC_t = p_t \sum_{y=1}^{Y} C_y \tag{3.3}$$

The deferrable device reward R_t^* is defined as:

$$R_t = -\sum_{y=1}^{Y}(1 - \phi_{y_t}) \cdot ((1 - f_{y_t}) \cdot DC_t + f_{y_t} \cdot \delta_y) + \phi_{y_t} \cdot \gamma_y \tag{3.4}$$

where ϕ_{y_t} checks for the correct use of deferrable devices and $\gamma_y \in R^+$ is the penalty applied for falsely stopping the function of a deferrable device (γ_y is an input parameter).

$$f_{y_t} = \begin{cases} 1 & \text{if } F_t^y \neq D_t^y \\ 0 & \text{if } F_t^y = D_t^y \end{cases}$$

,

$$\phi_{y_t} = \begin{cases} 1 & \text{if } D_y^{t-1} = 1 \wedge D_y^t = 0 \wedge U_y > 0 \\ 0 & \text{otherwise} \end{cases}$$

Final Reward:

The total reward for the time slot t when both priority and deferrable devices are integrated is:

$$R_t^{\text{total}} = R_t + R_t^* \tag{3.5}$$

3.2 Experiments

3.2.1 Experiment Setup

We use Gymnasium (2.1.1) to create the proposed household simulation environment. This environment contains the Observation space (3.1.3), the Action space (3.1.2), the Reward formula (3.1.5) and the Init, Step and Reset functions. To begin with, the Init function is what initializes the environment. Here, the user can set hyperparameters for the household. The Reset function resets all variables and rewinds the environment to its initial state. Finally, the Step function is where the transitions occur. It takes as input the action that the Reinforcement learning (RL) agent has decided to suggest and is responsible for changing the Observation state accordingly and calculating the reward based on the Reward formula, which is then going to be fed back to the agent so it can learn. It also checks if an Episode (2.1.1) has ended, and in that case resets the simulation using the Reset function.

We also use Ray-RLlib (2.1.1) to structure and run our experiments. Ray provides us with countless combinations of parameters with which we can experiment if we take into account both the proposed Gymnasium environment and the available RL algorithms and their hyperparameters that RLlib offers us. Therefore, we decided to keep certain parameters constant. To begin, in all our experiments we used the Long Island 2018-2019 electricity price data set provided by the New York Independent System Operator [1]. Additionally, we did not experiment with the hyperparameters of the A2C (2.1.2), MARWIL (2.1.2) and PPO (2.1.2) algorithms and only used the standard configurations shown in Table 3.1. Moreover, we used a probabilistic model of a user, based on 3.1.4, which is generally accepting of suggestions because

[1] NewYorkISO : https://www.nyiso.com/real-time-dashboard

we did not want a user's unique habits to heavily interfere with the RL agent's appliance management decision-making. To be more precise, for all devices in the household, we assign probabilities p1 and q1 the value of 0.9 and p2 and q2 the value of 0.1. Furthermore, we wanted to test the RL algorithms in environments with different complexities. Therefore, we decided to test four different versions of the environment, where the complexity gradually increased. First, we created an environment with only one priority device, which was considered the simplest. Next, we added more devices of the same type to test how efficiently algorithms managed many devices at the same time and created an environment with five different priority devices. The next step was to add a deferrable device to the environment, which examined the agent's ability to control different devices. In other words, this environment contained one priority and one deferrable device. Finally, we increased the number of devices for both types of devices, leading to the most complex environment with five different priority and five different deferrable devices. The energy cost of the device (in KW / h) and the function time (in hours) were based on real household data[2]. Table 3.2 contains details for the first environment, Table 3.3 for the second, Table 3.4 for the third, and Table 3.5 for the fourth. Lastly, we experimented with the frequency with which the RL agent could suggest actions to the user and the lengths of the episodes in which the agents would be trained. More specifically, we decided to test the algorithm's ability to suggest actions every 30-minutes and every 1-hour. In addition, we experimented with 1-day, 7-day, and 30-day episode lengths to understand the differences between the decision-making of algorithms with different training times. The final structure of the experiments can be seen in 3.2. The machine that ran the experiments had the following hardware specs:

- **CPU Intel Core i7 12700 1700/2.1 GHz/**
- **64 GB Ram Memory**
- **GeForceRTX3080 Graphics Card**

Tab. 3.1: RL algorithms configurations

Parameter	Value
learning Rate	0.001
Gamma	0.99
Train Batch Size	300
Gradient Clip	None
Training Episodes	150
Model Structure	Input Layer, 2 Hidden Layers with 64 neurons and an Output layer

[2]Household Device data collected from Daft Logic, url=https://www.daftlogic.com/information-appliance-power-consumption.htm

Tab. 3.2: 1 Priority Device Environment Details

Devices	Consumption (in KW / h)
Light bulb (Incandescent)	0.1

Tab. 3.3: 5 Priority Device Environment Details

Devices	Consumption (in KW / h)
Light bulb (Incandescent)	0.1
Light bulb (Incandescent)	0.06
Air Condition	2.5
Ceiling Fan	0.07
Hot Water Dispenser	1.3

Tab. 3.4: 1 Priority and 1 Deferrable Devices Environment Details

Devices	Consumption (in KW / h)	Function (in hours)
Light bulb (Incandescent)	0.1	None
Dishwasher	1.3	2.5

Tab. 3.5: 5 Priority and 5 Deferrable Devices Environment Details

Devices	Consumption (in KW / h)	Function (in hours)
Light bulb (Incandescent)	0.1	None
Light bulb (Incandescent)	0.06	None
Air Condition	2.5	None
Ceiling Fan	0.07	None
Hot Water Dispenser	1.3	None
Dishwasher	1.3	2.5
Dishwasher	1.0	1.0
Clothes Dryer	2.4	0.5
Food Dehydrator	0.8	4.0
Oven	2.15	2.0

Time-Step Duration:

- Every 30 minutes (0.5 hours):
  - **Days of training and testing**:
    - 1 day:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices
    - 7 days:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices
    - 30 days:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices
- Every 1 hour (1 hour):
  - **Days of training and testing**:
    - 1 day:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices
    - 7 days:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices
    - 30 days:
      - 1 priority device
      - 5 priority devices
      - 1 priority and 1 deferrable devices
      - 5 priority and 5 deferrable devices

Fig. 3.2: Experiment Structure

3.2.2 Experiment Results

One Priority Device Environment:

In this section, we present the results for the simplest possible simulation environment. This household contains only one device, a light bulb (3.2) with consumption of 0.1 KW. Figures 3.3, 3.4 and 3.5 present the results of the experiments. The main takeaways from these experiments are the following:

- The PPO Reinforcement learning agent in Figure 3.3 showed that with 30-minute decisions it remained stable and could not increase its performance, and with 1-hour decisions after the 30th episode it steadily got worse. In Figure 3.4 we noticed a gradual improvement with both time intervals showing an adaptation to the household environment. In Figure 3.5 we observe an increase in performance with 30-minute decisions and a steady decrease in performance after quickly reaching its peak at 30 episodes with 1-hour decisions. In general, based on the reward scores, we understand that the PPO agent is able to adapt to the consumer and the simplest environment.

- The A2C Reinforcement learning agent based on Figure 3.3 improved rapidly and the 30-minute decision model showed better performance. In Figures 3.4 and 3.5 we observe that the agent quickly reached its maximum performance and remained relatively stable afterward. Overall, the A2C model performed well with frequent decisions. It was able to learn faster the more it interacted with the user, which was expected because the Actor and the Critic had more time to adapt to the resident. Additionally, we noticed that increasing the length of training episodes did not help the agent to better adapt to the environment. In general, the agent was able to adapt to the simple environment and the user.

- The MARWIL Reinforcement learning agent showed the most promise in terms of behavior. Although its performance in Figure 3.3 was not the best, which was derived from its slow start, it consistently improved its behavior and slowly adapted to its environment. In Figure 3.4 we notice unstable behavior despite the high reward that it accumulated in 1-hour decisions at the beginning. In Figure 3.5 we again observed a promising behavior that showed a steady increase. In general, we believe that in more complex households, algorithms with this behavior tend to perform better. We confirm this assumption in the following sections.

- Table 3.6 shows that, in general, all Reinforcement learning methods performed relatively the same in this environment, which is expected since it was the simplest. Furthermore, we observed that both the A2C and the MARWIL agent perform really well with 30-minute decisions, while the PPO agent performed well with 1-hour decisions. Additionally, the MARWIL agent improved the longer the training episode was.

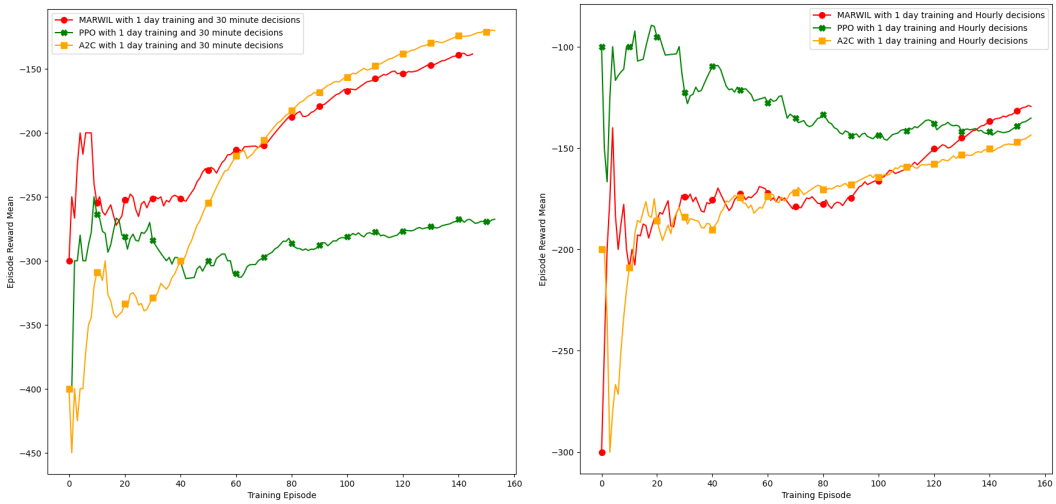

Fig. 3.3: Training Figures for the Environment with One Priority Device and 1 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

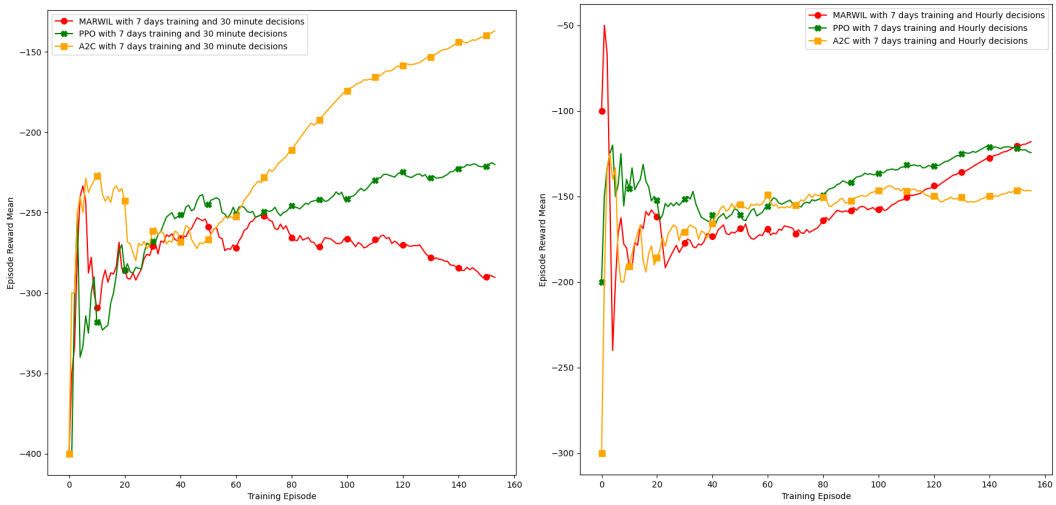

Fig. 3.4: Training Figures for the Environment with One Priority Device and 7 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

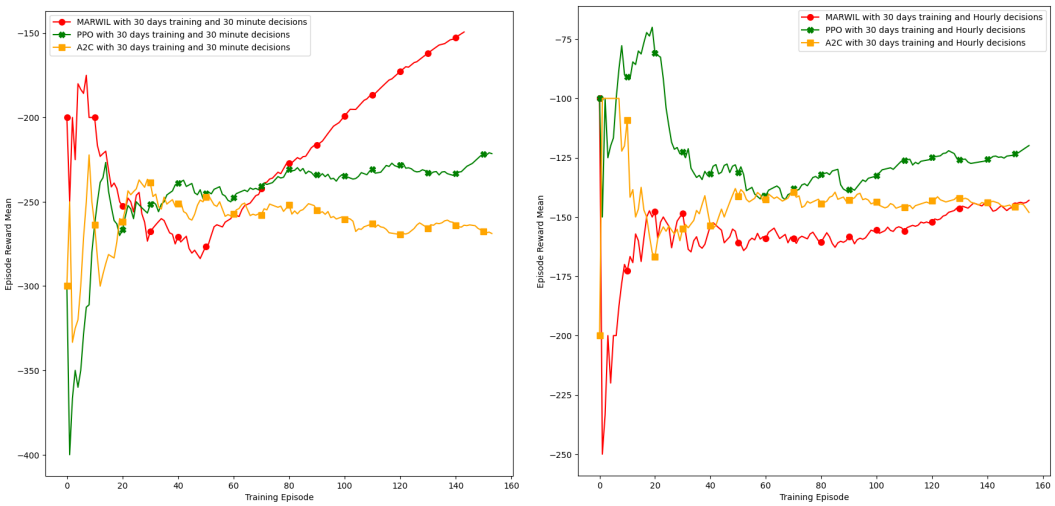

Fig. 3.5: Training Figures for the Environment with One Priority Device and 30 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

Tab. 3.6: One Priority Device Environment : A2C vs PPO vs MARWIL - Which algorithm yielded the best mean episode reward?

Parameters	Best Algorithm
30 minute timestep - 1 day training	A2C
1 hour timestep - 1 day training	PPO
30 minute timestep - 7 days training	A2C
1 hour timestep - 7 days training	MARWIL
30 minute timestep - 30 days training	MARWIL
1 hour timestep - 30 days training	PPO

Five Priority Devices Environment:

In this section, we experiment in a household with five different priority devices. The residence has two light bulbs, one with 0.1 and one with 0.06 KW consumption. It also has an Air Condition, with 2.5 KW consumption, a Ceiling Fan with 0.07 KW consumption, and a How Water Dispenser with 1.3 KW consumption (3.3). Figures 3.6, 3.7 and 3.8 present the results of the experiments. The main takeaways from these experiments are the following:

- The PPO agent showed that it struggled to handle more priority devices. In Figure 3.6 it showed little progress. More specifically, with 1-hour decisions after the first 10 episodes, it witnessed a big drop in performance that it could not recover from. In Figures 3.7 and 3.8 we observe that, apart from a small increase in the first 15 episodes, the reward in the environment remained the same.

- The A2C algorithm presented encouraging behavior. Although in Figure 3.6 we observe that it is not performing that well, in scenarios where it had more time to interact with the environment, it showed its capabilities. More specifically, we notice that in both Figures 3.7 and 3.8, the A2C agent rapidly increased its reward. Furthermore, it performed better with more frequent decisions and was clearly able to adapt to the resident's needs, which can be observed by the smooth learning curves.

- The results of the MARWIL agent confirmed the assumptions made in the previous section. Figures 3.6, 3.7, and 3.8 demonstrate its ability to handle many devices at the same time. The only case where we noticed that it struggled was with 1-hour decisions in 1-day training episodes, where its performance dropped after the 15th episode, indicating its need to interact more with the environment in order to improve. In general, within the more complex household, it managed to showcase a great learning curve in most combinations, even outperforming the A2C agent in some scenarios.

- In Table 3.7 we observe that the Reinforcement learning algorithms that utilize both value-based and policy-based methods were able to adapt better in the second, more complex household and claim better results in all six categories. On the contrary, the policy-based algorithm did not manage to keep up and did not beat the others in any scenario.

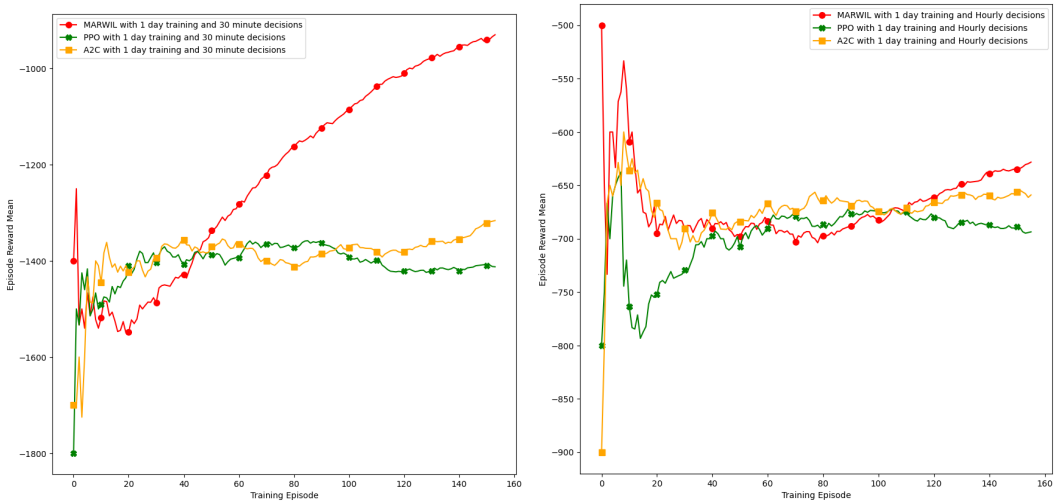

Fig. 3.6: Training Figures for the Environment with Five Priority Devices and 1 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

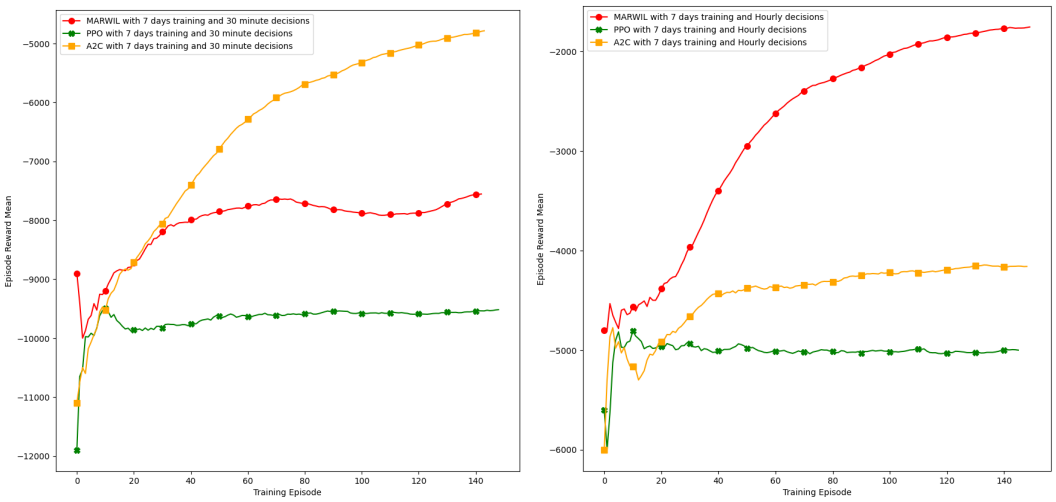

Fig. 3.7: Training Figures for the Environment with Five Priority Devices and 7 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

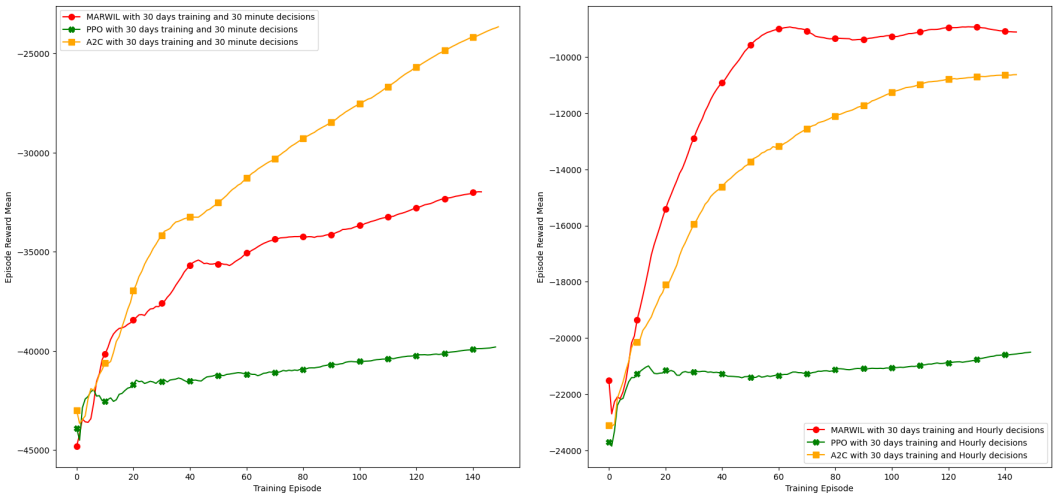

Fig. 3.8: Training Figures for the Environment with Five Priority Devices and 30 day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

Tab. 3.7: Five Priority Devices Environment : A2C vs PPO vs MARWIL - Which algorithm yielded the best mean episode reward?

Parameters	Best Algorithm
30 minute timestep - 1 day training	MARWIL
1 hour timestep - 1 day training	MARWIL
30 minute timestep - 7 days training	A2C
1 hour timestep - 7 days training	MARWIL
30 minute timestep - 30 days training	A2C
1 hour timestep - 30 days training	MARWIL

One Priority and One Deferrable Device Environment:

In this section, we test RL agents in a simulation environment with one priority and one deferrable device. To be more precise, the priority device is a light bulb with a consumption of 0.1 KW, and the deferrable device is a dishwasher with consumption of 1.3 KW and a function time of 2.5 hours (3.4). Experiments in this household produced many interesting insights. Figures 3.9, 3.11 and 3.13 show the general results of the Reinforcement learning algorithms. The core findings of these experiments are the following:

- The PPO agent struggled with the addition of a deferrable device. More specifically, in Figure 3.9 we observe that with 30-minute decisions there is a small improvement in the first 10 episodes, followed by a stable condition, and with 1-hour decisions, while the reward starts high, it immediately drops and remains stable after the 20th episode. These results can be explained by looking at Figure 3.10. We observe that the PPO agent in both time intervals receives a high reward for priority devices and a low reward for deferrable devices, which explains the trends we discussed previously. Furthermore, in Figures 3.11 and 3.13 the PPO agent does not show great improvement, especially after the first episodes. Looking at Figures 3.12 and 3.14, we realize that the reason for such low rewards is that the PPO agent cannot control well deferrable devices. Therefore, it is obvious that the PPO algorithm has problems managing both types of devices simultaneously.

- The A2C agent in Figure 3.9 showed promising results in 1-day training episodes, improving rapidly with both time intervals after the 40th episode. Furthermore, in Figure 3.10 we see that it achieved a high reward in both priority and deferrable devices. In Figure 3.11 we notice that with decisions of 30 minutes, its reward decreased dramatically after episode 60, and with 1-hour decisions, it slowly increased. If we also observe Figure 3.12, we understand that the reason for this behavior is the deferrable reward it received. In other words, in both scenarios, it did not manage deferrable devices well, leading to these mediocre results. Finally, in Figure 3.13 we observe that it performed really well with 30-minute decisions, but really poorly with 1-hour decisions. According to Figure 3.14, this is the result of mismanagement of the deferrable device. The results indicated that the A2C agent mainly understands how to use both devices simultaneously, but it can be unstable.

- The results of the MARWIL agent displayed its ability to gradually improve in complex environments. In Figures 3.9, 3.11 and 3.13 we observe that after the 20th episode the reward it receives greatly increases. Looking at Figures 3.10,

3.12 and 3.14, we immediately notice its ability to achieve high rewards with both types of devices.

- Table 3.8 makes clear that the MARWIL agent is the most potent in controlling both priority and deferrable devices at the same time. The only category that it loses is in 1-day training episodes and 30-minute decisions. However, this makes sense, since so far in our experiments we have observed that with shorter training episode lengths, the A2C algorithm often performs better than the MARWIL algorithm.

Tab. 3.8: One Priority and One Deferrable Devices Environment : A2C vs PPO vs MARWIL - Which algorithm yielded the best mean episode reward?

Parameters	Best Algorithm
30 minute timestep - 1 day training	**MARWIL**
1 hour timestep - 1 day training	**PPO**
30 minute timestep - 7 days training	**MARWIL**
1 hour timestep - 7 days training	**MARWIL**
30 minute timestep - 30 days training	**MARWIL**
1 hour timestep - 30 days training	**MARWIL**

Five Priority and Five Deferrable Device Environment:

In this section, we perform experiments in the most realistic and complex simulation environment. This household contains five priority devices and five deferrable devices. More specifically, the priority devices are two light bulbs, one with 0.1 and one with 0.06 KW consumption, an Air Condition with 2.5 KW consumption, a Ceiling Fan with 0.07 KW consumption, and a Hot Water Dispenser with 1.3 KW consumption. The deferrable devices are two dishwashers, one with 1.3 KW consumption and 2.5 hour function time and one with 1.0 KW consumption and one hour function time, a clothes dryer with 2.4 KW consumption and half an hour function time, a food dehydrator with 0.8 KW consumption and a 4 hour function time, and a 2.15 KW consumption oven with a 2 hour function time (3.5). Experiments conducted in this household yielded fascinating observations. Figures 3.15, 3.17 and 3.19 show the general results of the agents. The core takeaways of these experiments are the following:

- The PPO agent in 1-day training episodes (Figure 3.15) provided poor results with both 30-minute decisions and 1-hour decisions. More specifically, in Figure 3.16, we observe that with 1-hour decisions it struggled to manage both priority and deferrable devices, while with 30-minute decisions it failed to improve the reward for either after the 20th episode. In Figure 3.17 we see the exact opposite. Its performance was good with 30-minute decisions and poor with 1-hour decisions. Looking at Figure 3.18 we observe that the

model with 1-hour decisions was unable to increase its reward in both device types, while the model with 30-minute decisions had decent rewards until the 40th episode. Lastly, in Figure 3.19, we note that the PPO agent cannot adapt to the complex environment and that almost all device rewards are underwhelming (Figure 3.20). In general, in most situations where it managed to have good performance with the deferrable devices, its performance with the priority devices dropped, meaning that it could not handle the complexity of the problem and control many devices at the same time efficiently, along with adapting to the resident.

- The A2C agent performed well with shorter training episode lengths and had poor performance with longer training episode lengths. In Figure 3.15 we observe the constant increase in total reward after the 20th episode. More specifically, according to Figure 3.16, this is due to the positive increase in both rewards, which means that the A2C agent with 1-day training episodes learned to have good control of both types of devices. In Figure 3.17 we notice that only the 30-minute decision model performed. According to 3.18 this effect is related to the ability of the model to control the deferrable devices well. In contrast, the model that suggested actions every hour had good results with priority devices and struggled with deferrable devices, leading to poor rewards. Overall, both the 7-day training episode models could not control both types of devices. Finally, in Figure 3.19 both models have underwhelming performance. Figure 3.20 shows that the models have difficulty in balancing devices, resulting in mediocre rewards. In general, the performance of the A2C algorithm was great in 1-day training episodes and poor in the other scenarios.

- The MARWIL agent, as we have already established, works really well in extremely complex problems with hybrid action spaces (2.1.2). In this household scenario, where the number of devices has increased and the complexity is high, the strengths of this Reinforcement learning algorithm helped it achieve high performance. More specifically, in Figure 3.15 we notice that the algorithm performs extremely well with 30-minute decisions, but struggles with 1-hour decisions before the 30-episode mark. After the 30th episode, its performance with 1-hour decisions continually increases and achieves great rewards. To explain this behavior, we have to look at Figure 3.16, where we see that with decisions of 30 minutes, both the priority and the deferrable reward constantly increase, while with decisions of 1 hour, the agent struggles to use both devices before episode 30. In Figure 3.17 the MARWIL agent has a smooth learning curve and accumulates great rewards with 1-hour decisions, but witnesses a sudden decrease after the 35th episode with 30-minute decisions. Looking at Figure 3.18 we observe that with 30-minute decisions, after peaking at the 30th episode, the deferrable device reward dramatically decreases, and with

1-hour decisions both rewards constantly increase, reaching their peak around the 100th episode. Finally, in Figure 3.19 we observe amazing performance with 1-hour decisions. Additionally, the model that takes decisions every 30-minutes reaches its peak approximately at episode 60. Figure 3.20 shows that with 1-hour decisions the MARWIL agent seems to perform extremely well, while with 30-minute decisions its decision-making ability is limited and the deferrable device reward peaks at the 65th episode. Overall, the MARWIL agent performed great in all scenarios. More specifically, with 1-day training episodes, the 30-minute decision model showed better performance, while with long training episodes the 1-hour decision-making interval had greater results.

- Table 3.9 shows the dominance of the MARWIL agent in the most complex scenario. This behavior is expected since, as mentioned in 2.1.2, this RL algorithm is exceptional for handling complex environments with a hybrid action space, such as this household. At the same time, we have to outline the ability of the A2C algorithm to perform consistently great in 1-day training episode scenarios.

Tab. 3.9: Five Priority and Five Deferrable Devices Environment : A2C vs PPO vs MARWIL - Which algorithm yielded the best mean episode reward?

Parameters	Best Algorithm
30 minute timestep - 1 day training	**MARWIL**
1 hour timestep - 1 day training	**A2C**
30 minute timestep - 7 days training	**MARWIL**
1 hour timestep - 7 days training	**MARWIL**
30 minute timestep - 30 days training	**MARWIL**
1 hour timestep - 30 days training	**MARWIL**

3.2.3 Testing Experiments for the Best Algorithm

For this section, we decided to test the best RL algorithm on a specific day that was not included in the training data set. More specifically, we performed experiments in the most realistic environment (3.5) with the MARWIL algorithm. To be more precise, we tested the 1-day, 7-day, and 30-day training algorithms on how they handle both priority and deferrable devices throughout the day. Figure 3.23 shows the 1-day training MARWIL agent successfully shifting the load away from electricity peaks. In addition, we observe low usage of deferrable devices throughout the day and larger spikes for priority devices. By observing Figure 3.22 we immediately notice an increase in device usage throughout the day. It is possible that the MARWIL agent trained in 7-day scenarios approaches the appliance management problem from a more macroscopic perspective. In other words, the agent may ignore some temporarily high prices and use the appliances because it believes that there will

be higher peaks soon. Similarly, in Figure 3.23 the same patterns repeat. The usage remains high, ignoring the high electricity peaks, because the agent makes the assumption that there will be higher electricity prices on the other days of the month.

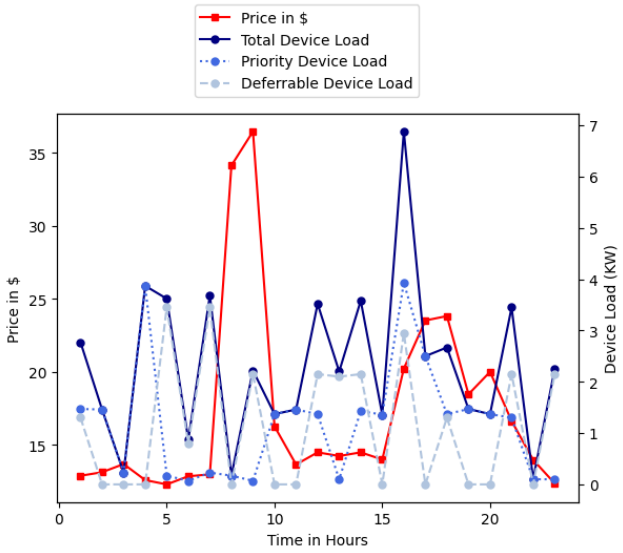

Fig. 3.21: Five Priority and Deferrable Device Experiment Test : MARWIL results with 1-day training episodes - Device Load is the sum of Priority and Deferrable Device Load

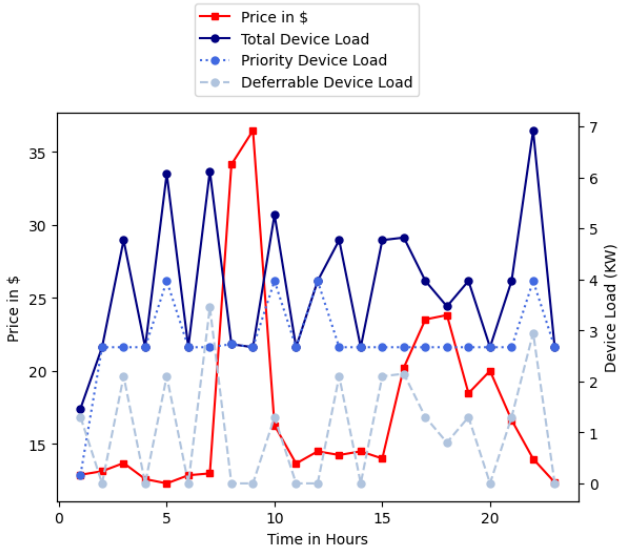

Fig. 3.22: Five Priority and Deferrable Device Experiment Test : MARWIL results with 7-days training episodes - Device Load is the sum of Priority and Deferrable Device Load

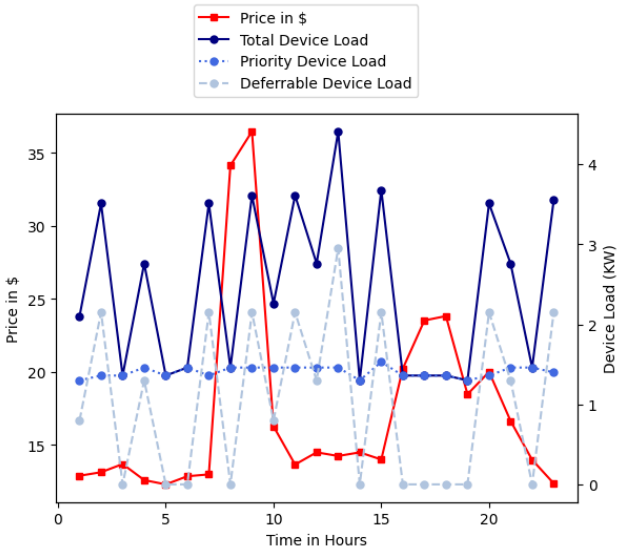

Fig. 3.23: Five Priority and Deferrable Device Experiment Test : MARWIL results with 30-days training episodes - Device Load is the sum of Priority and Deferrable Device Load

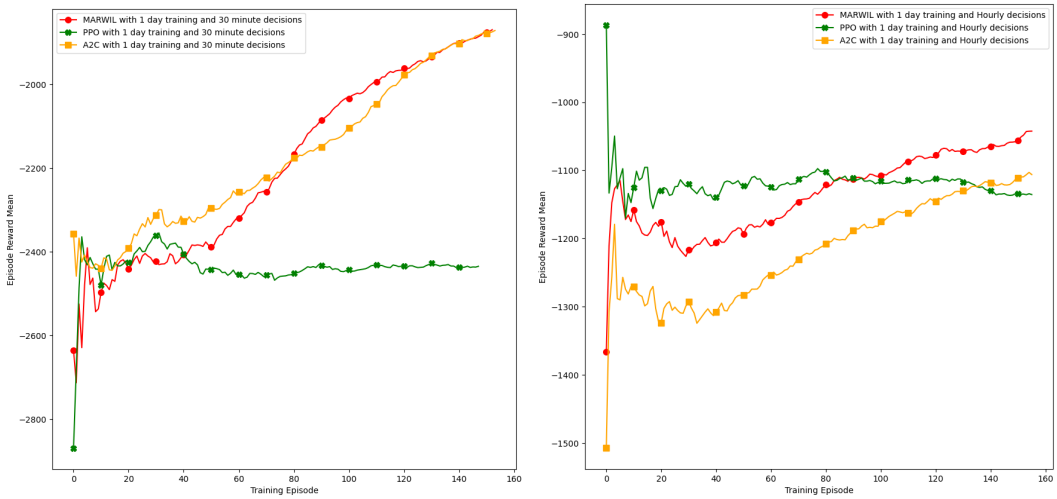

Fig. 3.9: Training Figures for the Environment with One Priority and One Deferrable Devices and 1-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

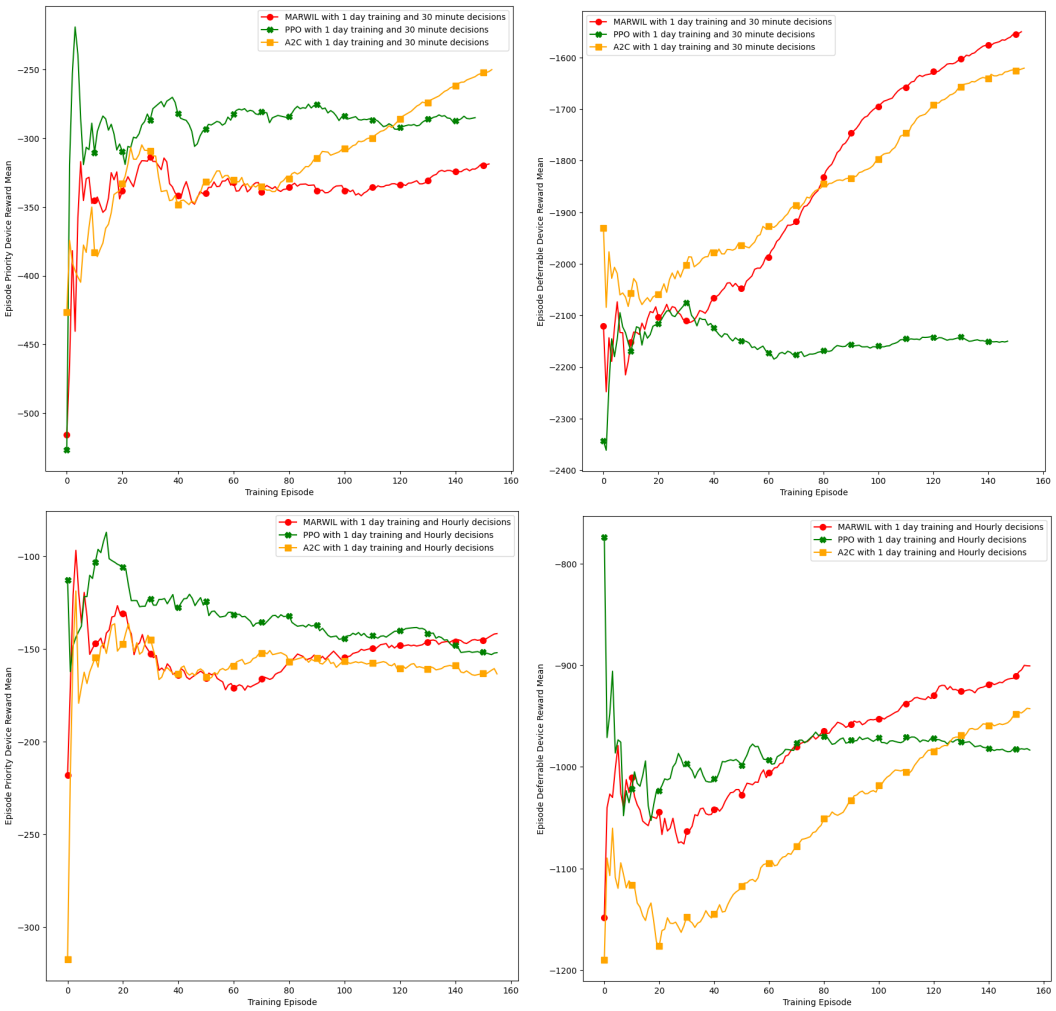

Fig. 3.10: Training Figures for the Environment with One Priority and One Deferrable Devices and 1-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

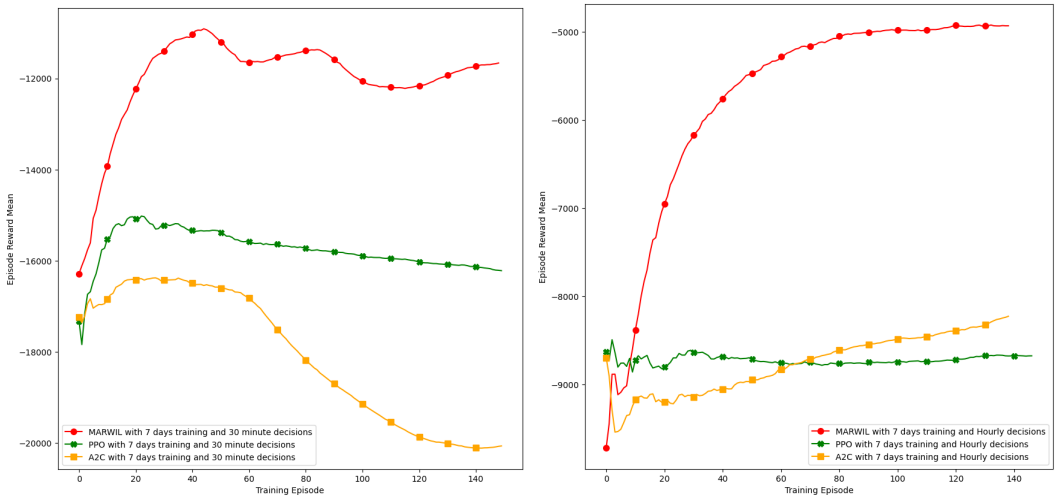

Fig. 3.11: Training Figures for the Environment with One Priority and One Deferrable Devices and 7-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

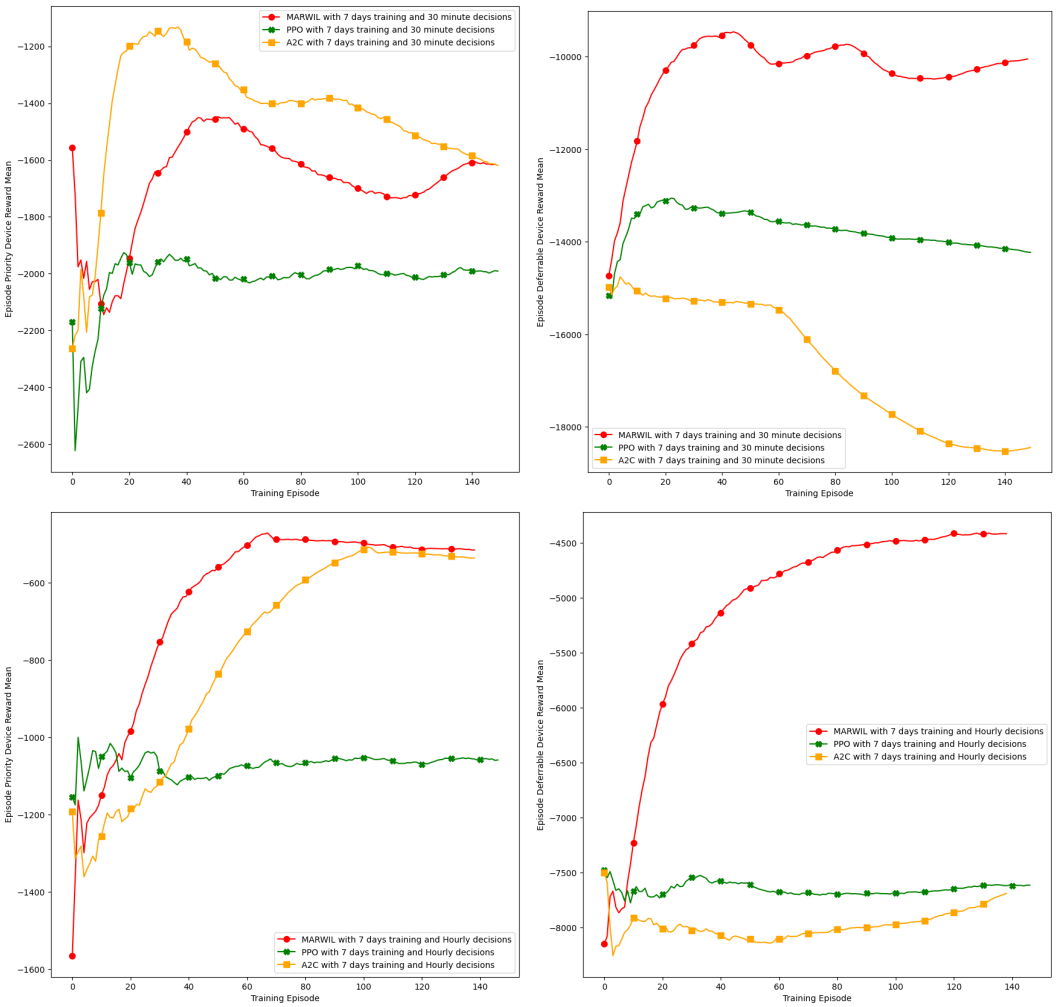

Fig. 3.12: Training Figures for the Environment with One Priority and One Deferrable Devices and 7-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

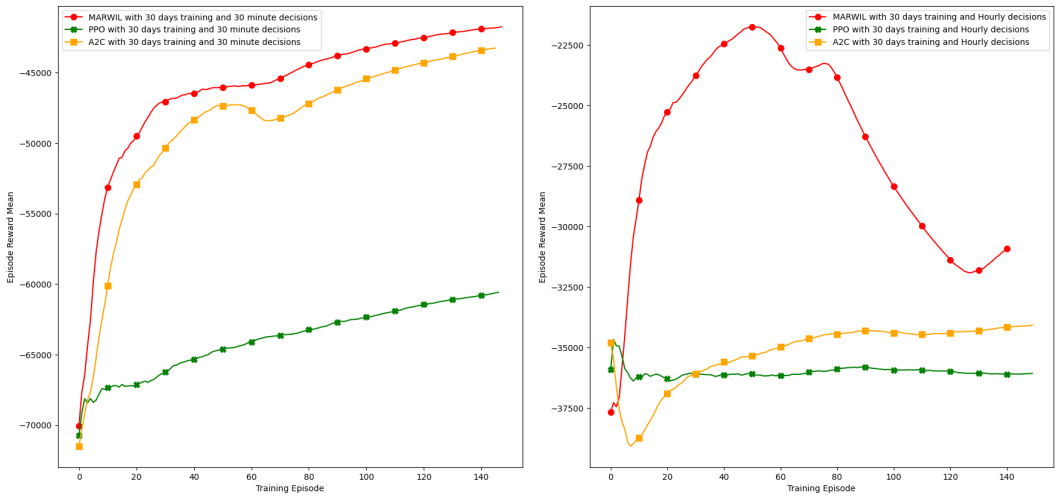

Fig. 3.13: Training Figures for the Environment with One Priority and One Deferrable Devices and 30-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

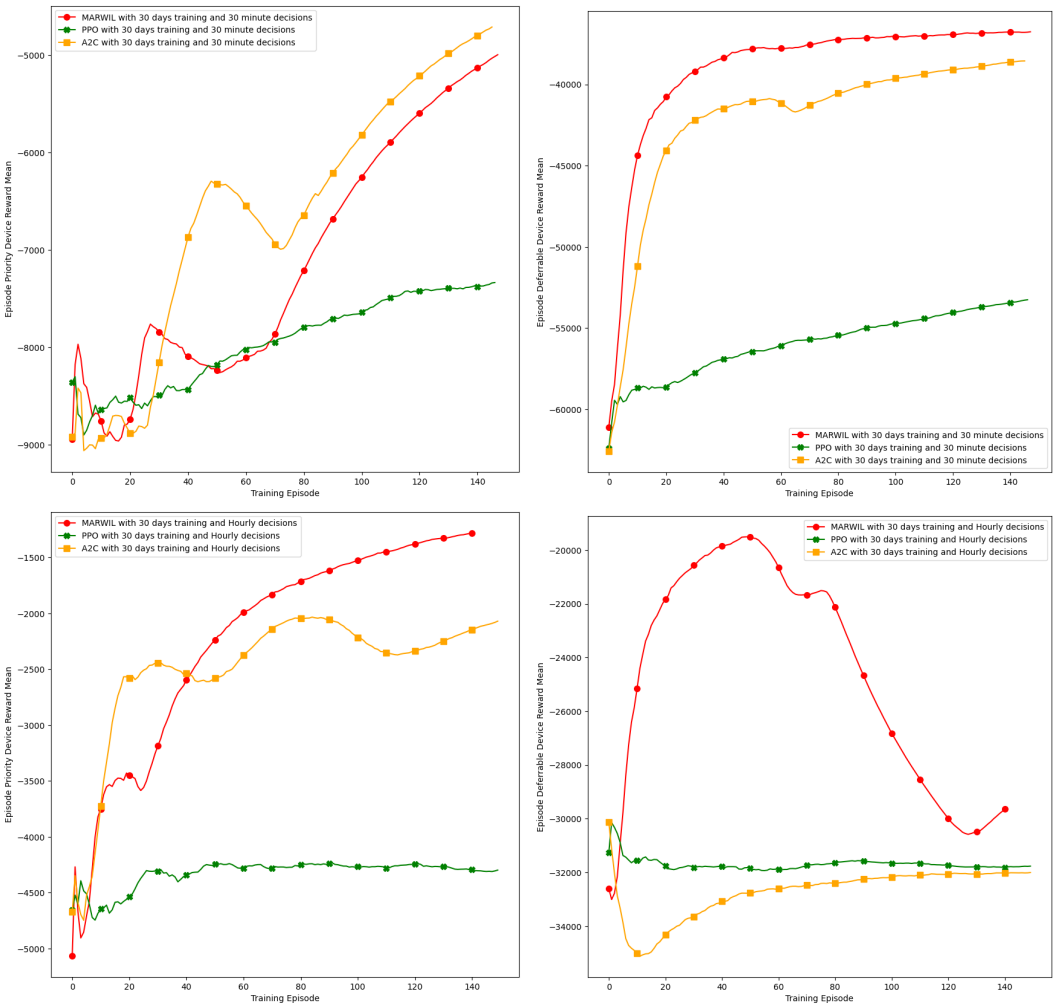

Fig. 3.14: Training Figures for the Environment with One Priority and One Deferrable Devices and 30-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

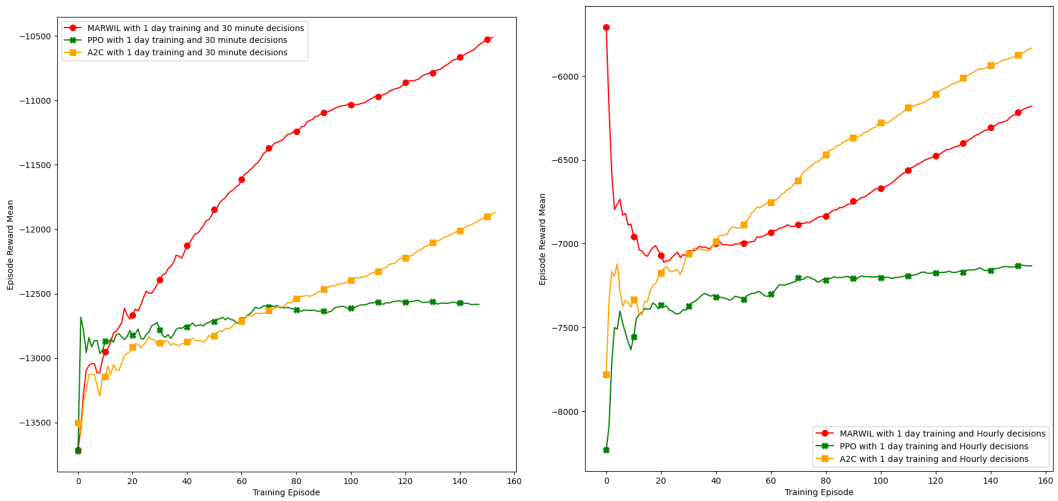

Fig. 3.15: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 1-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

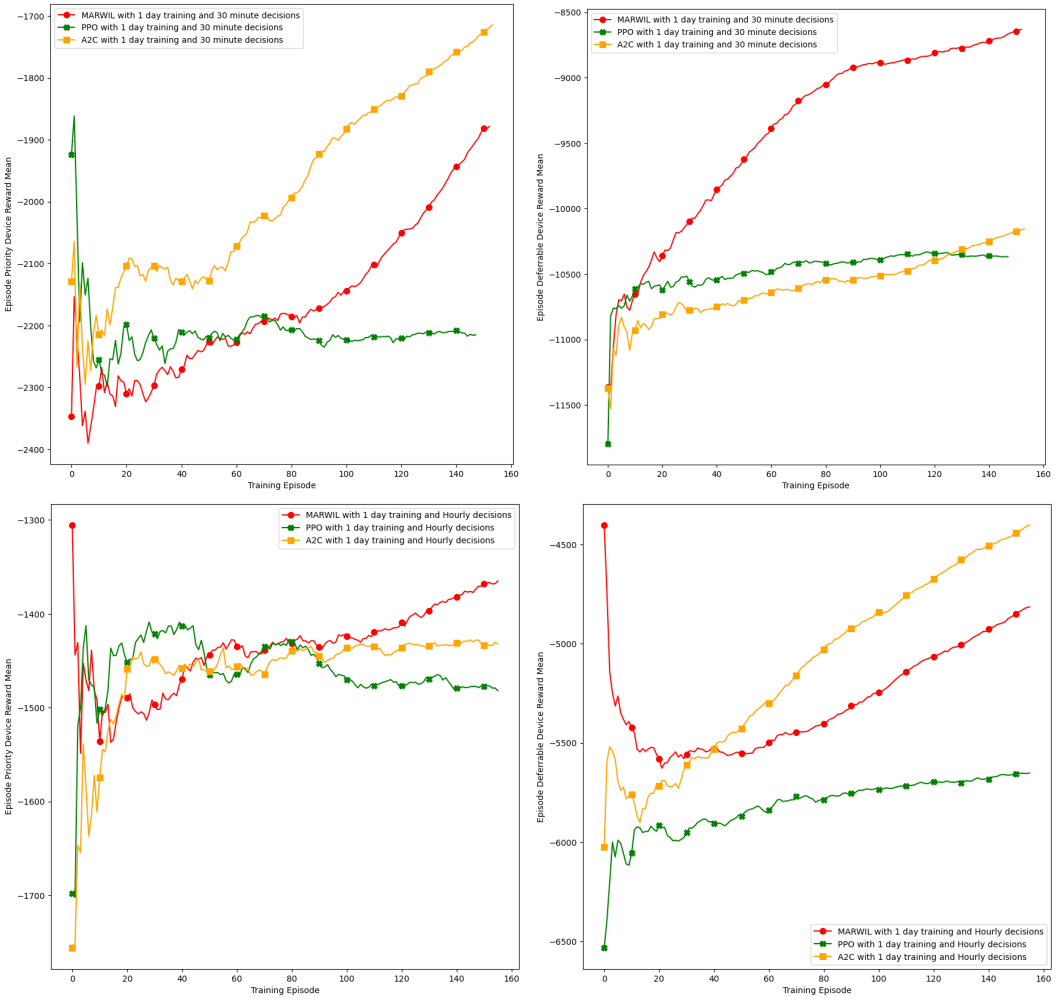

Fig. 3.16: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 1-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

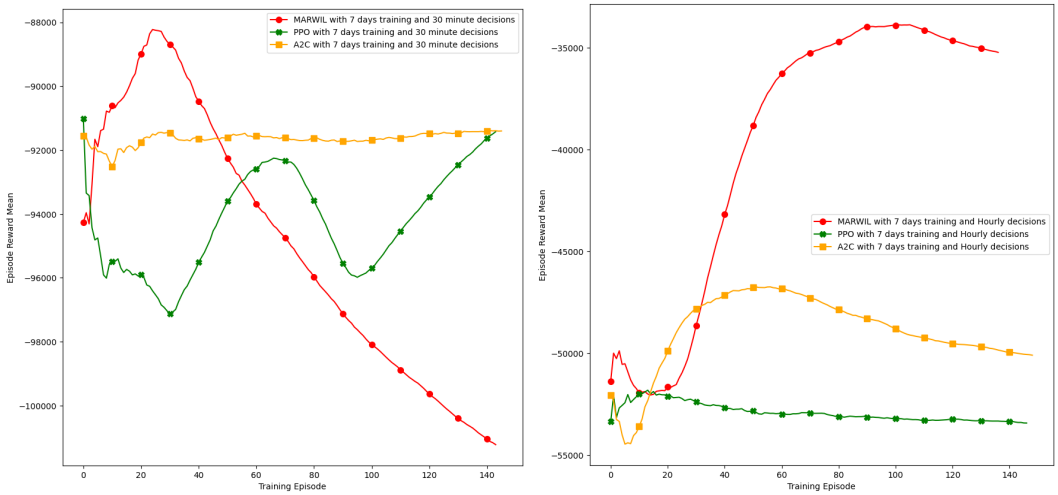

Fig. 3.17: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 7-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

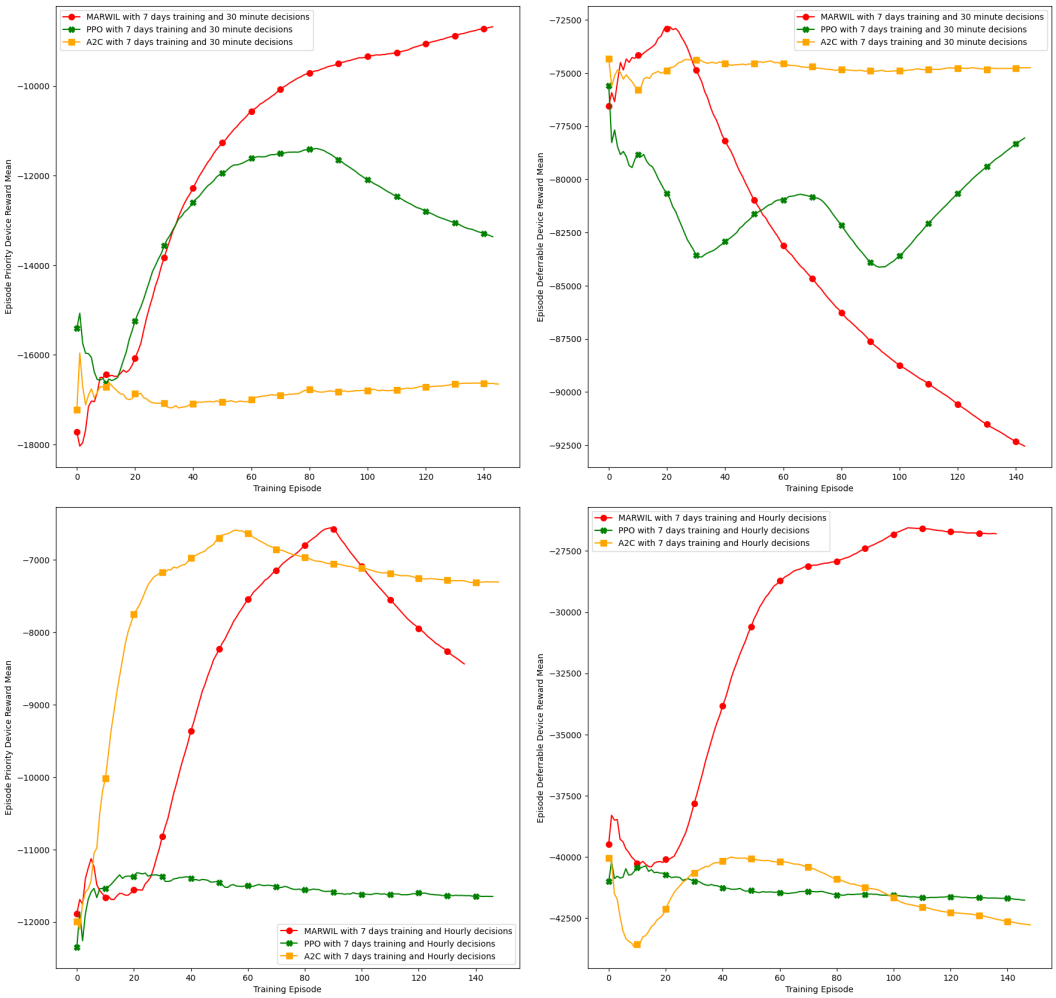

Fig. 3.18: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 7-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

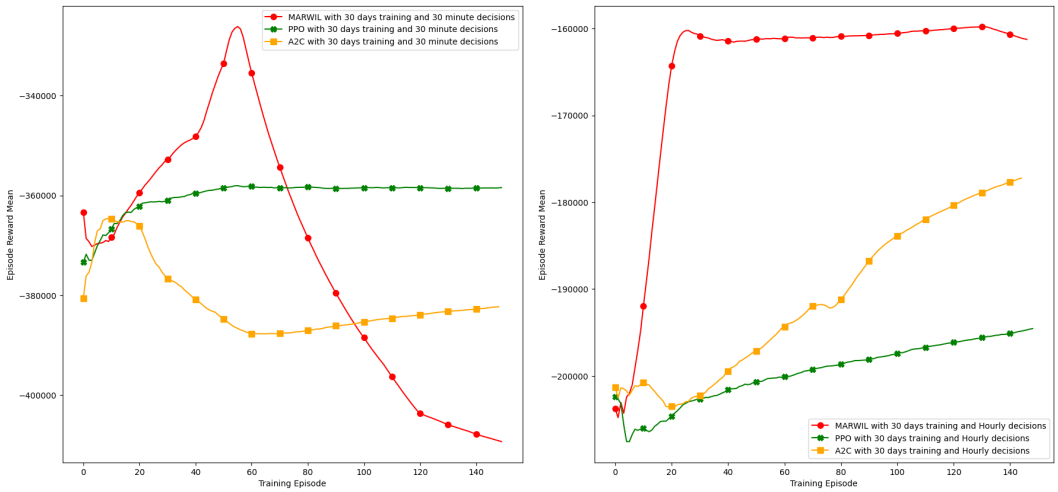

Fig. 3.19: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 30-day training episodes. On the left, the agent makes decisions every 30 minutes, while on the right, it makes decisions every 1 hour.

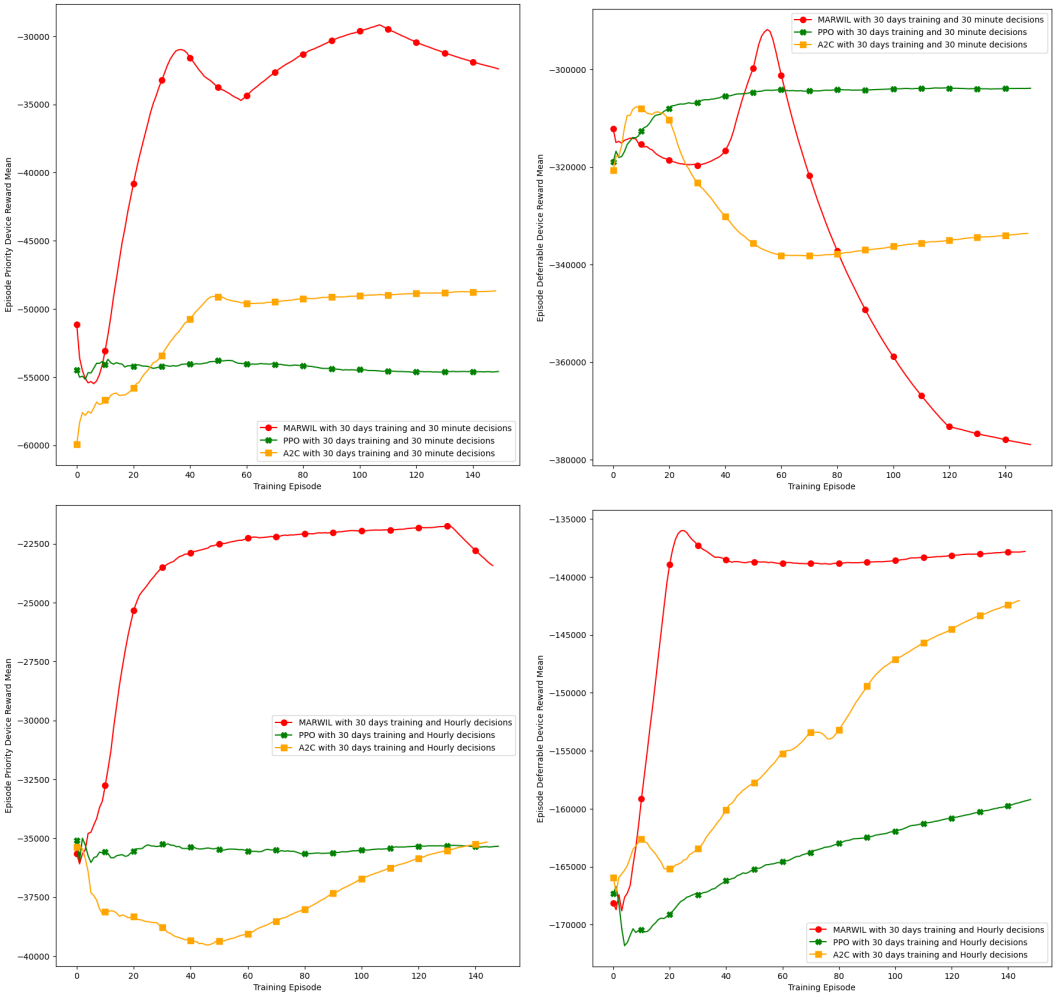

Fig. 3.20: Training Figures for the Environment with Five Priority and Five Deferrable Devices and 30-day training episodes. On the top left and right, the reward for priority and deferrable devices, respectively, is presented for the agent that makes decisions every 30 minutes. Similarly, on the bottom left and right the reward for priority and deferrable devices is presented, respectively, for the agent that makes decisions every 1 hour.

4 Conclusion

In this thesis, we demonstrate the potential that Reinforcement learning (RL) has to address the problem of home appliance management. Our work proposes a novel Residential Energy Manager (REM) that utilizes RL in order to adapt to the resident's habits and help reduce the electricity costs of the household. Moreover, we propose a novel household simulation environment in which the RL agent suggests actions for all electric devices collectively and takes into consideration possible spatial and temporal correlations between appliances and hidden residential behavioral patterns. In addition, we offer a fully customizable and open source environment for researchers that applies to most real-case household scenarios and the opportunity to test this environment on countless custom user profiles. The experiments we carried out in four different household setups, each with different complexity, with the Long Island 2018-2019 electricity price data set provided by the New York Independent System Operator, showcased the strengths and weaknesses of three state-of-the-art RL algorithms. The policy-based PPO algorithm was unable to manage priority and deferrable devices and was therefore deemed inappropriate for this complex problem with hybrid action spaces. The A2C algorithm presented better results, but ultimately failed to perform in the most realistic scenario, the household with five priority devices and five deferrable devices. The MARWIL algorithm had the best results of all three and showed extremely promising results in handling all household devices, adapting to the resident, and shifting power usage to hours with low electricity prices.

In the near future, we would like to further study the area of data analytics and Machine learning in the smart grid sector and especially test our proposed approach with more types of users and even more complex and realistic households. We would like to introduce a third category of devices commonly found in households, known as 'flexible devices,' such as Electric Vehicle chargers, which can adjust their energy consumption. Furthermore, we would also like to test other state-of-the-art RL algorithms that have potential in this field. Finally, it would be ideal if we could completely transition our research away from simulations, apply it to real-life residences, and use user feedback to improve our models.

Bibliography

[1] Mehdi Ahrarinouri, Mohammad Rastegar, and Ali Reza Seifi. "Multiagent Reinforcement Learning for Energy Management in Residential Buildings". In: *IEEE Transactions on Industrial Informatics* 17.1 (2021), pp. 659–666.

[2] Sanjiv Bhatia. "Adaptive K-Means Clustering." In: Jan. 2004.

[3] Ed Burns. *What is artificial intelligence (AI)?* URL: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence (visited on Feb. 18, 2023).

[4] Joseph M. Carew. *reinforcement learning*. URL: https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning (visited on Feb. 18, 2023).

[5] Spiros Chadoulos, Iordanis Koutsopoulos, and George C. Polyzos. "Mobile Apps Meet the Smart Energy Grid: A Survey on Consumer Engagement and Machine Learning Applications". In: *IEEE Access* 8 (2020), pp. 219632–219655.

[6] Eurostat. *Energy consumption in households*. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Energy_consumption_in_households (visited on Feb. 16, 2023).

[7] National Geographic. *Nonrenewable Resources*. URL: https://education.nationalgeographic.org/resource/nonrenewable-resources/ (visited on Feb. 16, 2023).

[8] Stephen Haben, Colin Singleton, and Peter Grindrod. "Analysis and Clustering of Residential Customers Energy Behavioral Demand Using Smart Meter Data". In: *IEEE Transactions on Smart Grid* 7.1 (2016), pp. 136–144.

[9] IBM. *What is artificial intelligence (AI)?* URL: https://www.ibm.com/topics/artificial-intelligence (visited on Feb. 18, 2023).

[10] Iordanis Koutsopoulos and Leandros Tassiulas. "Optimal Control Policies for Power Demand Scheduling in the Smart Grid". In: *IEEE Journal on Selected Areas in Communications* 30.6 (2012), pp. 1049–1060.

[11]Jungsuk Kwac, June Flora, and Ram Rajagopal. “Household Energy Consumption Segmentation Using Hourly Data”. In: *IEEE Transactions on Smart Grid* 5.1 (2014), pp. 420–430.

[12]Fintan McLoughlin, Aidan Duffy, and Michael Conlon. “A clustering approach to domestic electricity load profile characterisation using smart metering data”. In: *Applied Energy* 141 (2015), pp. 190–199.

[13]Elena Mocanu, Decebal Constantin Mocanu, Phuong H. Nguyen, et al. “On-Line Building Energy Optimization Using Deep Reinforcement Learning”. In: *IEEE Transactions on Smart Grid* 10.4 (2019), pp. 3698–3708.

[14]United Nations. *What is renewable energy?* URL: `https://www.un.org/en/climatechange/what-is-renewable-energy` (visited on Feb. 16, 2023).

[15]Daniel O’Neill, Marco Levorato, Andrea Goldsmith, and Urbashi Mitra. “Residential Demand Response Using Reinforcement Learning”. In: *2010 First IEEE International Conference on Smart Grid Communications*. 2010, pp. 409–414.

[16]David Petersson. *supervised learning*. URL: `https://www.techtarget.com/searchenterpriseai/definition/supervised-learning` (visited on Feb. 18, 2023).

[17]Eftychios Protopapadakis, Maria Kaselimi, Anastasios Doulamis, and Nikolaos Doulamis. “Non-Intrusive Load Monitoring for Personalized Strategies Towards Energy Consumption Reduction”. In: *2019 14th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)*. 2019, pp. 1–6.

[18]Salman Sadiq Shuvo and Yasin Yilmaz. “Home Energy Recommendation System (HERS): A Deep Reinforcement Learning Method Based on Residents’ Feedback and Activity”. In: *IEEE Transactions on Smart Grid* 13.4 (2022), pp. 2812–2821.

[19]George C. Polyzos Spiros Chadoulos Iordanis Koutsopoulos. “Deep4Ener: A universal deep learning approach for consumer-level energy demand forecasting with scarce data”. In: *To appear in ACM SIGEnergy Energy Informatics Review* (2023).

[20]Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[21]Qing Wang, Jiechao Xiong, Lei Han, et al. “Exponentially Weighted Imitation Learning for Batched Historical Data”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, et al. Vol. 31. Curran Associates, Inc., 2018.

[22]Zheng Wen, Daniel O’Neill, and Hamid Maei. “Optimal Demand Response Using Device-Based Reinforcement Learning”. In: *IEEE Transactions on Smart Grid* 6.5 (2015), pp. 2312–2324.

List of Acronyms

RL Reinforcement learning

DR Demand Response

EM Energy Manager

REM Residential Energy Manager

ML Machine learning

A2C Advantage Actor-Critic

PPO Proximal Policy Optimization

MARWIL Monotonic Advantage Reweighted Imitation learning

List of Figures

List of Tables