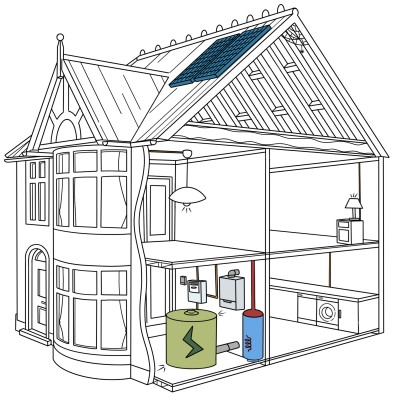
KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Reinforcement Learning with Knowledge Transfer for Residential Demand Response



Thijs Peirelinck

Supervisors:
Prof. dr. ir. G. Deconinck
Dr. C. Hermans
Dr. ir. F. Spiessens

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

May 2022

Reinforcement Learning with Knowledge Transfer for Residential Demand Response

Thijs PEIRELINCK

Supervisors:
Prof. dr. ir. G. Deconinck
Dr. C. Hermans
Dr. ir. F. Spiessens

Examination committee:
Prof. dr. ir. H. Van Brussel, chair
Prof. dr. ir. R. V. Sabariego
Prof. dr. ir. J. Suykens
Prof. dr. ir. T. Holvoet
Prof. dr. ir. C. Develder
(University of Ghent)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

May 2022

Preface

First of all I would like to thank my supervisors, both at KU Leuven and VITO. Thank you Prof. Geert Deconinck for providing guidance and feedback when necessary, without limiting my freedom to discover my own research interests. Thank you Chris Hermans, for always taking the time to have a conversation for as long as needed to fully explore an idea. And, thank you Fred Spiessens for your thorough feedback and for having confidence in me at the start of my PhD. I would like to thank all three of you for your support. Furthermore, I would like to thank VITO for funding my research.

I would also like to thank all members of the examination committee for their insightful feedback. You all took the time to thoroughly review my work and your comments and questions definitely improved the end result.

Over the past years, I have had the pleasure to work with a lot of talented, interesting and fun colleagues at ELECTA and EnergyVille. I can confidently say that I enjoyed every single coffee- and lunch break. It was a joy to be able to clear my mind form time to time, but also to have a technical conversation when needed.

Obviously, my friends deserve to be mentioned here as well. Over the years, I have had many evenings and weekends were I needed to clear my mind. I would like to thank all my friends that provided me with a possibility to do this, be it through climbing, hiking or visiting a pub with me. I would also like to thank my friends for helping me putting my work in perspective when I got carried away.

If not for the support of my parents, this research would have never happened. Thank you *moeke* and *vake* for giving me all needed support throughout the whole journey that has brought me here, at the end of my PhD. And, of course, thank you Lien, Toon and Joost for your unconditional support.

Last, but not least, I would like to thank Annelies. Meeting you during the

course of this PhD was a real blessing. It has undoubtedly been very valuable that I had someone to share time with when I was euphoric over positive experiment results. But, you were also there when I just wanted to quit. No one else has had to listen to me talk about energy as much as you. I would like to thank you for showing real interest in my work and for taking the time to fully understand my too detailed explanations.

Thijs Peirelinck, May 2022.

Abstract

One of the main challenges of the energy transition is the increasing need for demand flexibility due to the intermittency of renewable energy sources. While electrification of heating and transport further increases the need for sustainable energy sources, it also provides a source of flexibility. Power-to-heat technologies, such as heat pumps and electric water heaters, offer decoupled demand for heat and power, and thus provide a source of flexibility at the residential level.

Demand Response (DR) programs aim to activate demand flexibility by encouraging end-users to adapt their energy consumption based on grid signals. Over the past years, many countries have started to re-organise their electricity market(s) to activate dormant residential flexibility. For example, in the European Union every customer is entitled to an electricity contract with time-varying prices.

To harness this flexibility, researchers have been looking at Reinforcement Learning (RL). The main benefit of this control paradigm is its ability to solve complex control problems without the need for an environment model. This property mitigates several challenges of residential DR. Unfortunately, RL is relatively data inefficient. This manifests itself in extensive training times. During this initial period, the RL agent shows poor performance.

Inspired by recent efforts to improve RL data efficiency, this work builds upon transfer learning algorithms and contributes to their application in DR. This dissertation proposes several knowledge transfer learning approaches that increase data efficiency of RL algorithms, both in single- and multi-agent settings. This work shows how RL agents can be pre-trained in simulation, either to perform the same task in practice or to perform the same task for a different appliance. Furthermore, this work presents a method to incorporate domain knowledge into RL agents, without restricting their applicability. By means of these approaches, RL agents show good control performance immediately after deployment and, consequently, increased user comfort.

Beknopte samenvatting

Reinforcement Learning met kennisoverdracht voor toepassingen van residentiële vraagsturing

Een van de grootste uitdagingen van de huidige energietransitie is de stijgende behoefte aan flexibiliteit langs de vraagzijde. Deze flexibiliteit is nodig omdat het productieprofiel van hernieuwbare energiebronnen vaak moeilijk te voorspellen is en een wisselend karakter heeft. Bovendien zorgt de stijgende graad van elektrificatie van onze verwarming en ons transport ervoor dat we in de toekomst meer van die hernieuwbare bronnen zullen nodig hebben. Gelukkig bieden elektrische wagens, elektrische boilers en warmtepompen ook een grote mate van flexibiliteit. Dit komt omdat zij, door hun energiebuffer, voor een ontkoppeling van de warmtevraag en elektriciteitsvraag zorgen.

Door middel van verschillende signalen, en met behulp van vraagsturing, kunnen eindgebruikers aangemoedigd worden om hun flexibiliteit actief te gebruiken en hun verbruik aan te passen aan de noden van het elektriciteitsnet. In de voorbije jaren zijn verschillende landen begonnen met het herorganiseren van hun elektriciteitsmarkt. Dit met als doel het activeren van flexibiliteit die tot op heden niet gebruikt werd. Zo werd het binnen de Europese Unie onlangs ingeschreven dat het een recht is van elke consument om een dynamisch contract af te sluiten met zijn energieleverancier.

Om deze flexibiliteit te kunnen exploiteren kijken onderzoekers meer en meer naar Reinforcement Learning (RL). Het grootste voordeel van deze regelmethodologie is de mogelijkheid om complexe regelsystemen op te lossen zonder dit regelsysteem wiskundig te moeten beschrijven. Deze eigenschap biedt een oplossing voor enkele van de grootste uitdagingen op het vlak van residentiële vraagsturing. Jammer genoeg gaat RL vrij inefficiënt om met de beschikbare data. Dit resulteert in lange perioden van slechte sturing omwille van het onvoltooid leerproces.

Geïnspireerd door recente nieuwe ontwikkelingen die de data-efficiëntie van RL

verhogen, bouwt dit werk voort op algoritmes die kennisoverdracht mogelijk maken en draagt het bij aan de toepassing van deze algoritmes voor vraagsturing. Dit proefschrift stelt enkele mogelijkheden voor om kennis over te dragen tussen verschillende systemen en zo de data-efficiëntie van RL te verhogen, zowel voor systemen met één enkele agent, als voor systemen met meerdere agenten. De resultaten in dit proefschrift tonen aan dat RL-agenten in een simulatie kunnen worden voorbereid op het regelsysteem dat ze in de praktijk zullen moeten aansturen. Tevens kunnen agenten ook worden voorbereid op het uitvoeren van dezelfde taak, maar voor een ander apparaat. Dit werk toont ook aan dat het mogelijk is om domeinkennis van een expert rechtstreeks in te bouwen in de RL-agent. En dit op een manier waarbij de toepasbaarheid niet gelimiteerd wordt tot één enkel huishouden. Al de hierboven genoemde methoden dragen bij aan het verhogen van de data-efficiëntie van RL-algoritmes. Door middel van de voorgestelde werkwijzen wordt de prestatie van de RL-regeling, tijdens de eerste weken na de installatie, sterk verhoogd. Dit resulteert in verhoogd comfort voor de eindgebruiker en komt tegemoet aan de toenemende noden van de energiemarkt.

List of Abbreviations

DHW Domestic Hot Water.

DNO Distribution Network Operator.

DQL Double Q-learning.

DR Demand Response.

EWH Electric Water Heater.

FQI Fitted Q-Iteration.

HC Hysteresis Control.

HEMS Home Energy Management System.

HVAC Heating, Ventilation and Air Conditioning.

MAFQI Model-Assisted Fitted Q-Iteration.

MDP Markov Decision Process.

MILP Mixed Integer Linear Problem.

ML Machine Learning.

MMP Mean Month Peak.

MPC Model Predictive Control.

NN Neural Network.

POMDP Partially Observable Markov Decision Process.

PPO Proximal Policy Iteration.

PV photovoltaic.

RBC Rule-Based Control.

RES Renewable Energy Sources.

RL Reinforcement Learning.

SC Self-Consumption.

SDP Sequential Decision making Problem.

SoC State of Charge.

TCL Thermostatically Controlled Load.

ToU Time-of-Use.

VREG Vlaamse Regulator van de Energie- en Gasmarkt.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The increasing share of renewable energy sources in the electricity grid of today has raised major challenges. The intermittent nature of these energy sources reduces the available flexibility at the generation side. Simultaneously, most decarbonisation scenarios for the energy sector are built on electrification of heating and transport. This, however, significantly increases electricity demand, and the variability in generation is expected to increase sharply with more renewables in the grid. Taken together, these have the potential to disrupt stable electric grid operation [79]. These problems appear on multiple levels. For customers, this often means increasing energy costs, while also leading to voltage and power flow issues on the distribution grid [36]. On the transmission side, it can lead to frequency issues caused by inertia loss and steep ramp rates [67], as well as an increase in capacity requirements [38]. The latter is particularly dangerous as the contracted capacity is often in the form of polluting peaking plants, which counteracts the decarbonization objectives [22].

Batteries and Demand Response (DR) are considered enabling technologies in the energy transition. These technologies facilitate the shift towards more sustainable means of electricity generation and accelerate electrification of heating and transport by mitigating the challenges related to intermittent renewable energy sources. DR aims to exploit the flexibility potential of consumers by controlling energy consumption of loads. In this regard, Thermostatically Controlled Loads (TCLs) are very promising due to their inherent possibility to shift electricity consumption in time, while still being

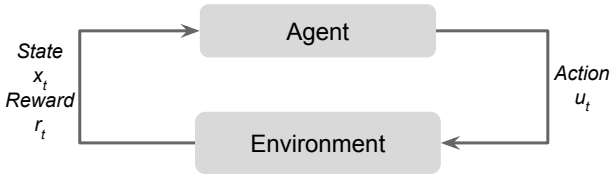


Figure 1.1: Agent-environment interaction in reinforcement learning.

able to satisfy heat demand. For example, an Electric Water Heater (EWH) can provide hot water for some time after the buffer has been heated.

While the sole purpose of a residential battery can be the provision of flexibility, TCLs only provide flexibility as a secondary goal. Their main aim is to provide heat. However, herein also lays their strength, they inherently add a source of flexibility to a residential consumer. There is no need for an extra investment. Unfortunately, actively controlling TCLs is challenging, especially in residential cases. To maximize impact on the electrical system, in this setting, large sets of small loads have to be controlled. Modeling the dynamics of each individual appliance in such a cluster is time-consuming and expensive [8, 70]. Therefore, in the quest for optimizing the usage of demand flexibility, researchers have been looking at artificial intelligence techniques. Model-free control methodologies, like RL, seem a valuable alternative [71, 70, 38, 9]. Yet, although RL mitigates the system-identification problem, several challenges and questions remain.

1.2 Challenges of Reinforcement Learning for Demand Response

DR control is a sequential decision-making problem which can best be formulated as a Markov Decision Process (MDP). Just like existing (and future) markets for energy and flexibility, the MDP formalism uses discrete time-steps at which action-controlled state transitions occur. These transitions come at varying costs and may have uncertain effects. The operator, or so-called agent, is in charge of choosing the *best* action, where *best* is defined by a reward-function. This function gives the agent a notion of the state transition quality. Everything which is exogenous to the agent is called the environment. Figure 1.1 shows the interactions between agent and environment [82].

Model Predictive Control (MPC) has been proposed multiple times to solve this decision-making problem [40]. In DR it has been used in a variety of use cases,

such as day-ahead scheduling of loads [5]. Moreover, in the specific case of TCL control it has achieved promising results. For example, to reduce space heating costs [8, 12, 26, 80]. Unfortunately, MPC has several drawbacks. By using a model of the physical system and assuming this reflects actual dynamics, results heavily depend on the accuracy of the model [64]. Additionally, while developing the controller, approximately 60% of human effort is spent on modeling [8]. Grey-box models and system-identification do not mitigate this pitfall [64]. The mentioned drawbacks all show there is potential for other approaches to solve the MDP.

RL is a machine learning technique very well suited to optimize MDPs. An important advantage over MPC is that it does not need models: it works with data samples. Recent advances have made it possible to apply RL to a large range of problems, including (residential) DR. Economically, residential DR is only compelling when a whole cluster of flexible loads (appliances) is managed [25]. Due to the heterogeneous nature of all these appliances the system-identification (modeling) step is exceptionally challenging. This has drawn DR researchers to RL. Mbuwir *et al.* [45] use RL in a battery management system of a micro-grid. The same algorithm has been used by Ruelens *et al.* [69, 71] to control a heat pump and EWH, respectively. De Somer *et al.* [13], on the other hand, use it for optimal self-consumption of local PV generation. This proves the versatility of RL algorithms and shows RL is a promising strategy for distributed control: by means of distributing control complexity, a whole cluster of loads can be controlled.

Although the previous examples look promising, RL also has its deficiencies. Arguably one of its main weaknesses is data inefficiency. State-of-the-art methods are still sample inefficient and require, for example, 29 million games of Go before reaching an above-human level [74]. Data is mostly not abundantly available in residential households. Neither is it likely users will allow long training times nor actions that disrupt their comfort heavily.

Recent advances have shown that techniques such as transfer and semi-supervised learning can considerably improve the performance of machine learning models, used in RL agents [2]. Such formulations allow models to leverage existing data, domain knowledge and human expertise [9, 35]. The biggest advantage of transfer learning is that it reduces the data complexity of machine learning models [101]. More specifically, by leveraging domain knowledge and/or previously gathered data, machine learning models tend to perform better with fewer data points, learn faster as more data becomes available [56], and achieve higher asymptotic performance than their naive counterparts [62].

Due to these benefits, transfer learning can enable large scale real-world roll-out of automated DR programs. This ranges from improved forecast and dynamics

models to more efficient reinforcement learning agents.

1.3 Challenges, Research Questions and Objectives

Challenges

In an effort to mitigate the challenges with respect to the human effort necessary for manual DR, researchers have looked at control methods for increasing the level of automation. Automated DR using model-free RL has mitigated certain challenges with respect to the modelling effort needed in traditional optimization techniques. However, general residential DR challenges as well as challenges related to the usage of RL in DR remain.

- **Heterogeneity**: In residential DR their are two main sources of heterogeneity: the TCLs and the users. There are a lot of different appliances, with different power ratings and dimensions available in today's market and different households have different behaviour. Therefore, the proposed solution should be generally applicable to all sorts of TCLs and be independent of user behaviour.
- **Scalability**: Combined with the heterogeneous nature of residential appliances a large-scale residential DR setting quickly becomes intractable. The proposed solution should, therefore, be able to scale well in terms of the amount of appliances (both in number and in type) that can be controlled simultaneously.
- **Modelling costs**: Implementation of Rule-Based Control (RBC) or model-based control requires domain experts. These rules and models need to be adapted for specific use cases and appliances. The cost of such domain expertise is not negligible. If a certain solution requires expert knowledge this would tamper its cost-effective large scale implementation.
- **Data-efficiency**: Finally, data is relatively scarce in residential applications. Additionally, users expect a working solution from the start and their comfort should be guaranteed. RL always requires a certain amount of training time. On top of this, RL algorithms trade-off exploration and exploitation of gained knowledge. At the start, explorative actions will be taken by the agent quite often. This is because state-of-the-art learning algorithms always start from scratch, without any knowledge about the problem at hand. This is not efficient and does not correspond with our intuitive notion of learning. Furthermore, these explorative actions can be counter-intuitive and challenge the user acceptance of the

control approach. Therefore, the proposed solution should efficiently use the available data and work as expected by the user, as soon as possible after installation.

- **RL algorithm design and parameter tuning**: In recent years, RL has gained a lot of attention due to promising research results. More often than not, good results are praised while design and parameter tuning efforts are played down. Algorithms, (hyper)parameters and reward-functions often need to be tweaked to the problem at hand. Therefore, a lot more domain knowledge, from both RL and the application domain, is needed than most expect.

Research Questions

1. Which residential DR settings would benefit from RL based control and how are consumers incentivised to participate in these DR programs?
2. How can we design cost-effective and generally applicable methods that benefit maximally from available data *before* the agent is deployed at the consumer's site?
3. If available data is not sufficient to guarantee performance from the start of operation, how can we incorporate domain knowledge in a general way, *i.e.*, without the need for extensive modelling?

Objectives

Based on the identified challenges and resulting research questions, the following objectives have been proposed for the present work:

1. Formalise a control problem for the different DR settings that exist or will be implemented in the near future.
2. Extend and design RL algorithms facilitating knowledge transfer from simulation to practice.
3. Extend and design RL algorithms for efficient adaptation to changing contexts.
4. Extend and design RL algorithms that incorporate domain knowledge in the learning pipeline, without restricting the applicability.

1.4 Main Contributions

Given the stated objectives, this dissertation advances the current state-of-the-art in a variety of ways.

1.4.1 Dissertation Contributions

- This dissertations identifies different residential DR settings which will be of practical importance in the upcoming years and formalises their optimisation problems.
- During this work different sensory input possibilities for TCLs have been identified. Their impact on control performance has empirically been compared by using and benchmarking them as input for RL agents in DR settings.
- This dissertation presents an extensive discussion on the progress and impact of transfer learning within DR settings. Based on this discussion, important next steps have been identified and explored.
- This dissertation shows that it is possible to pre-train RL agents in simulation by using general hot water buffer models. It does so by extending both Fitted Q-Iteration (FQI) and Double Q-learning (DQL), two RL algorithms, with domain randomization. The latter is a technique used to pre-train RL agents through randomised models. The results in this dissertation show that domain randomization does not only vastly increase data-efficiency of both algorithms, but also increases overall control performance of the agent. Additionally, using domain randomization, a pre-trained agent can generalise to unseen buffer dynamics.
- A novel multi-agent extension of FQI with experience replay (*i.e.*, saving and using past transitions), that allows for efficient adaptation to a changing environment, is presented. The proposed method allows to jumpstart performance of new agents entering the system.
- In this dissertation, Flanders' (Belgium) recently introduced capacity tariff is explored. An RL agent that successfully manages to exploit flexibility potential to reduce the end-consumer's energy bill, has been implemented.
- This dissertation shows RL agents can be pre-trained with artificially generated datasets, based on consumer statistics. These *average* RL agents are able to generalise to specific households. This allows to use readily available climate and consumer data that is collected for other purposes.

- In this work a novel extension of the algorithm Proximal Policy Iteration (PPO) has been proposed. This extension allows to incorporate domain knowledge in a non-restrictive way, by shaping the control policy beforehand. The results show above human-level control performance.
- Although tuning RL parameters and algorithms to different applications has shown to be one of the remaining challenges of RL. In this dissertation multiple algorithms have been adapted for several DR applications, as shown by the above contributions.

1.4.2 Scientific Output

The above mentioned contributions have lead to various collaborations with colleagues and have resulted in several submitted and accepted publications at scientific conferences and in (academic) journals.

Journals

- T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Domain Randomization for Demand Response of an Electric Water Heater". In: *IEEE Transactions on Smart Grid* 12.2 (May 2020), pp. 1370–1379. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.3024656
- T. Peirelinck, H. Kazmi, B. V. Mbuwir, C. Hermans, F. Spiessens, J. Suykens, and G. Deconinck. "Transfer learning in demand response: A review of algorithms for data-efficient modelling and control". In: *Energy and AI* 7 (2022), p. 100126. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2021.100126
- *(To be submitted)* T. Peirelinck, C. Hermans, F. Spiessens, G. Deconinck,"Combined Peak Reduction and Self-Consumption Using Proximal Policy Optimization", 2022

Conference Proceedings

- T. Peirelinck, F. Ruelens, and G. Deconinck. "Using reinforcement learning for optimizing heat pump control in a building model in Modelica". In: *2018 IEEE International Energy Conference (ENERGYCON)*. IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-3669-5. DOI: 10.1109/ENERGYCON.2018.8398832

- C. Patyn, T. Peirelinck, and G. Deconinck. “Intelligent Electric Water Heater Control with Varying State Information”. In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–7. ISBN: 9781538679548. DOI: 10.1109/SmartGridComm.2018.8587453
- T. Peirelinck, F. Spiessens, C. Hermans, and G. Deconinck. “Double Q-learning for Demand Response of an Electric Water Heater”. In: *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2019. DOI: 10.1109/ISGTEurope.2019.8905776
- T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. “Transfer learning for Demand Response of a Multi-Agent Battery and Electric Water Heater System”. In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2021. DOI: 10.1109/ISGTEurope52324.2021.9640081

Non-scientific Journals

- T. Peirelinck, "Zelflerend Systeem voor een Verwarmingsinstallatie", Sanilec, 2018.

1.5 Outline

Figure 1.2 gives an overview of the outline of this dissertation. Chapters 2 and 3 formally present the mathematical framework and challenges of this work. Chapter 4, 5 and 6 each present a different approach of transfer learning in DR applications. More specifically, the organisation of this dissertation is as follows:

- **Chapter 2 - Reinforcement Learning in Demand Response -** Formal presentation of the control problem and all its aspects. The models of the appliances and the main DR tasks are presented. Important aspects with respect to the user comfort guarantees are discussed. Additionally, the most important aspect of the RL algorithms implemented throughout this work are presented.
- **Chapter 3 - Transfer Learning in Demand Response -** Literature review of transfer learning applications within DR. Transfer learning and the difference with conventional machine learning is explained. Furthermore, a formal taxonomy of transfer learning is proposed. Using

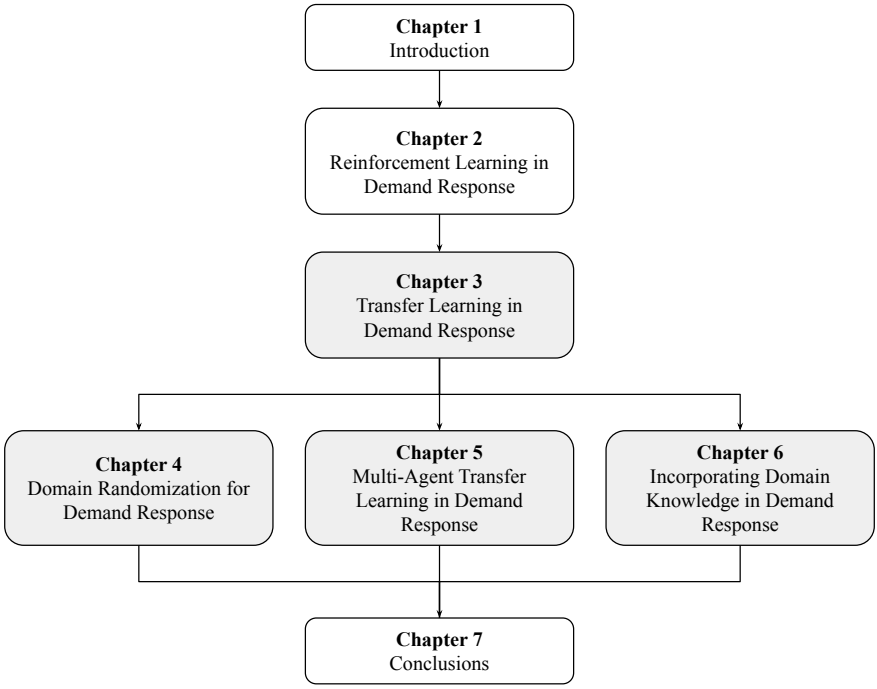


Figure 1.2: Outline of this dissertation.

this taxonomy future research opportunities have been identified. This work has been disseminated in a journal paper [58].

- **Chapter 4 - Domain Randomization for Demand Response -** An extension of FQI and DQL, as they were presented in Chapter 2, is presented. The benefits of domain randomization for transfer learning are discussed. Different experiments, following the control problem and DR tasks of Chapter 2, show that the proposed methods yield significant performance gains, with a lasting cost reduction throughout the simulation period. The presented extension has also been disseminated in a journal paper [56]

- **Chapter 5 - Multi-agent Transfer Learning in Demand Response -** A strategy for knowledge transfer between agents in a multi-agent DR setting is proposed. The presented experiments show that, using this strategy, new agents can jumpstart their performance. This strategy has been disseminated through a conference presentation and paper [57].

- **Chapter 6 - Incorporating Domain Knowledge in Demand Response Learning Problems -** A novel method to incorporate domain knowledge in actor-critic RL algorithms is presented. PPO, one such algorithm, introduced in Chapter 2, is adapted following the presented approach. The experiments show that households can successfully reduce their peak power consumption (and thus electricity bill), using this expert adapted version of PPO. Especially interesting is that it concerns general expert knowledge. Once adapted the trained model can be applied to different households, with vastly different user behaviour.
- **Chapter 7 - Conclusions -** Final overview of the dissertation. Summarizes all the main conclusions and presents future research opportunities and perspectives.

Chapter 2

Reinforcement Learning in Demand Response

This chapter is partly based on the work presented in the following publications:

- [54] C. Patyn, T. Peirelinck, and G. Deconinck. "Intelligent Electric Water Heater Control with Varying State Information". In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–7. ISBN: 9781538679548. DOI: 10.1109/SmartGridComm.2018.8587453
- [56] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Domain Randomization for Demand Response of an Electric Water Heater". In: *IEEE Transactions on Smart Grid* 12.2 (May 2020), pp. 1370–1379. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.3024656
- [57] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Transfer learning for Demand Response of a Multi-Agent Battery and Electric Water Heater System". In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2021. DOI: 10.1109/ISGTEurope52324.2021.9640081

2.1 Markov Decision Process

All control problems in this work are Sequential Decision making Problems (SDPs). Therefore, they can be formulated as MDPs. MDPs represent a way

to formally define all problems involved with an agent that learns to achieve a pre-defined goal from interactions with its environment [82].

The control problems considered in this work use either an EWH or a battery as example, although they are more generally applicable to all TCLs (and batteries). The agent has to control an EWH (battery) with the goal of maximising cumulative reward. It can do so by shifting the EWH (battery) energy consumption in time. For the EWH case, Domestic Hot Water (DHW) always needs to be available when the user requests it. To comply with this constraint, the outlet water temperature T_{out} of the EWH is required to stay above a minimum temperature T_{min}.

As the agent interacts with its environment, every time step t this interaction results in a state transition, according to transition function $f_T(x_t, u_t, P^{u_t}_{x_t x_{t+1}})$, with $x_t \in \mathcal{X}$ the current state, $u_t \in \mathcal{U}$ the current action and $P^{u_t}_{x_t x_{t+1}}$ the state transition probabilities. Furthermore, each state transition comes with a certain reward, as defined by the reward function $r(x_t, u_t, x_{t+1})$. In the remainder of this work, we assume a discrete-time (in)finite horizon MDP, with step-size $\Delta t = 15$ minutes.

As is typical in residential DR settings, the exact transition probabilities are unknown. The goal of the RL agent is to learn a policy $\pi : \mathcal{X} \to \mathcal{U}$ which can cost-efficiently operate the environment, given this uncertainty. In other words, the agent should maximise the expected discounted reward of policy π, $J(\pi)$ (2.1), with discount factor $\gamma = 0.99$, $r_{t+1} = r(x_t, u_t, x_{t+1})$ and T the optimization horizon [62]

$$J(\pi) = \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right]. \tag{2.1}$$

We call

$$V(x) = \mathbb{E} \left[\sum_{t=0}^{T-1} \gamma^t r_{t+1} \right] \tag{2.2}$$

the value function. The goal of the agent is thus to find a policy π which maximises its expected cumulative discounted reward. The following subsections introduce each part of the MDP.

2.1.1 State-space

Based on the environment's current state, the agent decides which action to execute. The agent can only observe certain values of the state features, as not all state information is directly observable/sensable. The observed state-space is

thus a subset of the EWH's full state space: $X^O \subset X$. Therefore, this problem is actually a Partially Observable Markov Decision Process (POMDP) and the Markov property does not hold [82]. This means the current observed state is not only dependent on the previous state but also on earlier states. As proposed in [82], adding a history h of state features will allow the agent to recover the Markov property. The resulting augmented state $x^{aug} \in X^{aug}$ is described in 2.3.

$$x^{aug} = \left[\left[x_a^t, x_a^{t-1}, ..., x_a^{t-h}\right], ..., \left[x_z^t, x_z^{t-1}, ..., x_z^{t-h}\right]\right] \tag{2.3}$$

where $x_a^t, ..., x_z^t$ are the features assumed to be observable by the agent.

In what follows, whenever the state-space $\mathcal{X}$ has been mentioned, it concerns the augmented observable state-space. Throughout this work, the state-space varies a bit between use cases. However, the smallest common divider includes a notion of time, an estimate of the appliance's State of Charge (SoC) and assumes $h = 4$. This configuration has empirically shown to result in the best performance [54]. Thus, at every time-step $t \in \{0, \ldots, 96\}$ the agent observes state x_t, given by (2.4).

$$x_t = \left[SoC_{t-3}, SoC_{t-2}, SoC_{t-1}, SoC_t, t_{\cos}, t_{\sin}\right]. \tag{2.4}$$

With $t_{\cos}$ and $t_{\sin}$ the projection of the time-step onto a circle, as in (2.5).

$$t_{\sin} = \sin\left(2\pi \cdot t/96\right) \tag{2.5a}$$

$$t_{\cos} = \cos\left(2\pi \cdot t/96\right) \tag{2.5b}$$

Section 2.2 presents the various simulation models used and how their SoC has been determined. Note that, while these models are needed to simulate the system and analyse the performance of an agent, they have been used solely as a virtual test-bed. The agent has no information about the internal formulations whatsoever. It can only observe the state.

2.1.2 Action-space

At each time step t, the agent takes a binary control action $u \in \mathcal{U} = \{0, 1\}$ in order to turn the heating element of the EWH on or off. In the physical world this corresponds to drawing 0 or P_r kW of electric power during a period $\Delta t = 15$ minutes, assuming rated power P_r.

2.1.3 Reward-function

The DR setting determines the reward function. Throughout this work three different DR settings, and thus reward functions, have been considered: energy

arbitrage, self-consumption and peak power pricing. All three tasks and their resulting reward function are discussed in section 2.3 of this chapter.

2.2 Appliances

Appliances of most interest for DR applications are flexible and use a considerable amount of energy relative to the overall household (electrical) energy consumption. Therefore, TCLs have been identified to be especially promising [70]. Throughout this work an EWH has mainly been used as an example appliance. Additionally, the environment in Chapter 5 also includes a small battery. The models of both appliances are presented and discussed next.

2.2.1 Electric water heater models

Throughout this work we use three buffer models. We will start by introducing them and end with using them for calculating water temperature, which clearly shows where each model resembles or differs from the others.

All three models rely on the heat balance equation. As a result, they share mostly the same parameters and variables, summarised in Table 2.1 and 2.2. The buffer has rated power P_r and tank volume V. While hot water is tapped at the top, it is replaced by cold water at the bottom, where the heating element is located. Total boiler capacitance C_{boiler} equals $V \cdot C_{pw}$, with C_{pw} the specific heat capacitance of water.

Tap demand $\dot{m}_w$ has been obtained from hot water tap profiles at a 5 minute resolution [15]. The daily average DHW consumption equals 189 l/day. Edwards *et al.* [15] state it corresponds to an average consumption level of a household mainly demanding hot water in the evening.

All buffers include a hysteresis controller, which preserves user comfort and turns the heating element on when water temperature drops below $T_{\min}$ and off when it raises above $T_{\max}$. We assume this controller cannot be by-passed. External control is only possible when the water is in between these temperature limits. This backup controller is presented in subsection 2.2.3. Requested water T_{asked}, inlet water T_{iw} and ambient T_{amb} temperature are assumed constant.

We assume the internal SoC of the buffer can be calculated from the available measurements [90]. The SoC expresses a measure of the buffer energy content, relative to its minimal and maximal allowed energy stored. Hence, $SoC = 0$

Table 2.1: Buffer model parameters.

Name	Explanation	Value	Unit
C_{boiler}	Buffer capacitance	798142	J/K
U	Buffer thermal transmittance	0.75	W/(m^{2}K)
C_{pw}	Water specific heat capacitance	4182	J/(kgK)
h	Buffer height	1.2	m
d	Buffer diameter	0.45	m
A_s	Buffer side surface	1.7	m^2
A_t	Buffer top/bottom surface	0.16	m^2
V	Buffer volume	190.85	kg
P_r	Rated power	2400	W
T_{amb}	Ambient temperature	20	°C
T_{iw}	Inlet water temperature	17	°C
T_{min}	Minimal buffer temperature	55	°C
T_{max}	Maximum water temperature	70	°C
T_{asked}	Asked output water temperature	45	°C

Table 2.2: Buffer model variables.

Name	Explanation	Unit
$\dot{Q}_{\text{heat}}$	Heat flow rate heating element	J/s
$\dot{m}_w$	Hot water tap demand	kg/s
T_i	Water temperature (of layer i)	°C
SoC	State of Charge	[-]

occurs when the whole water content is at temperature T_{min} and $SoC = 1$ when it is at T_{max}. Water colder than T_{min} is not considered.

Uniform buffer model

We use the buffer model as presented by Farooq *et al.* [18]. This model describes the heat balance of a mass of water, with temperature T_L. It models three processes: heat supplied by the heating element, transfer of heat by the inlet water and losses to the environment. The model is given by

$$\frac{dT_L}{dt} = \frac{1}{C_{\text{boiler}}}[\dot{Q}_{\text{heat}} + \dot{m}_w C_{pw}(T_{iw} - T_L) + U(A_s + 2A_t)(T_{\text{amb}} - T_L)]. \tag{2.6}$$

All symbols are defined in Table 2.1 and 2.2. Since the water content of the buffer is assumed to have a uniform temperature, the SoC is defined as (2.7).

$$SoC = \frac{T_L - T_{\min}}{T_{\max} - T_{\min}}. \tag{2.7}$$

Two-layer buffer model

To take into account thermal stratification we can extend the previous model with a second water layer [76]. Sinha *et al.* [76] model 2 separate layers, each with their own temperature. T_h and T_c for top (hot) and bottom (cold) layer, respectively ($T_c \leq T_h$).

Sinha *et al.* argue that a thermocline exists inside the buffer, which limits mixing between top and bottom layer. As a consequence, they assume a hard boundary between both layers. When both layers reach the same temperature, they merge. Both layers diverge when hot water is tapped and, therefore, cold water is added at the bottom. As more cold water is added, the cold (hot) layer volume increases (decreases) and the boundary between hot and cold layer moves up. We call m_c the cold layer mass and V_c the cold layer volume. We add another model variable M indicating if the layers are merged ($M = 1$) or not ($M = 0$). We define the share of cold and hot water as in (2.8) and (2.9), respectively.

$$X_c = \frac{V_c}{V} \tag{2.8}$$

$$X_h = \frac{V - V_c}{V} = 1 - X_c \tag{2.9}$$

Heat balance is defined by (2.10) and (2.11), with $F = 1 - (1 - M)(1 - X_c)$. Thus $F = X_c$ when $M = 0$ and $F = 1$ when $M = 1$.

$$\begin{aligned}\frac{dT_h}{dt} = \frac{1}{X_h C_{\text{boiler}}}[&M\dot{Q}_{\text{heat}} + M\dot{m}_w C_{pw}(T_{iw} - T_h) \\ &+ U(A_s X_h + A_t + MA_t)(T_{\text{amb}} - T_h)]\end{aligned} \tag{2.10}$$

$$\begin{aligned}\frac{dT_c}{dt} = \frac{1}{F C_{\text{boiler}}}\ [&\dot{Q}_{\text{heat}} + \dot{m}_w C_{pw}(T_{iw} - T_c) \\ &+ U((1 - M)A_s X_c + MA_s + A_t + MA_t)(T_{\text{amb}} - T_c)]\end{aligned} \tag{2.11}$$

The cold layer mass and the binary variable M are then determined by (2.12) and (2.13), respectively.

$$m_c = \begin{cases} 0 & M = 1 \text{ and } \dot{Q}_{\text{heat}} \neq 0 \\ \int \dot{m}_w dt & \dot{Q}_{\text{heat}} = 0 \end{cases} \tag{2.12}$$

$$M = \begin{cases} 1 & \text{if } T_h = T_c \text{ or } V \leq V_c \\ 0 & \text{else} \end{cases} \tag{2.13}$$

Heat balance equations (2.10) and (2.11) can more easily be interpreted by separating them in three.

1. When the layers are merged ($M = 1$) and no hot water is tapped ($\dot{m}_w = 0$) both layers have an equal temperature. The model reduces to the uniform case, *i.e.*, (2.6) with $\dot{m}_w = 0$.
2. When the layers are merged ($M = 1$), hot water is tapped ($\dot{m}_w > 0$) but the heating element is on ($\dot{Q}_{\text{heat}} \geq 0$), no thermocline is created. Again, the model reduces to (2.6), now with $\dot{Q}_{\text{heat}} \neq 0$.
3. Otherwise, the temperature of both layers is described by (2.10) - (2.11), with $M = 0$, $F = X_c$ and $m_c = \int \dot{m}_w dt$.

Using this model we define the buffer SoC as in (2.14), with $L = \{i \in \{c, h\} | T_i \geq T_{\min}\}$ and $V_i = X_i \cdot V$ for $i \in \{c, h\}$.

$$SoC = \frac{\sum_{i \in L} V_i (T_i - T_{\min})}{V(T_{\max} - T_{\min})} \tag{2.14}$$

Stratified buffer model

We can take the idea of modelling separate layers of water even further. Here, we model $N = 50$ different water layers in the buffer. The stratified buffer model we use has been presented and validated in the lab by Vanthournout *et al.* [90]. This model has been used in DR research several times [54, 61, 69], and for more details we refer to Vanthournout *et al.* [90].

In contrast with the two-layer model's variable layer size, here each layer has an equal and constant volume V/N. Furthermore, the temperature T_i of each layer $i \in \{0, \ldots, N\}$ is uniform and monotonically increasing with buffer height ($T_i \leq T_{i+1}$). Vanthournout's model also considers heat exchange between layers.

Equation (2.14) can also be applied to the stratified model SoC [90], with $L = \{i \in \{1, \ldots, N\} | T_i \geq T_{\min}\}$ and $V_i = V/N$ for all $i \in \{1, \ldots, N\}$.

Comparison

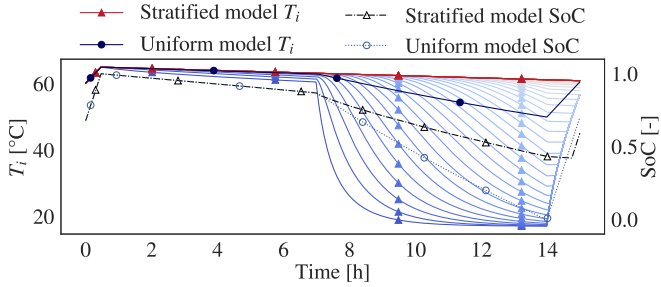
The three models have been simulated according to the same tapping pattern, and Fig. 2.1 shows their T_i and SoC. Fig 2.1a shows the water temperature of the uniform and stratified buffer on the left axis. At the start, temperature is 60 °C and the heating element is turned on for 40 minutes. Until hour 7, the buffers are left idle, which shows standing heat losses to the environment. The next 7 hours hot water is drawn at 2 ml/s. While taking hot water the stratified model's layers diverge. This highlights the difference between both models. When the heating element is turned on again the stratified model's layers heat from bottom to top, since the heat source is at the bottom of the tank. The right axis depicts the SoC of both models. With no water demand and when all layers of the stratified model have an approximately equal temperature, SoC of both models is almost identical. However, when water is drawn, the uniform model's SoC differs from the stratified model. This underestimation of SoC occurs since the uniform model does not capture stratification. While in the stratified model a large portion of layers is still relatively hot, the uniform model assumes the whole buffer's water temperature decreases.

This effect is mitigated by modelling a second layer. Fig. 2.1b shows SoC of the two mass and stratified model match more closely. In the two mass model, as soon as hot water is tapped a second layer is created with temperature $T_c = T_{iw}$. Notice that after hour 7, while the cold layer's temperature is constant, its volume is increasing, resulting in SoC decrease. Fig. 2.2 visualizes the simplification made by modelling 2 instead of 50 layers. It shows buffer section, *i.e.*, temperature in function of height at different times. Comparing hour 9 and 13 shows the mentioned effect of constant cold layer temperature but raising thermocline, when hot water is being tapped. At hour 14 the heating element is turned on, layers are heated from top to bottom. Due to the two mass model's larger share of water at temperature T_{iw}, this results in a buffer section as shown in the rightmost graph of Fig. 2.2 at 14h30.

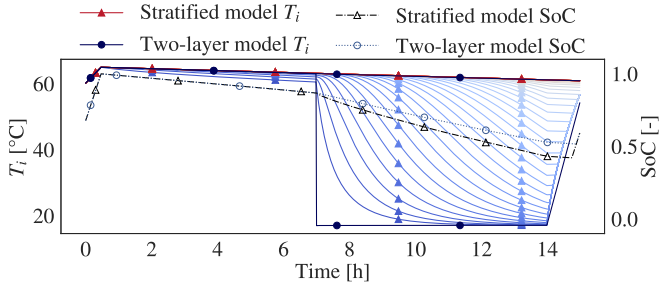
2.2.2 Battery Model

An energy based battery model has been assumed for this work. The energy balance in the battery is defined by (2.15), with energy content E^B, battery rated power P_r^B, (dis)charge effiency η and u^B_{phys} depending on the action as in (2.18).

$$\frac{dE^B}{dt} = \eta u^B_{\text{phys}} \tag{2.15}$$



(a) Layer temperature(s) and SoC of uniform and stratified buffer model.



(b) Layer temperatures and SoC of two-layer and stratified buffer model.

Figure 2.1: Comparison between buffer models. First hour (h = 00:00 - 00:40): heating element is turned on no tap demand ($\dot{Q}_{\text{heat}} > 0$ and $\dot{m}_w = 0$). Hour 0:40 to 7:00: no heating and no tap demand ($\dot{Q}_{\text{heat}} = 0$ and $\dot{m}_w = 0$). Hour 7:00 to 14:00: no heating and small tap demand ($\dot{Q}_{\text{heat}} = 0$ and $\dot{m}_w = 2$ ml/s). Finally, h = 14:00 - 14:30: $\dot{Q}_{\text{heat}} > 0$ and $\dot{m}_w = 0$.

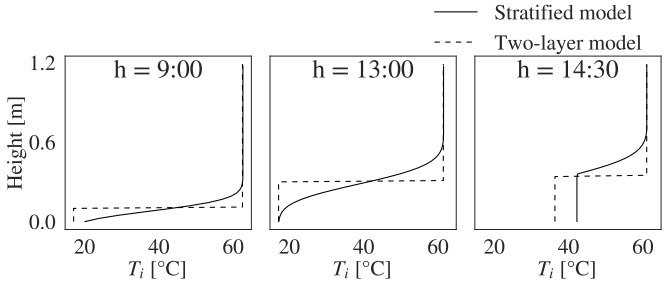


Figure 2.2: Section of two mass and stratified buffer model at hour 9, 13 and 14 : 30.

The energy content of the battery is limited to $E^B_{\max}$. The battery SoC is then given by

$$SoC^B = \frac{E^B}{E^B_{\max}}. \tag{2.16}$$

2.2.3 Particularities: The Backup Controller

User comfort is very important and should always be guaranteed. At first sight, it seems reasonable to incorporate user satisfaction in the reward function. Unfortunately, even a large penalty does not always guarantee user comfort, due to necessary exploration steps within every RL algorithm. Therefore, Ruelens *et al.* [70] proposed to incorporate a backup controller in the environment itself. In RL literature such a backup controller is known as a *shield* [1]. A shield monitors the actions of the RL agent and corrects them in case they violate safety. Or, in this case, comfort.

The EWH's internal (backup) controller maps the agent's action u to a physical action u_{phys} (*i.e.*, power setpoint) according to (2.17), with T_b the temperature at the backup sensor location.

$$u_{\text{phys}} = H(u) = \begin{cases} P_r & T_b \leq T_{\min} \\ uP_r & T_b > T_{\min} \text{ and } T_b < T_{\max} \\ 0 & T_b \geq T_{\max} \end{cases} \tag{2.17}$$

The battery's backup controller overrules the battery-agent's action when the buffer is full, empty or the agent is not allowed to consume or produce from the battery. These overrule actions occur because we assume it is not allowed to perform energy arbitrage solely from the grid, *i.e.*, one cannot discharge energy from the battery to the grid or charge the battery directly from the grid. The battery backup controller acts according to (2.18), with P_{net} the net power consumption of the household.

$${u_{\text{phys}}}^B = H^B(u) = \begin{cases} \min(0, uP^B_r) & E^B_{t+1} > E^B_{\max} \\ \max(0, uP^B_r) & E^B_{t+1} < E^B_{\min} \\ 0 & P_{\text{net}} \cdot u > 0 \\ \min(|P_{\text{net}}|, uP^B_r) & P_{\text{net}} < 0, u > 0 \\ \max(-P_{\text{net}}, uP^B_r) & P_{\text{net}} > 0, u < 0 \end{cases} \tag{2.18}$$

While $H(u)$ and $H^B(u)$ are unknown to the agent, their effect can indirectly be observed through the cost-function.

2.3 Tasks

DR is a general term indicating all use cases where a certain demand is responsive to external signals. Not all of these DR use cases bring about challenging control problems. However, some of the most interesting and practical do. As mentioned in the previous chapter, especially in the residential sector these control problems quickly become intractable and thus very challenging to solve. This thesis focuses on three major residential demand response tasks. These three tasks require special care. Not only due to their practical interest but also due to the challenging control problem they result in.

2.3.1 Energy Arbitrage

The concept of energy arbitrage is well understood. It is probably one of the most intuitive forms of DR. Energy arbitrage is the act of consuming during off-peak, or low-cost, times in order to avoid having to consume during peak, or high-cost, times [70]. The goal is thus to shift (electrical) energy consumption to low-cost times. Currently, dynamic pricing electricity contracts are increasingly being introduced across the world, with the digital or smart meter as key enabler. For example, the Clean Energy Package of the European Union states that all European consumers with a digital meter should have at least one supplier that offers a dynamic pricing contract in their market [17].

It is thus clear that residential energy arbitrage is becoming ever more important. The goal of this work is to develop methods that enable residential consumers to act on these changing prices with the flexible appliances at their disposal. Because, from a grid point of view it is not sufficient that consumers have such contracts. They also should (be able to) act on the incentives that these contracts provide.

Figure 2.3 shows the mean Belgian day-ahead wholesale electricity price (dashed line) of 2021. As an example, imagine this represents the prices included in a dynamic pricing contract of a Belgian consumer. The full gray line depicts a typical Belgian household's consumption profile throughout the day. With these prices and this consumption profile, this consumer would pay €1.10 per day, on average, summing up to € 402.66 for an entire year (in reality, taxes and transmission costs should be added). If this same consumer would shift some of its consumption from in between $6-8$h and $14-17$h to $9-11$h and $20-22$h, this would result in a cost reduction of approximately € 12 per year. Their new consumption profile is shown by the dotted grey line in figure 2.3.

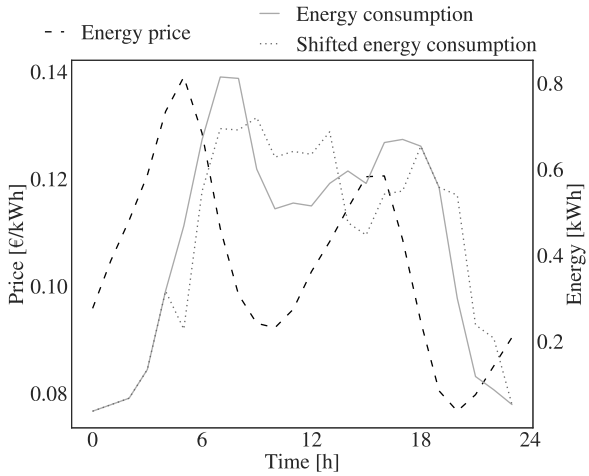


Figure 2.3: Example of how a typical Belgian household could shift its electrical energy consumption in an energy arbitrage scenario.

More formally, and following the MDP framework, given a Time-of-Use (ToU) price λ and energy consumption E the reward r_t (or cost c_t) at time-step t for the energy arbitrage DR setting is given by:

$$r_t = -c_t = -\lambda E_t. \tag{2.19}$$

2.3.2 Self-consumption

Here, self-consumption is considered the act of changing one's energy consumption based on the availability of on-site (behind the meter) energy production. While self-consumption has always been a hot topic in the Belgian media, for residential consumers there has never been a real incentive for it. At least not for instantaneous self-consumption. Until recently, Belgian residential consumers all were equipped with a Ferraris disc meter. As this meter turns backwards whenever there is net production, (instantaneous) self-consumption is no different from consuming some energy and producing a similar amount of energy at a later point in time. In both these cases the meter reading at the end would be the same. However, it is clear that from the electricity grid point of view, these two situations are completely different.

In recent years, more households have been equipped with a digital meter. Net-metering is thus disappearing. As a result, a feed-in tariff has been introduced [14]. This feed-in tariff is, however, lower than the energy consumption price [95].

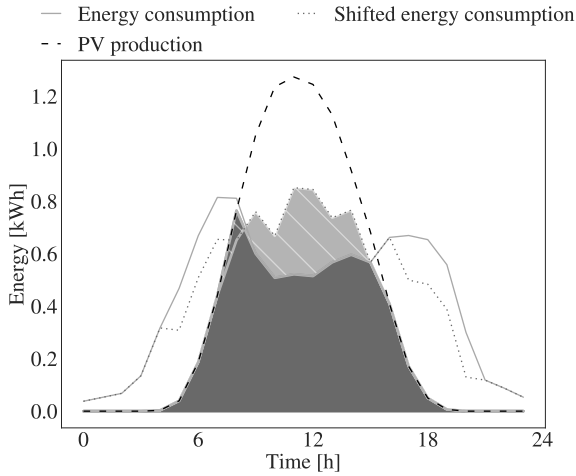


Figure 2.4: Example of how a typical Belgian household could shift its electrical energy consumption to improve self-consumption.

At the same time, energy consumption also results in transmission costs and taxes. These additional costs can be avoided with self-consumption. Therefore, self-consumption has now become interesting for residential consumers who invested in a photovoltaic (PV) installation.

As a visual example, Figure 2.4 again shows the typical Belgian household's (electrical) energy consumption. Additionally, this figure shows the production profile of an average residential PV installation in Leuven's climate (dashed line). The self-consumed energy is shaded grey. To increase this amount, *i.e.*, self-consumption, this household could shift some of its energy consumption to the afternoon. An example of their new, improved, power profile is given by the dotted line. The additional self-consumption due to this change is hatched.

While we do care about the overall self-consumption of a household, we cannot control all aspects of the household's energy consumption. Therefore, we assume our agent is only able to control certain flexible appliances, *e.g.*, an EWH. As a result, the reward function design is based on maximising self-consumption of this appliance alone. Mathematically, the reward function represented by self-consumption of an EWH is given by (2.20), with E^{PV} the PV installation energy production, E^{m} the inflexible energy consumption and E^{EWH} the EWH energy consumption. The intuition behind this reward function is to maximise the part of the EWH energy consumption that can be covered with PV production after

the inflexible load has been subtracted.

$$r_t = \min(\max(E_t^{PV} - E_t^m, 0), E_t^{\text{EWH}}) \tag{2.20}$$

2.3.3 Peak Power Pricing

A final DR task that has been considered in this work is peak power pricing. The Belgian (or rather Flemish) use case is again used as an example. At the time of writing (Q1/2022), grid costs in Flanders are billed according to a household's energy consumption, *i.e.*, in €/kWh. The Flemish energy and gas market regulator (Vlaamse Regulator van de Energie- en Gasmarkt (VREG)) has, however, decided this will change in the near future (Q3/2022). They argue that grid costs are more tied to power consumption than energy consumption. *E.g.*, grid equipment, such as transformers and cables, dimensioning is based on maximal power consumption, rather than energy. Therefore, residential consumers in Flanders will soon have an incentive to reduce their power consumption.

While peak power reduction is tied to self-consumption, it is not exactly the same. In self-consumption one is only interested in shifting demand for energy to time-slots when energy production is available. In a peak reduction scenario, one is also interested in shifting power demand to times when overall demand is lower. Figure 2.5 shows the net power consumption of the same household as the previous two figures (dashed line). The peak of this profile, depicted by a grey dot, can be lowered by performing the same shift as in Figure 2.4, and increase self-consumption. This resulting net power consumption is depicted by the dotted line. While the household might not be flexible enough to further increase self-consumption, it could potentially still reduce its peak power consumption by a small consumption shift from hour 18 to 20 and from hour 19 to 20, as shown by the full line.

Intuitively, given net power consumption P_{net}, an agent could be penalised for behaviour that results in big power peaks by a cost function that is proportional to the peak. However, as the VREG has decided a minimal tariff applies for everyone with a peak less than 2.5 kW, there is no need to try to further reduce power consumption. Therefore, this DR results in the following reward function:

$$r_t = \min(2.5 - P_{net}, 0). \tag{2.21}$$

Chapter 6 presents the Flemish tariff design in more depth.

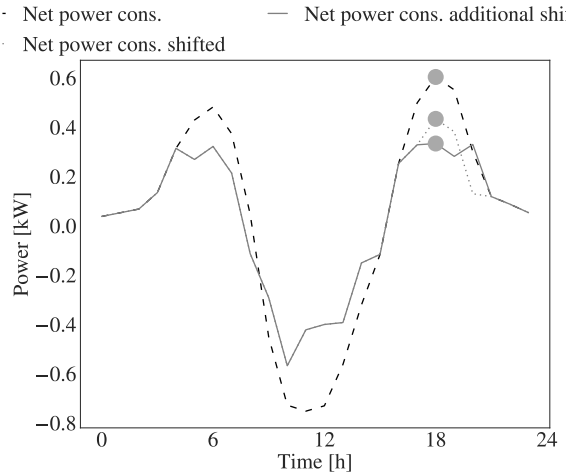


Figure 2.5: Example of how a typical Belgian household could shift its electrical energy consumption to reduce its power peak.

2.4 Algorithms

RL is the study of algorithms to solve SDPs, where a learner needs to discover which actions to take in order to maximise expected reward [82]. Therefore, RL is not a single algorithm. Rather, it is a class of solution methods. Over time different RL algorithms have been designed and used on a variety of problems. In the course of this work, three such algorithms have been implemented, adapted and tested for their suitability on the above mentioned DR use cases. The following sections each present one such algorithm.

2.4.1 Q-learning

A popular approach to solve the optimization problem as given in (2.1) is to define the state-action value function

$$Q(x_t, u_t) = r_{t+1} + \gamma \max_u Q(x_{t+1}, u). \tag{2.22}$$

The Q-function represents the value of taking action u_t when the environment is in a certain state x_t. Since we are interested in maximising our value over time, the optimal policy is to choose the action that gives maximum expected reward. This intuition determines policy $\pi(x)$ according to (2.23).

$$\pi(x) = \operatorname*{argmax}_u Q(x, u) \tag{2.23}$$

The basic idea behind many RL algorithms is to estimate the Q-function using an iterative strategy. The Bellman equation (2.24) defines the iterative update rule [82].

$$Q_{i+1}(x_t, u_t) = \mathbb{E}\left[r_{t+1} + \gamma \max_u Q_i(x_{t+1}, u)\right] \tag{2.24}$$

Using this update rule Q_i converges to the optimal policy when $i \to \infty$.

Mnih *et al.* [46] introduced the usage of non-linear function approximation, such as Neural Networks (NNs), for estimating the Q-function. Throughout this work two variations of Q-learning have been used.

Unfortunately, due to the usage of non-linear function approximation, all convergence guarantees are lost [82]. Some exploration is necessary in order to avoid converging to a local optimum. Here, we use an ϵ-greedy exploration strategy, *i.e.*, random actions are taken with probability ϵ [82]. Over time ϵ decreases.

Double Q-learning (DQL)

DQL has been proposed, and later updated, by Van Hasselt et al. [88, 89]. It incorporates two NNs: the online network, with parameters θ_t at time step t, and the target network, with parameters θ_t^-. Van Hasselt [89] proposes the following update rule for the online network

$$Y_t = r_{t+1} + \gamma Q(x_{t+1}, \operatorname*{argmax}_u Q(x_{t+1}, u; \theta_t); \theta_t^-). \tag{2.25}$$

The targets Y_t are calculated using uniform samples from experience replay memory $\mathcal{F}$, containing transitions of the form $\{x_t, u_t, x_{t+1}, r_{t+1}\}$. And are used to update θ^t. We use update rule (2.26) [37], with $\tau = 0.1$, for the target network's parameters.

$$\theta_t^- = \tau\theta_t^- + (1-\tau)\theta_t \tag{2.26}$$

This means that both NNs are updated according to Algorithm 1.

Fitted Q-Iteration (FQI)

Due to its proven track record in DR applications, we also consider an adapted version of FQI [69, 38, 54] in our experiments. In a discrete-time MDP with a one day horizon ($T = 96$) we can reformulate the SDP as a sequence of T control problems. Approximating the Q-function $Q_t(x_t, u_t)$ at each time step t gives rise to T supervised learning problems. Assuming $Q_T = r_{T+1}$, Q-values

Algorithm 1 Double Q-learning (DQL)

Input: τ, γ, α, stop condition
Initialise $Q(x, u; \theta_t)$, $Q(x, u; \theta_t^-)$
while not done **do**
    $u_t = \epsilon$-greedy $(\text{argmax}_u\, Q(x_t, u; \theta_t))$
    Observe: $\{(x_t, u_t, x_{t+1}, r_{t+1})\}$
    $Y_t = r_{t+1} + \gamma Q(x_{t+1}, \text{argmax}_u\, Q(x_{t+1}, u; \theta_t); \theta_t^-)$
    $Q(x, u; \theta_{t+1}) \leftarrow Q(x, u; \theta_t) + \alpha\,(Y_t - Q(x, u; \theta_t))$
    $\theta_{t+1}^- = \tau\theta_{t+1}^- + (1 - \tau)\theta_{t+1}$

at time steps $t \in \{0, \ldots, T-1\}$ are defined as in equation (2.27).

$$Q_t(x_t, u_t) = r_{t+1} + \gamma \max_u Q_{t+1}(x_{t+1}, u) \tag{2.27}$$

FQI distinguishes itself from other value iteration algorithms by a special form of the experience replay technique, where all transitions seen so far are used in every iteration [68]. Furthermore, the update is performed off-line. The combination of these properties has shown to increase data efficiency compared to other value iteration algorithms. We thus calculate Q_t using all samples in $\mathcal{F}$ and represent it with any function approximator. In this work we often use a random forest [6] for all T Q-functions, as it has shown to be computationally more efficient than a NN, with limited performance sacrifices [43]. Furthermore, we denote an approximated Q-function, with regressor parameters θ, as $Q(x, u; \theta)$.

One of the main advantages of the finite optimization horizon is the ability to divide the SDP into T supervised learning problems, as this allows for iterative re-training of the policy. This iterative re-training of all Q-functions inherently allows a changing ToU-price profile, as rewards can be recalculated each iteration, which might be the case in certain DR settings.

The mentioned properties of FQI are thus a good fit for certain DR control problems. The usage of FQI in these problems can be separated in three steps. In a first step, the agent collects experiences, *i.e.*, tuples in the form $(x_t, u_t, x_{t+1}, r_{t+1})$. In the second step, the training batch is prepared using the reward function valid in the next optimization period. New rewards are calculated according to $r'_{t+1} = r(x_t, u_t, x_{t+1})$. In a third step, Q-values are calculated with the prepared batch of tuples $(x_t, u_t, x_{t+1}, r'_{t+1})$, going backwards though time and according to (2.27). Additionally, the function approximators are fitted using update rule (2.25). These steps have been summarized in Algorithm 2.

Algorithm 2 (Adapted) Neural Fitted Q Iteration (FQI)

Input: γ, $r(x_t, u_t, x_{t+1})$, $\{\lambda_t\}_{t=0}^{\infty}$, stop condition, T
Initialise $Q_0(x, u; \theta_0), \ldots, Q_T(x, u; \theta_T)$
while not done **do**
    Perform T steps, save transitions in $\mathcal{F} = \{(x_t, u_t, x_{t+1}, r_{t+1})\}$
    Take $\{\lambda_i\}_{i=t}^{t+T}$
    Let Q_{T+1} be zero everywhere on $\mathcal{X} \times \mathcal{U}$
    for $N = T, \ldots, 0$ **do**
        for $k = 1, \ldots, |\mathcal{F}|$ **do**
            $r_{k+1} = r(x_k, u_k, x_{k+1}, \lambda_N)$
            $Q_{N,k} \leftarrow r_k + \gamma \max_{u \in \mathcal{U}} Q_{N+1}(s_{k+1}, u)$
        $\theta_N \leftarrow \text{argmin}_\theta \frac{1}{2} \sum_k ||Q_N(x_k, u_k; \theta_N) - Q_{N,k}||^2$

2.4.2 Proximal Policy Optimisation (PPO)

Algorithm 3 Proximal Policy Optimization (PPO)

Input: h, γ, λ, ϵ, T, α_a, α_c, stop condition
Initialise θ_i^{actor}, θ_i^{critic}, $i = 0$
while not done **do**
    Perform T steps, save transitions in $\mathcal{F} = \{(x_t, u_t, x_{t+1}, r_{t+1})\}$
    Sample past T time steps from $\mathcal{F}$
    for $k = 1, \ldots, T$ **do**
        $g_k = \frac{\pi_{\theta_i^{\text{actor}}}(u_k|x_k)}{\pi_{\theta_{i-1}^{\text{actor}}}(u_k|x_k)}$
        $H_k = -\sum_{l=0}^{l=|\mathcal{U}|} \pi(u_l|x_k) \log \pi(u_l|x_k)$
        $L_k^{\text{actor}} = \min\left(g_k \hat{A}_k, \text{clip}(g_k, 1-\epsilon, 1+\epsilon)\hat{A}_k\right)$
        $Y_k^{\text{actor}} = L_k^{\text{actor}} - hH_k$
        $Y_k^{\text{critic}} = \sum_{j=k}^{T} \gamma r_{j+1}$
    $\pi(x; \theta_{i+1}^{actor}) \leftarrow \pi(x; \theta_i^{actor}) + \alpha_a \left(Y_k^{\text{actor}} - \pi(x; \theta_i^{actor})\right)$
    $V(x, \theta_{i+1}^{critic}) \leftarrow V(x, \theta_i^{critic}) + \alpha_c \left(Y_k^{\text{critic}} - V(x; \theta_i^{critic})\right)$
    $i++$

PPO [73] can be used to obtain a stochastic policy, maximising expected total reward, given a certain reward-function. It is an actor-critic algorithm. The family of actor-critic algorithms has been introduced in an leffort to combine the advantages of both policy-iteration and value-iteration algorithms. The actor *acts*, *i.e.*, it decides which action to take, and is trained based on a policy-iteration approach. The critic informs the actor about the value of certain

actions and how it should update its policy in the right direction. The critic is trained with a value-iteration algorithm. Both actor and critic are parametrised using a (separate) NN, with parameters θ_{actor} and θ_{critic}, respectively.

At every iteration k, the actor's objective L_k^{actor} (2.28) and the critic's objective L_k^{critic} (2.29) are (approximately) minimised.

$$L_k^{\text{actor}} = \hat{\mathbb{E}}_k \Big[\min \Big(g_k(\theta_{\text{actor}}) \hat{A}_k, \text{clip}(g_k(\theta_{\text{actor}}), 1-\epsilon, 1+\epsilon) \hat{A}_k \Big) \Big] \tag{2.28}$$

$$L_k^{\text{critic}} = \hat{\mathbb{E}}_k \left[\left(V_{\theta_k^{\text{critic}}}(x_k) - V_k^{\text{targ}} \right)^2 \right] \tag{2.29}$$

By using these update rules, an estimate of the value-function $V(x)$ and policy $\pi(x)$ is obtained by the critic and actor, respectively. In Eq. (2.28), $g_k(\theta)$ denotes the probability ratio and is defined by (2.30) [73]. $\hat{A}_k$ denotes an advantage estimate and is defined by (2.31) [72], with γ and λ hyper-parameters and δ_t^V given by (2.32). The clipping in update rule (2.28) avoids destructively large policy updates [73]. We use a clipping value of $\epsilon = 0.2$, and $\gamma = \lambda = 0.99$.

$$g_k(\theta) = \frac{\pi_\theta(u_k|x_k)}{\pi_{\theta_{\text{old}}}(u_k|x_k)} \tag{2.30}$$

$$\hat{A}_k = \sum_{l=0}^{\infty} (\gamma\lambda)(\delta_{t+l}^V) \tag{2.31}$$

$$\delta_t^V = r_t + \gamma V(x_t + 1) - V(x_t) \tag{2.32}$$

Thus, with PPO both NNs are trained following Algorithm 3.

Chapter 3

Transfer Learning in Demand Response

This chapter is based on

- [58] T. Peirelinck, H. Kazmi, B. V. Mbuwir, C. Hermans, F. Spiessens, J. Suykens, and G. Deconinck. "Transfer learning in demand response: A review of algorithms for data-efficient modelling and control". In: *Energy and AI* 7 (2022), p. 100126. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2021.100126

The supervised learning discussion has been omitted, as it is of limited importance for the remainder of this work.

3.1 Introduction

Recent advances have shown that techniques such as transfer and semi-supervised learning can considerably improve the performance of machine learning models – both in supervised and reinforcement learning contexts [2]. Such formulations allow models to leverage existing data, domain knowledge and human expertise [35, 9]. The biggest advantage of transfer learning is that it reduces the data complexity of machine learning models [101]. More specifically, by leveraging domain knowledge and/or previously gathered data, machine learning models tend to perform better with fewer tasks-specific data points, learn faster as

more data becomes available [56], and achieve higher asymptotic performance than their naive counterparts [62]. Due to these benefits, transfer learning can enable large scale real-world roll-out of automated DR programs. This ranges from improved forecast and dynamics models to more efficient reinforcement learning agents.

3.1.1 Previous Reviews

A number of recent reviews address machine learning and DR [49, 3], as well as how reinforcement learning relates to it [93]. However, the focus in these reviews is typically on general techniques, and not specifically on how to use transfer learning to operationalize them in practice. On the other hand, a number of transfer learning surveys have been presented in recent years, both on general transfer learning [97, 85] and on transfer within the reinforcement learning setting. None of these focus on applications within the smart grid or DR setting.

It is important distinguish between transfer learning and the broader field of informed Machine Learning (ML). Informed machine learning covers a broader range of possibilities to inform an ML agent. This can be through adding differential equations to the loss-function, simulation results, knowledge graphs, etc [96]. Furthermore, informed ML also incorporates what is referred to as physics informed ML [28]. This branch of ML aims to incorporate physical knowledge into the learning pipeline.

Transfer learning, on the other hand, focuses specifically on methods that inform an ML agent through transfer, mostly of data, model parameters or feature representations. In transfer learning, one thus typically distinguishes between the source domain (or task), *i.e.*, the source of the initial knowledge and the target domain (or task), *i.e.*, where the knowledge is used to improve performance. Thus, while informed machine learning consists of all genres to inform a machine learning agent, transfer learning is its subset, only focusing on those methods that inform ML agents by transferring data or model parameters from a source domain/task to a target domain/task. Very recently, Von Rueden *et al.* [96] presented an extensive literature review on informed machine learning. However, this literature review is general in its scope, and does not address the smart grid or DR case. In a subsequent section, we make the link between transfer and informed ML more explicit.

3.1.2 Organisation

The following section discusses the other transfer learning reviews more in depth and formalises a taxonomy based on the adopted definitions. The two sections thereafter present how the different DR settings fit into the taxonomy. Section 3.3 focuses on transfer learning in RL. Section 3.4 presents different transfer learning applications within the DR setting. Section 3.5 identifies future research opportunities. Finally, Section 3.6 concludes this review.

3.2 Taxonomy of Transfer Learning Algorithms

3.2.1 Conventional Machine Learning

In conventional machine learning, a functional mapping is learned between input and output variables for a specific problem using a well-defined dataset. The only expert or domain knowledge in this case is typically in the learning pipeline, *i.e.*, in the choice of learning algorithm or feature selection. Expert knowledge can therefore enable the practitioner to identify the correct learning algorithm or set its hyperparameters accurately. Likewise, expert knowledge is useful in feature engineering or hand-crafting features, which are amenable to the learning process. In all this, there is no notion of domain knowledge being used explicitly in the machine learning process, neither is there any transfer of knowledge across subsequent models that may be built.

3.2.2 Transfer Learning

Informally, transfer learning can be defined as the process of extracting information from a source domain and task and using this information to improve in a target domain and task [50]. Both Pan *et al.* [50] and Weiss *et al.* [97] formally define transfer learning as the process of improving the target predictive function $f_T(\cdot)$, given target domain $\mathcal{D}_T$ and target task $\mathcal{T}_T$, by using the information of a given source domain $\mathcal{D}_S$ with corresponding source task $\mathcal{T}_S$.

Fig. 3.1 shows how transfer learning differs from conventional machine learning. In contrast to a single learning pipeline, with a well-defined dataset, there is no knowledge flow from a source domain/task to a target domain/task. This allows for two learning pipelines in the overall problem. For example, in *type 1* first a model is learned using available historic data. Thereafter, this model can be used in the target domain. Furthermore, apart from domain knowledge in

the learning pipeline, there thus arises the possibility to add domain knowledge about the learning problem itself, this is identified as *type 2* in Fig. 3.1. A similar approach is used in *type 3*. However, here, a domain expert constructs a physics-based model that can be used to guide the learner in the target domain. *Type 4* considers these transfer learning applications that extract a feature representation from the source to initialise the target feature representation.

Fig. 3.2 illustrates how the initial and asymptotic performance of conventional machine learning models can be improved with transfer learning algorithms, which incorporate domain knowledge or human expertise into machine learning models, both in the supervised and reinforcement learning contexts. It should be noted that neither initial nor asymptotic performance increase is guaranteed.

Based on the above definition, and the applications of transfer learning, Pan *et al.* [50] subdivide transfer learning in three categories; inductive, transductive and unsupervised transfer learning. These categories can be further subdivided based on the ML setting they are applied in. DR control applications mainly benefit from ML in the supervised or RL setting. However, Pan *et al.* [50] explicitly mention that they do not consider transfer in reinforcement learning. Therefore, and because RL differs considerably from both supervised learning and unsupervised learning [82], we believe transfer in reinforcement learning and supervised learning should be dealt with separately. To date, the unsupervised transfer learning setting has only seen limited applications within the smart grid setting, and especially the DR setting. Consequently, a thorough discussion of this transfer learning setting has been considered outside the scope of this review.

Next Subsection 3.2.3 begins with a description of the different methods to evaluate transfer learning and the differences with evaluating other ML approaches. Thereafter, Subsection 3.2.4 describes the different categories of transfer learning: transductive and inductive transfer learning. This is important to understand how and when what type of transfer learning can be applied in practice. Throughout this discussion the DR control use cases have always been kept in mind.

3.2.3 Evaluating Transfer Learning

Reinforcement learning agents are evaluated on how much reward they accrue over time. Likewise, supervised learning algorithms are evaluated on a predefined loss function. When these supervised learning algorithms are applied online, *i.e.*, the model parameters are updated over time with newly observed data, the evolution of this loss function over time is also an important metric to consider. Therefore, in practice, it is not sufficient to consider how a supervised

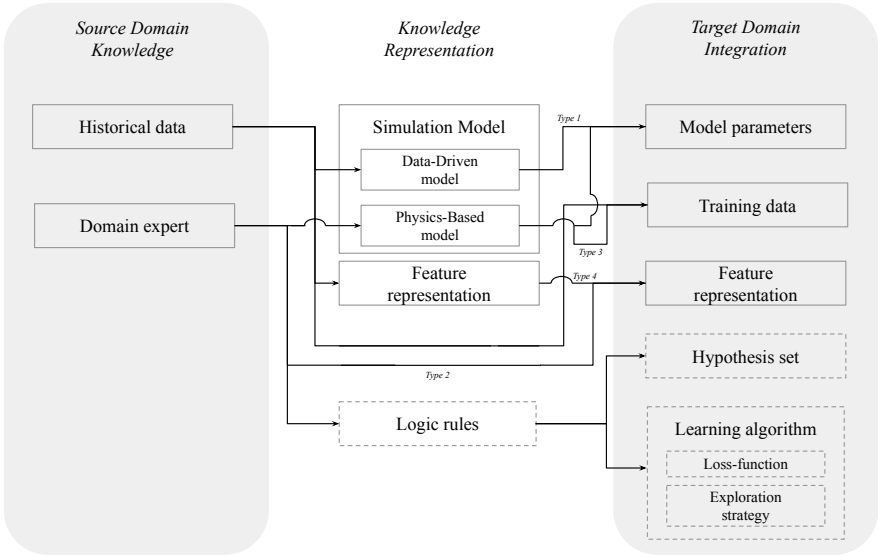


Figure 3.1: Knowledge representation and integration in informed machine learning. The knowledge is transferred from left to right and 4 archetypes of transfer learning have been identified, which have been explained in Section 2. Full lines: research in demand response exist. Dashed lines: open research questions.

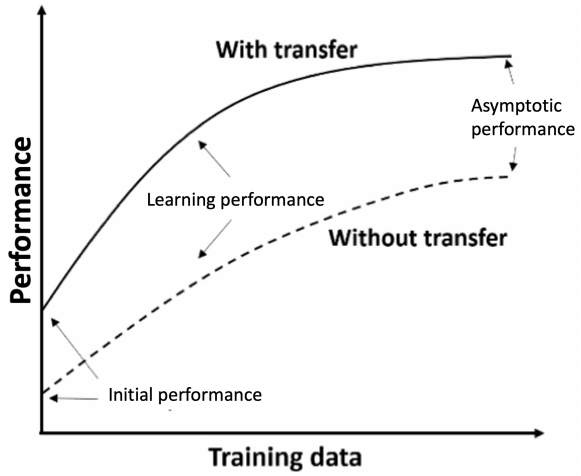


Figure 3.2: Rationale for transfer learning in supervised and reinforcement contexts.

learning model or reinforcement learning agent is performing, rather it is more relevant to track its performance over time as it gains access to increasing amounts of data. These metrics can be summarized by the initial performance, learning performance and asymptotic performance, as shown in Fig. 3.2 for supervised learning. It is straightforward to extend these metrics to the case of reinforcement learning. It is also important to note that these three metrics are agnostic to the defined performance metric, *i.e.*, reward or loss function.

3.2.4 Categories of Transfer Learning

There are slight variations on how different authors have categorized transfer learning methods in the past. These different types of categories arise from using either the feature space or the task and domains as separators. For instance, Pan *et al.* [50] use differences in task ($\mathcal{T}$) and domain ($\mathcal{D}$) to subdivide transfer learning methods. In an RL setting, the task is defined by the reward function. In contrast to Pan *et al.*, Weiss *et al.* [97] subdivide transfer learning methods based on the feature space. In an RL setting, the feature space is defined by the state-space $\mathcal{X}$. Consequently, Weiss *et al.* [97] use two categories: homogeneous and heterogeneous transfer learning. In homogeneous transfer learning, the source and target feature spaces are the same, *i.e.* $\mathcal{X}_S = \mathcal{X}_T$. On the other hand, in heterogeneous transfer learning, source and target domains are represented in different feature spaces.

Contrary to a focus on solution methods, Pan *et al.* [50] adopt a focus on the field of transfer learning or, rather, the transfer learning problems. Naturally, this results in a subdivision based on the task, *i.e.*, the reward function, and the domain $\mathcal{D}$. Taylor *et al.* [85], whose survey of transfer learning focuses on RL, in some sense also use this division. We have adopted this as well. In transductive transfer learning the source and target task are the same. However, the domain differs. Both types of transfer learning taxonomy, and their interactions, have been visualised in Fig. 3.3. An example of transductive transfer learning within the DR setting could be optimising local PV self-consumption with a battery in the source domain and another device, such as an EWH, in the target domain. Intuitively, one can see that transfer learning can be of use in such a scenario, as the control policy will be fairly similar in source and target domain. Potentially, transfer learning could thus provide a *jump start* in the target domain.

Transductive transfer learning loans its name from transduction, or transductive learning, as introduced by Vapnik [91, 92]. They are related in the sense that in the transductive transfer learning setting, like in transductive learning, there is no interest in building a general model that can be transferred (as $\mathcal{D}_S \neq \mathcal{D}_T$) [19, 91], *i.e.*, there is no interest in a general model for transferring all future

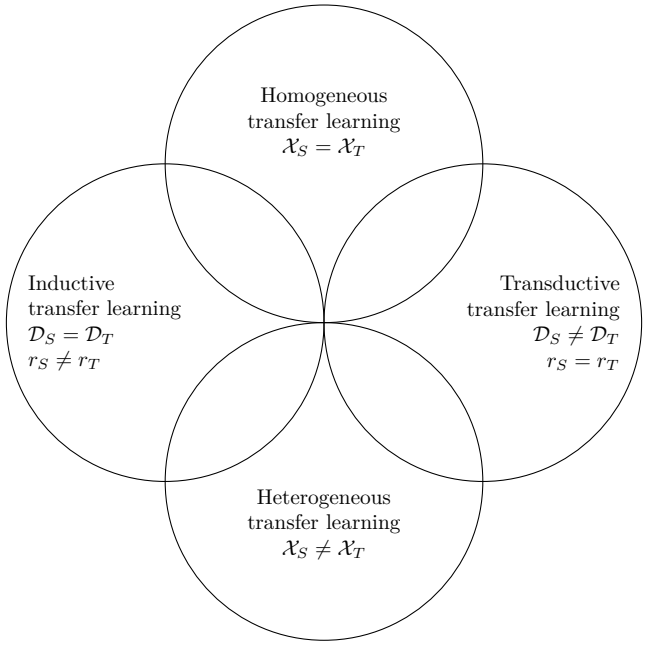


Figure 3.3: Venn diagram of different transfer learning taxonomies.

new tasks. Rather, interest lies in knowledge transfer for this specific task. Thus, while Vapnik introduced transductive learning on the level of a single data-point, here transductive learning is concerned with tasks and domains.

On the other hand, inductive transfer learning contains transfer learning problems with a different task, within the same domain. Going back to the above mentioned battery control use case, inductive transfer learning would be to switch using the battery for a self-consumption goal to an energy arbitrage goal. The dynamics of the battery stay the same in both cases, and, therefore, relevant knowledge about the control problem can be transferred between these two use cases. A more detailed treatment of how different transfer learning techniques are applied in practice is deferred to Section 3.5.

3.3 Transfer in Reinforcement Learning

There does not exist one single method of utilising transfer learning for RL, and different parts of a typical RL pipeline can benefit from it. This section first summarizes the earlier introduction to RL and the terminology used. Thereafter,

the different possibilities of transfer learning within RL are explored, with a focus on DR applications. Fig. 3.4 summarizes the different categories of utilizing transfer learning within RL.

Transfer in Reinforcement Learning	
Transductive Transfer	Inductive Transfer
$\mathcal{D}_S \neq \mathcal{D}_T$	$\mathcal{D}_S = \mathcal{D}_T$
$r_S = r_T$	$r_S \neq r_T$
Transfer in Model-Based Reinforcement Learning	

Figure 3.4: Summary of Section 3.3

3.3.1 Background

State-of-the-art RL algorithms introduce function approximation in one or several parts of the MDP. As a first example, consider a value iteration algorithm, such as Q-learning. In Q-learning one aims to iteratively update $Q(x, u)$ in order to find the optimal Q-function, and as a result the optimal policy. With a finite state-space, Q can be represented in tabular form. However, with a large state-space, and in particular with an infinite one, this becomes infeasible. Therefore, Mnih *et al.* [46] proposed to use a NN to approximate the Q-function. After Mnih *et al.* showed the benefits of using function approximation within RL, it has been widely used by researchers in other parts and/or other RL algorithms. A second part where function approximation might be useful, is in policy iteration algorithms. These algorithms aim to estimate the policy directly, rather than through the (state-action) value function, as in Q-learning. As a consequence, function approximation can be used to represent $\pi(x)$. An example of such an algorithm has been proposed by Schulman *et al.* [73]. Finally, in model-based RL the transition function itself is approximated [27]. In almost every instance of function approximation, it is possible to use transfer learning. The following subsections discuss transfer learning for each of the three parts of an RL algorithm.

Vazquez *et al.* [93] show in their survey of RL in DR, that almost all applications use some form of Q-learning. Hence, our focus on Q-learning. However, many of the algorithms subsequent to deep Q-learning [46], have been introduced in an attempt to improve upon the sample efficiency of RLs algorithms [73]. More recently, researchers have been looking at transfer learning to increase sample efficiency in the target domain [62]. Furthermore, in some critical applications, exploration, which is inherent to RL, should be avoided. In such situations, transfer learning can be used to jump-start the agent's performance [62]. While

exploration might not necessarily lead to critical errors in the DR case, users can benefit from reduced amounts of exploration in the start-phase of deployment [9].

3.3.2 Transductive Transfer: source and target domain differ

Transductive transfer learning, as defined earlier and by Pan *et al.* [50], encompasses these scenarios of transfer learning where source and target domains differ, but source and target task coincide. In RL, transductive transfer learning thus refers to these learning problems where source and target environment, and thus transition function f, are different. But, the reward function remains the same after transfer.

The idea of transferring RL agents between environments became widespread in the ML research community when openAI launched its retro contest in 2018 [24]. This contest aimed to accelerate (transductive) transfer learning by providing researchers with benchmark computer games. Algorithms submitted were tested on a new set of (unseen) levels in their respective games. Good performing RL agents should thus be able to generalise to unseen levels of the same game, *i.e.*, the domain is different while the reward is the same.

In a smart grid, and more specifically in a DR context, there is vast potential for transductive transfer learning. Different parts of the environment can alter the underlying transition function, while the general principles of the problem can remain the same. In DR programs there are three knowledge sources that can be used for transfer: (1) (other) real world data, (2) domain knowledge - including simulations, and (3) shared domain features. All three call for different transfer strategies, which we discuss next.

1. **Real world data.** Transfer from real world data includes transfer from earlier work, but also transfer from other related data sources with the same reward-function. As a first example, consider the work of Paridari *et al.* [51] and Mbuwir *et al.* [42]. In their work, they aim to design a plug-and-play Home Energy Management System (HEMS) for a PV-battery system. However, they realise user behaviour can result in differences in the transition function and, therefore, optimal policy. To mitigate this challenge, they cluster different households and use transfer learning between households that end up in the same cluster. A new household might lack enough historical data to design a tailor made HEMS, but by using limited data it is possible to pick the right cluster and use that cluster's policy as an initial starting point.

2. **Domain knowledge.** DR potential arises when there is a form of energy flexibility available, such as a battery in the previous example. TCLs provide another source of energy storage and their potential for DR has been proven numerous times in recent literature [70, 31, 44, 55, 54, 59, 56, 60, 78]. Although RL does not strictly need a model of the control environment, it certainly can benefit from one. Domain knowledge, for example in the form of a dynamic model of the environment, has been used to mitigate low data efficiency of certain RL algorithms. Lampe *et al.* [35] introduced Model-Assisted Fitted Q-Iteration (MAFQI). MAFQI is a variation of the Q-learning algorithm, in which virtual trajectories, originating from a learned environment model, are added to the RL agent's training set used to update the Q-function. Their results show an improved data efficiency, compared to regular Q-learning. Costanzo *et al.* [9] show these results can also be obtained in a DR application. Consequently, Patyn *et al.* [53] have expanded this idea to obtain informed FQI. In their approach, FQI is provided with domain knowledge through the use of models constructed by domain experts.

   In the previous examples, models of the environment have been used to provide the RL agent with an increased amount of state-transitions in the start phase. Further research, mainly in the domain of robotics, has shown simulations can also be used to explicitly initialise the policy $\pi(x)$ [62]. Peng *et al.* [62] observe that there will always be discrepancies between source, here simulated, and target domain. With domain randomization, these discrepancies are modelled as variability in the source domain. We have shown that domain randomization successfully provides an RL agent with a jump-start [56].

3. **Shared domain features.** A third opportunity for transfer learning lies in the nature of energy flexibility options. This is because energy flexibility can be provided based on different technologies, yet the main principles largely stay the same. In a lot of applications, energy flexibility is provided by some form of energy storage, *e.g.*, heat or chemical storage. While it is clear that the transition function of these types of storage is different, at a high enough abstraction layer their functionality remains the same. It remains to be seen if, for example, RL agents trained on battery storage applications have policies that are general enough to be used as initial policies in an application with TCLs. But, when differences between domains are minor, for example different rooms in the same building, sharing features can be a successful approach. Kim *et al.* [32] demonstrate this in their multi-task learning setting, by sharing features (and transitions) between control policies for different rooms in a building.

3.3.3 Inductive Transfer: source and target task differ

Recall that, by definition [50], inductive transfer learning is the case where the domains are the same, but tasks differ. This is visually presented in Fig. 3.3. In a RL scenario, this means both tasks share transition function f, but have a different reward function r. Note however that, although the transition function is the same, the conditional probability distribution $P(x_{t+1}, r_{t+1}|x_t, u_t)$ in the two domains can differ, as these probabilities are policy dependent (and the policy depends on the reward function). Since domains are the same, methods used for inductive transfer learning often correlate with those used for multi-task learning [50].

In a supervised learning setting this implies labels have to be available in both the source and target domains. Or, it needs to be possible to induce them [50]. In an RL scenario, the reward function should be available in both domains. The different approaches for inductive transfer in RL can be divided in similar categories as the above mentioned transductive learning scenarios.

A first intuitive approach is to transfer knowledge of instances, *i.e.*, use source domain instances to accelerate and jump-start target domain performance [50]. As domains are the same, it is not strictly necessary to use schemes such as domain randomization to account for domain difference. One approach is to weigh source and target domain samples differently in order to prioritise target domain experience [86]. The energy arbitrage application, as presented by Ruelens *et al.* [70], can be considered an example of inductive transfer with RL. Day-ahead electricity prices are changing from day to day and, thus, rewards of previously seen state transitions are not representative for future rewards. Even if the exact same transition would occur in the future, the received reward (or cost) would (likely) be different, since the electricity price will be different. Therefore, Ruelens *et al.* recalculate all rewards for the new prices occurring the next day. They thus use samples of the past (in the same domain) to train for a new task (new day-ahead electricity prices).

Furthermore, knowledge can be transferred using feature representations [50]. If the possible tasks, *i.e.*, reward functions, are known in advance, the RL agent can be trained on the different tasks together [10]. When using a NN for function approximation, this allows to share learned feature representations.

In a similar fashion, knowledge can be transferred using the knowledge incorporated in the parameters of the RL agent [50], be it the hyperparameters or the parameters of the regressor used to represent the policy or value-function. In this type of setting, model-based RL comes to mind. As the domain is the same, but the tasks differ, it is possible to learn an approximate model of the environment, which can then be used in an optimization setting for multiple

objective functions.

3.3.4 Transfer Learning for Model-Based RL Algorithms

It is clear RL plays a vital role in recent DR control applications. The main benefit of model-free RL is the lack for the need of an environment model, and, therefore, domain knowledge. In all examples presented until now, this was achieved by directly learning a (state-action) value-function or a control policy. It is, however, also possible to eliminate the need for a domain expert by *learning the model* of the environment, using the transitions the agent experienced. With model-based RL, there is still no need for a domain expert, as the model is learned using a data-driven approach. The dynamic model of the environment is learned (mostly) online, while control is active. This predictive model can then be used to estimate the cost or reward of a certain action, when in a certain state [27].

While model-based RL has proven to be relatively sample-efficient, compared to model-free RL, there is still room for improvement [27, 84]. Similar to model-free RL, model-based RL can use transitions of a source domain to jump-start control performance in the target domain [84]. Taylor *et al.* [84] developed a model-based transfer learning method where source domain transitions are transformed to fit the target domain and task. This transformed source-data can then be used to build an initial model in the target domain.

Recent literature has shown that transfer learning can be used to mitigate the lack of sensing in EWHs [31]. With transfer learning Kazmi *et al.* accomplish few-shot learning, both with homogeneous and heterogeneous appliances. It thus enables all benefits of black-box modeling, while limiting the need for extensive sensing. It is exactly in this regard that the aim of transfer learning for model-based RL differs from general supervised learning. In model-based RL, the model is needed for control of a certain environment, and one mostly operates in a few-shot learning setting. Sample efficiency is therefore very important.

3.4 Application Scenarios

Transfer learning for RL in the context of DR is still in an early stage of development. Nevertheless, a number of case studies have appeared in the recent past that use it to improve models or control policies.

One of the very first applications of transfer in RL was for forecasting energy demand of buildings with limited historical data [47]. The authors transferred forecasting models trained using RL algorithms - SARSA and Q-learning with deep belief networks for function approximation - and data from source buildings to predict the energy demand in (commercial and residential) buildings with unlabelled historical data. Similarly, Kong *et al.* [34] transferred knowledge on the user's electricity price elasticity from regions where DR has been implemented to areas with unknown elasticity. This elasticity is used to estimate the electricity demand of the users in the new region, which is then used to train an RL algorithm - SARSA - that selects suitable retail electricity prices to enforce DR in this new region.

A parallel thread of research has focused on using transfer learning to improve reinforcement learning based control strategies of flexible assets. An early example of using prior knowledge in an RL system using a hybrid simulation learning control can be found in [39], where the authors adopt a two-step approach. In the first step, they pre-train the controller using a calibrated model of the HVAC system under consideration. Then, this controller is updated online during the operational phase in an experimental environment. Likewise, in the works of Costanzo *et al.* [9] and Ruelens *et al.* [70] the authors used transfer learning in the form of expert knowledge to shape and enforce monotonicity in a control policy previously learned from a limited number of observations.

The authors of [53] proposed using expert knowledge in the form of grey-box model predictive control transitions for kick-starting of an informed fitted Q-iteration-based controller. The authors used a linear grey-box model predictive control approach as the expert and showed an increase in cost savings. Peirelinck *et al.* [56] used transfer learning through domain randomization to facilitate knowledge transfer and reduce the exploratory time of the learning agent. The authors reported an 8.8% increase in cost savings compared to the setting without any knowledge transfer. This approach has the added advantage of not requiring a well-calibrated simulation model of the system under consideration. Likewise, domain randomization in RL was also used by Kazmi *et al.* [29] to control batteries in order to solve voltage problems in the low voltage grid. Control agents were trained offline using randomly sampled load and PV generation profiles in many different simulated topologies of the distribution grid. The authors showed that by employing domain randomization more grid violations were resolved compared to the case without.

In the work of Mbuwir *et al.* [42], the authors used cluster-based transfer learning to transfer knowledge in the form of control policies amongst buildings with similar energy usage patterns. A control policy learned using data from a data-rich building in a cluster is used to initialise learning in a target building belonging to the same cluster. The authors showed a faster convergence to a

near optimal policy compared to when no knowledge was transferred. Similarly, Paridari *et al.* [51] proposed a plug-and-play planning and control framework for control energy storage devices in buildings with PV installations. In their work, knowledge was transferred in the form of a policy function approximation to new end users with no historical data – from which a control policy could be learned – leading to a 29% increase in cost savings. Likewise, in [32] the authors exploit the structural similarities in the control policies across rooms in a building by sharing features (and transitions) to obtain a policy that can set the suitable energy levels for lighting and air-conditioning units. To avoid excessive use of cloud resources and speed up the training process when training RL-based control agents from scratch in DR applications, Tao *et al.* [83] transferred weights of the control policies between batteries and Heating, Ventilation and Air Conditioning (HVAC) units. The authors showed: a significant cost savings in a homogeneous setting (knowledge transferred between two battery control agents), and a slight cost savings in a heterogeneous setting (knowledge transferred from a battery to an HVAC control agent) compared to training an entirely new policy. Moreover, the authors showed that knowledge can be transferred between different DR programs (price-based to direct load control).

Even though the above applications have shown the positive impact of transfer learning on the target domain learner, the effectiveness of the knowledge transfer is not always guaranteed. Knowledge transfer can also lead to reduced performance in the target domain/task. This is termed negative transfer and can occur due to source and target tasks being unrelated or the domain data distributions being too different. For example, a 1.36% reduction in prediction accuracy was reported in [98] due to negative transfer. Several approaches exists in the literature for mitigating negative transfer as summarised in [99]. In the context of demand response, negative transfer has been tackled by selecting appropriate source tasks using a Gaussian process-based selection algorithm [100] or using the TrAdaBoost algorithm [98], which decreases the weights of source instances with distributions different from that of the target. Establishing the similarity between source and target tasks, and performing optimization to determine the appropriate number of source tasks – a large number of source task could increase the computational time while too few tasks might not provide sufficient supplementary information – also mitigates negative knowledge transfer [98].

3.5 Future Directions

The presented review shows that transfer learning has gained considerable traction in recent years, and is currently the subject of intensifying research

efforts in DR. This attention can mainly be attributed to the success of transfer learning methods in other domains, and early indications that the methods hold enormous potential for DR applications as well. The review also shows that major research opportunities remain untapped, with most research to date being concentrated on improving demand forecasts using transfer learning. This section explores and elaborates the identified limits and resulting research opportunities of transfer learning within a DR setting. Table 3.1 summarises all applications that have been reviewed in the preceding section, and provides an overview of the research gaps that remain within transfer learning applications in DR.

A first observation made based on Table 3.1 is that most of the reinforcement learning based transfer learning applications within the DR setting focus on utilising domain knowledge to increase control performance. An example of this is controllers trained using simulators.

Furthermore, from Table 3.1, it is clear that in every area of transfer learning within DR there remain research opportunities. But, especially inductive transfer learning and transductive learning with feature sharing between different domains remain open challenges. Although inductive transfer within reinforcement learning has seen few research applications, it could be a promising field in the near future. For example, one can think of a DR setting where a flexibility provider offers different services to the market, *e.g.*, frequency response and peak shaving. At different times, this provider would then have to switch between reward-functions, transferring as much relevant information about the underlying system dynamics as possible. Likewise, transductive learning with feature sharing has relevant applications in practice but has seen relatively little research interest. For example, energy storage devices can be based on multiple technologies. However, at a high enough abstraction level they have similar dynamics. Therefore, features could be shared among different storage appliances. Transfer could then be used to jump-start performance of a new storage technology entering a flexibility provider's set of appliances. This is something we will explore in chapter 5. It could also be used to increase overall performance of the flexibility pool among all active appliances.

Finally, it is interesting to revisit Fig. 3.1. It is now clear that almost all applications considered represent the knowledge of the source domain in the form of data. Either from *Simulation Results* or from *Historic Data*. Yet, other representations of knowledge are possible. Simultaneously, most knowledge of the source domain and task is integrated in the target domain and task by incorporating it in the agent's training set. Taking the broader field of informed ML into account, it is also possible to integrate knowledge through other means. For example, one could restrict the hypothesis set by a pre-defined model structure of the NN, or by modifying the loss function of the model.

Table 3.1: Classification of transfer learning applications in demand response.

	Reinforcement Learning
Transductive transfer	
Domain knowledge	[35, 9, 53, 56]
Real-world data	[51, 42, 47]
Shared features	[32], [34]
Inductive transfer	[70, 69]

As far as the reinforcement learning algorithm goes, our review clearly shows Q-learning is widespread and well-used in DR applications. Many researchers seem to agree that Q-learning is promising, as it has shown good results in past research and in other domains. However, in other domains, such as robotics, state-of-the-art policy iteration algorithms have proven to handle challenging tasks better [73]. Therefore, it is important to follow-up on the advances that have been made elsewhere and investigate their applicability to DR, especially in the context of transfer learning. This might be necessary for such algorithms to become sophisticated enough to handle real-world challenges, which arise as much from sparse data as they do from other limitations such as poor quality data, and ill-defined and often conflicting objectives. In Chapter 6 we will take a look combining PPO, the current state-of-the-art RL algorithm, with transfer learning in a DR context.

3.6 Conclusion

The recent adoption of machine learning-based techniques in demand response applications has been influenced by the availability of data through smart metering and the smart grid in general. However, using these techniques in newly constructed systems remains challenging due to their lack of (sufficient) historical data. Transfer learning has the potential to solve this challenge and improve generalizability of machine-learning based models and control policies as seen in this review. This is evident in the performance gains we have observed in the papers reviewed herein. In many cases, this can be the difference between machine learning models that can learn from the available limited data vs. those that fail to converge to an accurate solution. However, to date, a majority of articles has focused on transfer learning for forecasting energy demand, with only limited attention paid to renewable energy generation and electricity price forecasting. A few, but increasing number of, articles on modelling system dynamics and control of electric water heaters and batteries for energy storage have also appeared in the very recent past.

In transfer learning literature, a wide variety of methods – ranging from data and parameter sharing to learned representations have been explored. Although a plurality of the articles reviewed in this paper have considered knowledge transfer in the form of pre-trained weights from the source model to initialise learning in the target, a few articles have looked into sharing feature representations. In the context of reinforcement learning, knowledge transfer in the form of domain/expert knowledge has been explored with the literature showing how domain knowledge can accelerate learning of adequate control policies.

A critical shortcoming we have identified is that almost half the articles reviewed do not provide any quantification of gains attributed to transfer learning. Furthermore, even when such numbers are included, they mostly do not provide a complete representation of the gains that could be associated with transfer learning. These include improvements to initial and asymptotic performance as well as the rate of improvement (Fig. 3.2). These concerns are further exacerbated by the fact that most studies do not open-source their data or codebase, often due to privacy concerns, and are consequently not reproducible. A recommendation for future research is therefore to quantify all of these three metrics, especially against strong baselines, and open-source the trained models in a responsible manner when sharing code and data is not possible.

While research on transfer learning for demand response applications is still in its infancy, we have found that most of the use cases have only been tested in simulation environments. Consequently, in order to prove the effectiveness of transfer learning for demand response applications, more real-world experiments need to be conducted. This is especially true due to the added challenges that can arise while deploying models that use transfer learning. Moreover, in the context of reinforcement learning in DR, we have observed only limited applications of transfer learning despite its promising potential.

Chapter 4

Domain Randomization for Demand Response

This chapter is largely based on:

- [56] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Domain Randomization for Demand Response of an Electric Water Heater". In: *IEEE Transactions on Smart Grid* 12.2 (May 2020), pp. 1370–1379. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.3024656

The first section gives an introduction to domain randomization and its application within DR. Section 4.2 recapitulates the different buffer models used throughout this chapter, presents the problem formulation and, subsequently, looks at the RL algorithms used for this part of the work. Section 4.3 presents the methodology. Consequently, Section 4.4 presents the experiments and findings. Section 4.5 draws some conclusions.

4.1 Introduction

While the idea of transfer learning is intuitive and straightforward, a naive application of the concept might not give the best results. Differences between source and target environment (or domain), for example due to modelling error, can lead to sub-optimal results or, in worst case, negative transfer. Therefore, methods need to be developed that take into account these differences. Domain

randomization is such a method. This chapter shows how domain randomization can be applied in a DR setting and investigates if it is beneficial.

Recent breakthroughs in RL indicate we can avoid learning a model from observations in case we have prior knowledge about the system dynamics. Tobin *et al.* [87] showed it is possible to transfer NNs from simulation (source domain) to practice (target domain). They proposed domain randomization, here one randomizes the source domain in certain parameters. The general idea is that through this randomization the agent can simultaneously be trained on a multitude of source domains. As a result, the agent will perform better when applied in the target domain. Further research [62] has shown that domain randomization greatly reduces target domain training time and policies are capable of adapting to unfamiliar target-system dynamics. In another example, Andrychowicz *et al.* [2] have shown a robotics use case where an agent was trained to perform dexterous in-hand manipulations in simulation. In these simulations different parameters, such as friction coefficients and object appearance, were randomised. This allowed the agent to generalise to unseen (real-life) manipulations. It is unclear if these results are applicable to the DR setting.

A wide variety of EWH buffer models has recently been presented [18, 76, 90]. We aim to exploit both the modelling and RL research. Our objective is two-fold: apply RL for EWH control *and* minimise initialization time. Our main contribution consists of combining the usage of these buffer models with domain randomization, in order to reduce initialization time when applying RL in a DR setting. The aim of this work is thus to verify that the combination of buffer models, domain randomization and RL is fruitful for DR. As a first step, both our source and target domain are simulated. However, as the lab-validated target domain model is more elaborate than the source domain model(s), this results in a clear modelling difference.

To explore the suitability of domain randomization for transfer from simulation to practice in DR applications, this chapter looks into transfer from the uniform or two-layer model to the stratified EWH model. Earlier in this text the clear modelling difference between these EWH models has been shown. Therefore, we believe our results can indicate if domain randomization should be applied in DR settings.

4.2 Environment Models and Reinforcement Learning Algorithms

4.2.1 The Markov Decision Process

This chapter follows the typical MDP considered throughout this work, as presented in Section 2.1. An MDP can be defined based on its transition probabilities, its state-space, action-space and cost-function. The transition probabilities are considered to be unknown in the target domain. The other MDP properties are defined below.

State-space

The augmented observable state-space $\mathcal{X}$ incorporates SoC and time information. State $x \in \mathcal{X}$ at every time step t is given by

$$x_t = [SoC_{t-3}, SoC_{t-2}, SoC_{t-1}, SoC_t, t_{\cos}, t_{\sin}]\,. \tag{4.1}$$

Action-space

As presented earlier, a binary action-space $\mathcal{U}$ has been considered. Action $u = 1$ implies the EWH's heating element is turned on its at rated power P_r, $u = 0$ turns the heating element off. Note that the comfort controller $H(u_t)$ can overrule this action when necessary to ensure user comfort, according to (2.17).

Cost-function

An energy arbitrage DR setting with ToU pricing has been considered in this chapter. Given energy price λ_t and energy consumption $E_t = H(u_t)P_r\Delta t$, the reward at time-step t is given by

$$r_{t+1} = -\lambda_t u_{\text{phys},t} P_r \Delta t = -\lambda_t E_t. \tag{4.2}$$

Experiments have been conducted with two different price profiles, defining λ_t. These two profiles are presented together with the respective experiment and results in Section 4.4.

4.2.2 Electric Water Heater Models

All three models presented in Section 2.2.1 are used throughout this chapter. The stratified model has been validated in a lab and proven to behave in a similar fashion as a real stratified buffer [90]. Therefore, the stratified model, the model which most closely represents reality, has been considered as the target domain. The goal of the experiments is thus to show that it is faster to fine-tune a control policy in this domain than it is to learn one from scratch.

Section 2.2.1 shows there is a clear modelling difference between all three EWH models. Therefore, different experiments have been conducted both with the uniform and the two-mass model acting as source domain. This could give an indication of the complexity needed for successful transfer of control policies. For example, the uniform model completely disregards all notion of stratification. At this point, it is not clear if a policy trained using the uniform model as source domain would provide a significant benefit over directly training in the target domain

4.2.3 Reinforcement Learning Algorithms

The experiments have been conducted with both value iteration algorithms, DQL and FQI. Both NNs used in DQL consist of two hidden layers, each with 64 neurons. Each hidden layer has a $ReLu$ activation function. In contrast, random forests are used for the sake of computation speed in FQI. The iterative re-training of all Q-functions within FQI has the advantage that it inherently allows for changing price ToU-price profiles. However, its main disadvantage is the increased computational complexity, compared with DQL. Research has shown, however, that using random forests instead of NNs with FQI strikes a good balance between computation speed and performance [43].

Due to their differences each of the two algorithms requires a distinct transfer learning approach. The transfer methodology for both algorithms is explained in the next section.

4.3 Methodology

Our objectives are two-fold. We want to keep all benefits of a model-free approach when controlling an EWH in energy arbitrage DR settings, and we want to exploit the limited a priori available information to reduce learning time. The main hypothesis is that pre-training can be used for policy initialization

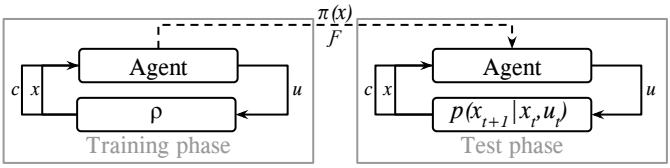


Figure 4.1: Policy $\pi(x)$ and experience replay memory $\mathcal{F}$ transfer approach. Training phase model is a distribution ρ over the source domain. The agent observes stratified model dynamics ($p\left(x_{t+1}|x_t, u_t\right)$) during test phase.

despite the presence of modelling errors, and that it reduces initialization time. In this work, we know the RL agent is going to be applied on an EWH to minimize operating cost when energy consumption is charged with a (given) ToU price. The stratified model presented in Section 2.2.1, with parameters as given in Table 2.1, represents the target domain. This implies we can only use it as a virtual test-bed. The uniform and two mass model represent the source domain, *i.e.*, simplified (or approximated) models of the target domain. Our objectives then translate to training policies that can control the EWH under the stratified model dynamics $p(x_{t+1}|x_t, u_t)$, while reducing learning time using its approximate dynamics $\hat{p}(x_{t+1}|x_t, u_t)$.

All experiments consist of a training and test phase, as illustrated in Fig. 4.1. During training, the agent samples from $\hat{p}(x_{t+1}|x_t, u_t)$, by interacting with the source domain. It saves the samples in $\mathcal{F}$ and updates $\pi(x)$ using either DQL or FQI. To account for discrepancies between source and target domain, *i.e.*, modelling differences, we include variability in the former [62]. We use domain randomization and expose the agent to a distribution ρ over environments $\hat{p}(x_{t+1}|x_t, u_t) \sim \rho$ [87]. We draw samples from this distribution once, at the start of the training phase, and train the agent through iteration over the sampled models. Tobin *et al.* [87] state that with enough variability during the training phase, the RL agent will be able to generalize to dynamics seen in the test phase. Instead of training a policy that can perform the EWH control task under one dynamics model, we train a policy that can perform the task over a variety of models.

In the remainder of this work, we assume the EWH's height h, diameter d and rated power P_r are known in the training phase and equal to the value in Table 2.1. Furthermore, the thermal transmittance of the target domain U equals $0.75\,\mathrm{W/(m^2K)}$. In the training phase, we assume a normal distribution over $U \sim \mathcal{N}(\mu, \sigma^2)$ in the source domain, with the value of μ and σ depending on the experiment. During this phase, the agent's objective is then to maximize expected return across a distribution of dynamics models ρ [62]. The agent learns

a control policy using DQL or FQI. The training phase lasts for $T_{\text{train}} = 15T$ simulation days. We assume a separate tap water profile $\dot{m}_w$ of 15 days is available for this phase. Both in the training and test phase we use an ϵ-greedy exploration policy [82]. After pre-training, the policy π and/or the replay memory $\mathcal{F}$ are transferred, as shown in Fig. 4.1. The transfer approach differs slightly for DQL and FQI.

4.3.1 DQL

The parameters θ of the two NNs are iteratively updated during training phase according to Algorithm 1. They encapsulate information about the MDP. The target domain uses the final source domain's θ and θ^- as initial values and further updates these parameters (also) according to Algorithm 1. The experience replay memory's maximum size $|\mathcal{F}|_{\max}$ is 35 days, or $|\mathcal{F}|_{\max} = 35 \cdot 96$ state-transitions. Each time step t, we perform one update of online and target NN according to (2.25) and (2.26), with a batch of size $3T = 288$, uniformly sampled from $\mathcal{F}$.

4.3.2 FQI

At each iteration the whole set of random forests is refitted, rather than updated. Therefore, we only transfer the replay memory $\mathcal{F}$ from source domain to target domain. This is an extension of the MAFQI [35, 9] idea. While Riedmiller and Lampe [35] propose to train a NN to approximate the state-transition function and add virtual tuples to $\mathcal{F}$, we use a physics-based approximate model with domain randomization to (initially) populate $\mathcal{F}$. Again, $|\mathcal{F}|_{\max} = 35T$. This is sufficiently large, as Costanzo *et al.* [9] show that FQI converges after approximately 20 days. As a result of this limitation, source domain state-transitions start to be removed from $\mathcal{F}$ after 20 simulated test days ($|\mathcal{F}|_{\max} - T_{\text{train}} = 20T$). Both in the source and target domain, Algorithm 2 is used.

4.4 Experiments and Results

The experiments aim to verify our main hypothesis: is it faster to fine-tune a DR controller learned on an approximate model rather than learn one from scratch? All simulation set-ups and their results are presented next. The same experiment has been run 20 times, in order to account for the variability present in the training and test phase. We drew new samples $U \sim \mathcal{N}(\mu, \sigma^2)$ each of

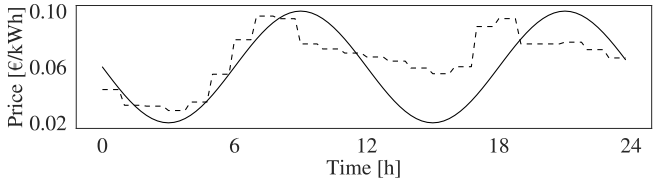


Figure 4.2: Sinusoidal price profile (full line) and a one day example of the Belpex price profile (dashed line).

the 20 times. The two ToU price profiles divide the experiments into two main categories. First, we present the experiments which use a sinusoidal price profile. Thereafter, we display the ones using the Belgian day-ahead wholesale electricity prices (of 2018) [16].

4.4.1 Sinusoidal price profile

Fig. 4.2 shows the sinusoidal price profile. We have based this tariff both on the Belgian day-ahead prices and on the ToU price profile which has been proposed to one of the Belgian energy regulators (VREG) [11]. Prices vary throughout the day, but the same profile occurs every day.

Double Q-Learning: direct vs. pre-training

In a first experiment we have applied DQL directly in the target domain, and compare it with the pre-training approach as presented in Section 4.3.1. We employed the two mass model, presented in Section 2.2.1, in the source domain, as this resembles the target domain (stratified model) most closely. Each simulation run we sampled five two mass models, with $U \sim \mathcal{N}(0.75, 0.1^2)$.

Fig. 4.3 shows the cumulative cost of three control approaches over 35 days. The most expensive control approach, with a final cost of € 23.9, is the hysteresis controller. This controller acts according to $H(u)$ (2.17), with $u = 0$. The dotted and dashed line show the cumulative cost of the direct and pre-training approach. The bars depict the 95 % confidence bound (over 20 simulation runs). As known from previous research [69, 54, 61], RL (without pre-training) manages to reduce cost compared to a hysteresis controller. Here, cost has been reduced by 23.4 %. With DQL and given this price profile, pre-training reduces cost 8.8 % further, and has a total cost reduction of 32.2 % compared to the hysteresis controller.

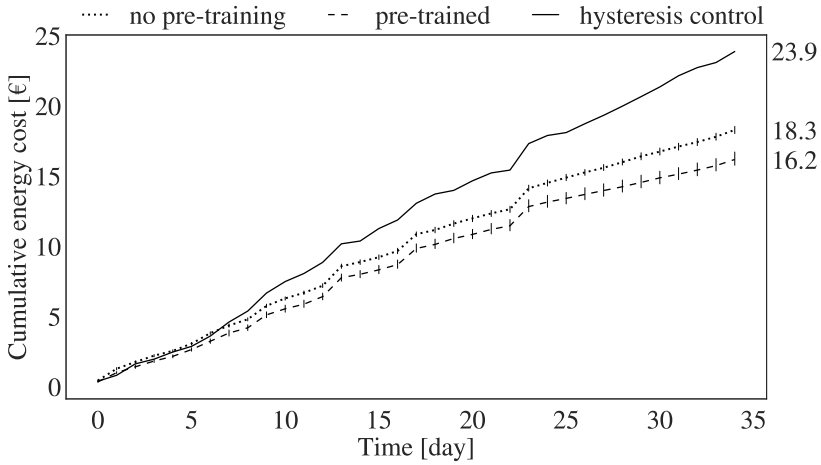


Figure 4.3: Cumulative cost comparison: DQL-sin experiment. Mean of 20 simulations, vertical bars indicate 95 % confidence interval.

Table 4.1: Total cost and p-values for intervals of 7 days: DQL-sin experiment.

Days	[0,6]	[7,13]	[14,20]	[21,27]	[28,34]	**[0,34]**
p-value	$2e^{-6}$	0.029	0.007	$2.1e^{-5}$	$4.9e^{-4}$	$\mathbf{3e^{-9}}$

Fig. 4.4 compares performance of both approaches. It separates the simulation days in 5 intervals of 7 days and shows the mean total cost and standard deviation (of 20 simulation runs). Table 4.1 shows the t-test's p-value for the mean of these costs, *i.e.*, the probability that these two distributions have an identical average. The table shows total cost of all intervals differs significantly for this experiment. Additionally, the whole simulation run's total cost differs significantly, as the p-value is close to zero. While cost between days within one approach can vary because of tap water demand, the same days have the same tap demand across all approaches.

Fig. 4.5 shows the policy in the target domain for both DQL approaches. The top row visualizes the policy learned from scratch, after 5 and 35 days. The bottom row shows the same after pre-training. To create this figure, snapshots of the target domain policy NN were saved during simulation at the given time steps. Afterwards, random state-space samples were fed through these snapshots. As is clear from (6.7), each state x consists of four SoC values. Thus, given a certain value for SoC_t, the policy outcome can be either $u = \pi(x) = 0$ or $u = \pi(x) = 1$, depending on the other three SoC values. Fig. 4.5 is a two-dimensional representation of the policy, *i.e.* states with the same value

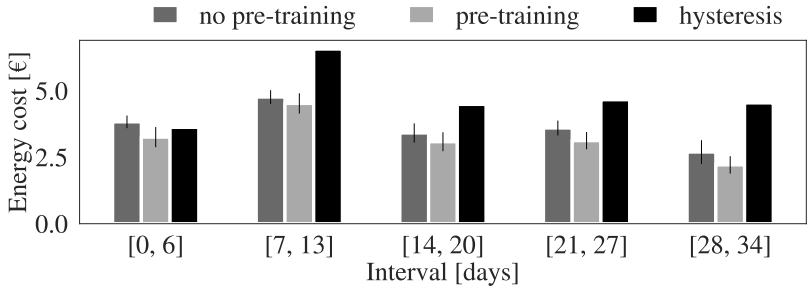


Figure 4.4: Cost comparison of intervals of 7 days: DQL-sin experiment.

for SoC_t and t are depicted on the same point in this two-dimensional space, although they might have different values for SoC_{t-1}, SoC_{t-2}, SoC_{t-3}. Black indicates the EWH is turned on ($\pi(x) = 1$) for that particular time-step t and SoC_t, for all state-space samples. Lighter shades indicate energy consumption only for certain samples with the same SoC_t and t (but possibly different values for SoC_{t-1}, SoC_{t-2}, SoC_{t-3}).

The top left figure shows that only a limited amount of the state-space has been explored in the direct approach, and the policy is initialized only in this part. After pre-training, the policy is already initialized over a larger part of $\mathcal{X}$. After 5 days, the pre-trained agent has learned a basic policy, turning the EWH on when prices are low. After 35 days, the RL agent that started from scratch has also learned this behaviour. Additionally, when SoC turns out to be low before the highest peak, the agent turns the EWH on. We see the policy learned after 35 days and with pre-training is similar, but the big charging cycle has moved to the afternoon price drop. This results in a cheaper policy, as most of the DHW consumption occurs in the evening [15]. In the morning it is sufficient to charge to $\pm 50\,\%$.

Fitted Q-Iteration: direct vs. pre-training

The same figures are shown, now for the FQI approach. Fig. 4.6 shows the cumulative cost of the compared control approaches. Again, the hysteresis controller is the most expensive, while the pre-training approach is the cheapest. Table 4.2 shows that the difference between pre-training or not is significant for all but the last interval. Which is to be expected, since by then most pre-training samples have disappeared from the experience replay memory $\mathcal{F}$. Although learning from scratch already reduces total cost with $27.6\,\%$, compared to hysteresis control, pre-training reduces cost with $35.1\,\%$. Fig. 4.7 compares

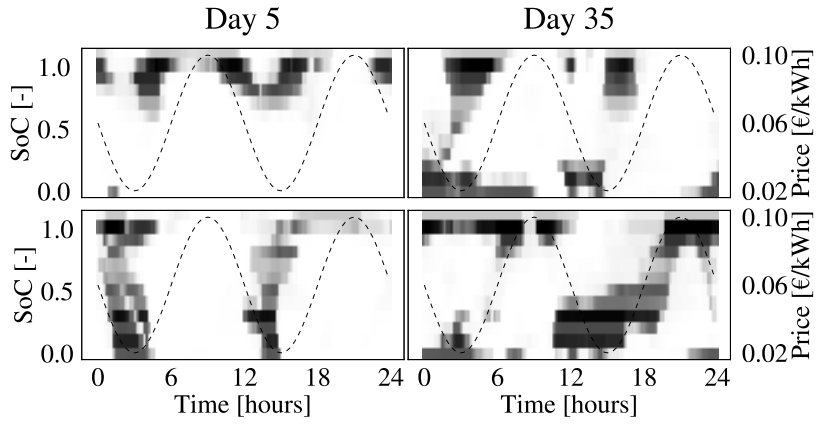


Figure 4.5: Policy comparison: DQL-sin experiment. Top row: policy without transfer learning, bottom row: policy with transfer learning. Left column: snapshot of policy after 5 days, right column: snapshot of policy after 35 days. Back indicates $u_t = \pi(x_t) = 1$

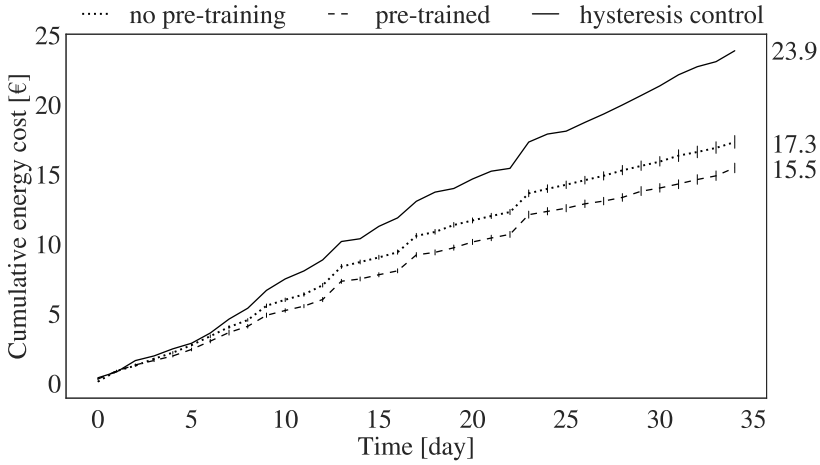


Figure 4.6: Cumulative cost comparison FQI-sin experiment. Mean of 20 simulations, vertical bars indicate 95% confidence interval.

all three controllers in this experiment. It shows both FQI-based RL approaches are already cheaper than the hysteresis controller in the first interval. The figure also shows pre-training results in additional cost gains, starting from the first interval.

Table 4.2: Total cost and p-values for intervals of 7 days: FQI-sin experiment.

Days	[0,6]	[7,13]	[14,20]	[21,27]	[28,34]	**[0,34]**
p-value	$3e^{-4}$	$6.3e^{-8}$	$2.5e^{-5}$	0.04	0.88	$\mathbf{4e^{-7}}$

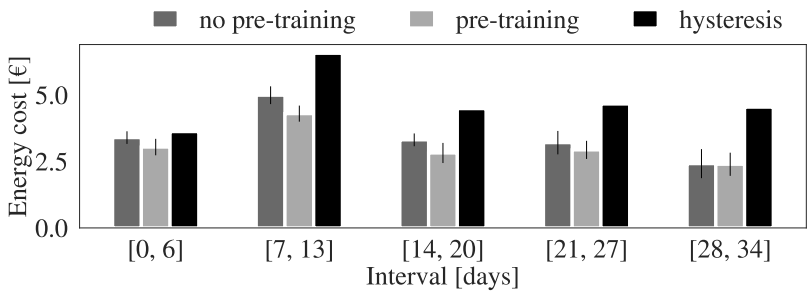


Figure 4.7: Cost comparison of intervals of 7 days: FQI-sin experiment.

Comparison with Model Predictive Control

Of the three presented buffer models, the uniform model is the only one that is linear. For this case, the source domain control problem can thus be formulated as a Mixed Integer Linear Problem (MILP), given in (4.3). Therefore, it is possible to use MPC. This experiment compares MPC with both transfer learning approaches, when using the uniform model. With MPC, the target domain's (observable) state is sampled at time step t. Here, the target domain (stratified buffer) backup sensor temperature at that time step ($T_{b,t}$) is used as initial state (for the temperature T_L) in the source domain (uniform model), as shown in (4.3e), and $U = 0.75\,\mathrm{W/(m^2K)}$ in both source and target domain. The MILP solution is a vector of length T with the source domain optimal control actions for period $[t, t+T]$. After applying action u_t in the target domain, the procedure is repeated, and this for every time step t. For a more elaborate explanation of MPC we refer to other literature [65]. We solve the MILP using the Gurobi solver [21].

$$\min_{u_k} \quad \sum_{k=t}^{t+T} \lambda_k u_k P_r \Delta t \tag{4.3a}$$

$$\begin{aligned} \text{subject to} \quad T_{L_k} &= T_{L_{k-1}} + \Delta t \frac{1}{C_{\text{boiler}}} [u_k P_r \\ &+ \dot{m}_w C_{pw} (T_{iw} - T_{L_{k-1}}) \qquad (4.3b) \\ &+ U(A_s + 2A_t)(T_{\text{amb}} - T_{L_{k-1}})] \end{aligned}$$

$$T_{L_k} \geq T_{\min} \tag{4.3c}$$

$$T_{L_k} \leq T_{\max} \tag{4.3d}$$

$$T_{L_0} = T_{b,t} \tag{4.3e}$$

Fig. 4.8 shows the cumulative cost of all 4 controllers (MPC, pre-trained FQI, pre-trained DQL and hysteresis control). Pre-trained DQL final cost is €15.6, and thus 26 % cheaper than MPC, with a final cost of €21.2. Pre-training FQI with the uniform model results in a total cost of €16.1. It is thus clear it does matter to capture the non-linearities of the stratified model, which a RL agent is able to do. Although RL manages to capture relevant information from the uniform model, the model does not seem to be sufficiently accurate for MPC. Furthermore, this figure also shows there is only a small performance difference between FQI and DQL.

U out of distribution

In a last experiment with this price profile, we investigated the effect of a bad distribution over U, both for DQL and FQI. Apart from sampling U from $\mathcal{N}(0.55, 0.1^2)$ the procedure is equal to the first and second experiment. The target domain thermal transmittance U is now 2σ away from the source domain's mean value.

Table 4.3 shows the results. For both algorithms, the performance with a *good* guess for the distribution over U (target domain value equal to source domain mean) and with a *bad* guess is very similar. This indicates that pre-trained policies are indeed able to generalize over different buffer model parameters, and that we can use RL to further fine-tune policies in the target domain.

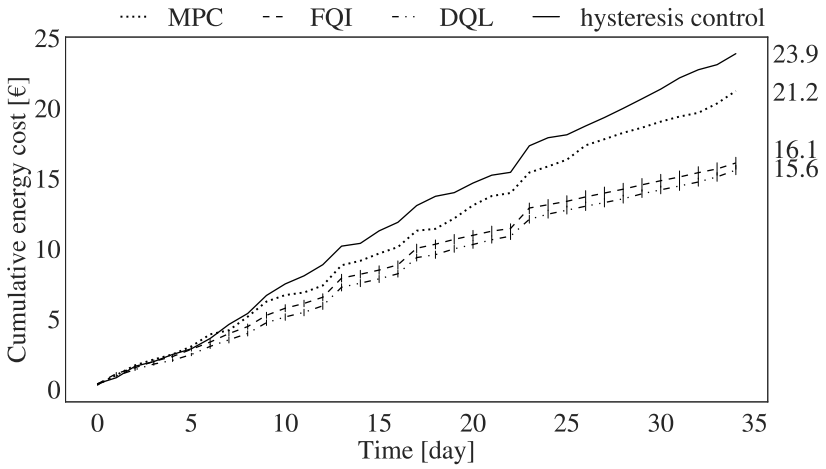


Figure 4.8: Cumulative cost comparison: FQI and DQL vs. MPC experiment (uniform model). Mean of 20 simulations, vertical bars indicate 95 % confidence interval.

Table 4.3: Results for different distribution parameters over U.

U		$\mathcal{N}(0.75, 0.1^2)$	$\mathcal{N}(0.55, 0.1^2)$
FQI [€]	μ	15.49	15.25
	σ	0.79	0.97
DQL [€]	μ	16.20	16.08
	σ	1.04	1.02

4.4.2 Belpex price

In a final experiment, we have used the Belgian day-ahead electricity prices [16] (dashed line in Fig. 4.2), and compared performance difference between the use of pre-training or not. Belpex is an hourly-varying price profile which differs every day. *E.g.*, Fig. 4.2 shows the first day of the test-set prices. FQI separates the SDP in T supervised learning problems and refits the regressor for every control step t. Ruelens *et al.* [69] propose to exploit this property when dealing with a time-varying ToU price profile, which is different every day, by recalculating all rewards in $\mathcal{F}$ at every step. As DQL is an online learning algorithm, adapting it in a similar way is non-trivial, as information of previous rewards is encapsulated in the NN's weights. Another possibility, as explored by Cao *et. al.* [7], is to add future prices to the state. Thus, for the FQI-based controller no additional adaptations are needed to the state, but every training

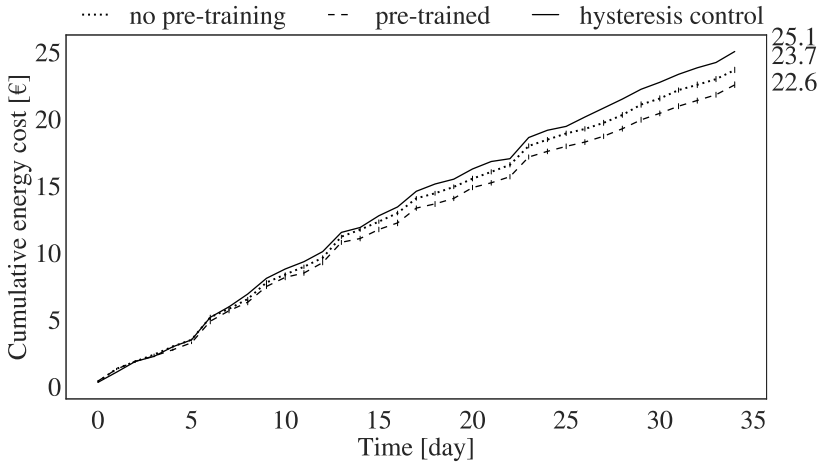


Figure 4.9: Cumulative cost comparison: FQI-Belpex experiment. Mean of 20 simulations, vertical bars indicate 95 % confidence interval.

day the rewards in $\mathcal{F}$ need to be recalculated. For the DQL-based controller on the other hand, this recalculation step is unnecessary, but the observed state at time step t is now given by (4.4).

$$x_t = [SoC_{t-3}, \ldots, SoC_t, t_{\cos}, t_{\sin}, \lambda_t, \ldots, \lambda_{t+24}] \tag{4.4}$$

First, all FQI results are presented. Fig. 4.9 shows the cumulative cost over the simulation period for all considered FQI controllers. While their order stays constant, the difference between them reduces. With this price profile, direct RL is 5.5 % cheaper than the hysteresis controller and the transfer learning approach is another 4.4 % cheaper. Compared to the same experiment with the sinusoidal price profile, the cost gains are smaller. This can be explained by the smaller valleys and peaks in the price profile, as is clear from Fig. 4.10. This figure shows the target domain's final five simulation days, for the pre-trained case. The top graph depicts Belpex price on the right axis and the buffer's SoC on the left axis. The grey areas indicate if the EWH is consuming power ($u_{\text{phys}} = 1$) or not ($u_{\text{phys}} = 0$). The bottom graph depicts DHW consumption. The control policy turns the EWH mostly on when prices are relatively low. However, completely avoiding energy consumption during higher priced hours is not always possible due to hot water consumption.

Table 4.4 divides the simulation period in 5 intervals, each of 7 days. The table shows that each group's mean cost for direct FQI is larger than the group's cost with pre-trained FQI. The difference is, again, significant for all but the last

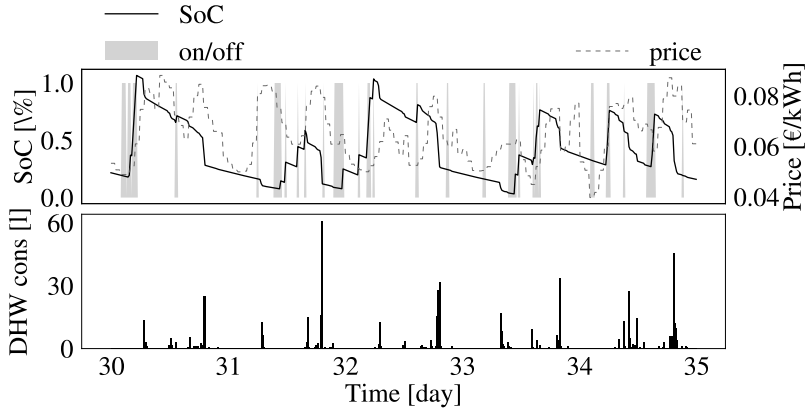


Figure 4.10: Final five simulation days: FQI-Belpex experiment.

Table 4.4: Total cost and p-values for intervals of 5 days: FQI-Belpex experiment. Mean of 20 simulations, vertical bars indicate 95% confidence interval.

Days		[0,6]	[7,13]	[14,20]	[21,27]	[28,34]	**[0,34]**
No pre-training	μ	5.19	6.02	4.34	4.17	3.95	**23.67**
$\sum \lambda_t E_t$ [€]	σ	0.40	0.16	0.25	0.32	0.19	**0.46**
Pre-trained	μ	4.90	5.90	4.09	3.84	3.84	**22.56**
$\sum \lambda_t E_t$ [€]	σ	0.45	0.18	0.15	0.22	0.22	**0.49**
p-value		0.04	0.03	$7e^{-4}$	$6e^{-4}$	0.11	$\mathbf{1.2e^{-8}}$

group. As expected since $\mathcal{F}$ is almost only filled with target domain transitions at that time.

With pre-trained DQL, and using states as defined in (4.4), the Belpex price profile resulted in a mean total cost of €23.22 with a standard deviation of 0.64. These results seem to indicate FQI is more suited in a setting with daily varying prices. Additionally, with the Belpex price profile, MPC resulted in a total cost of €25.5. This result indicates that, with this price profile, more system-identification effort is needed to get satisfactory MPC performance.

4.5 Conclusion

We demonstrated the use of pre-training with domain randomization in a residential DR setting for two different RL algorithms and with two ToU

price profiles. We show adapting a pre-trained policy to the target domain is significantly faster than starting from scratch. Pre-trained policies allow for cost savings immediately at the start of operation, which, in the considered cases, results in a significant price reduction throughout the simulated period. Using DQL, pre-training results in 8.8 % cost reduction compared to starting from scratch and a 32.2 % reduction compared to a hysteresis controller. Although the pre-training approach differs slightly between DQL and FQI, both algorithms benefit very similarly from it. Moreover, despite the two mass model's more accurate SoC estimate, compared to the uniform model, the RL agent manages to benefit from pre-training almost equally. Average total cost with the two mass model is €16.2 while it is €15.6 with the uniform model, for the DQL agent. Our experiments also showed pre-trained RL agents, with domain randomization, show similar cost savings when there is a larger discrepancy between source domain and target domain. This discrepancy can be caused by modelling (uniform model) or by system-identification (bad guess of model parameter U).

Finally, we showed it is relevant to use a non-linear policy which adapts to the target domain, as this results in 26 % cheaper operation than applying optimal source domain control actions in the target domain, *i.e.* MPC. While state-of-the-art MPC might result in better performance, our results indicate that the uniform model suffices for RL but does not suffice for MPC.

It is not certain that dissimilarity between uniform / two mass model and stratified model is as large as the dissimilarity between the stratified model and a real buffer. Therefore, future work is directed towards verifying the presented approach for transfer from simulation to practice. Additionally, we aim to investigate other methods of including prior knowledge in the policy as we have showed this can result in better policies.

Chapter 5

Multi-agent Transfer Learning in Demand Response

This chapter is based on

- [57] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Transfer learning for Demand Response of a Multi-Agent Battery and Electric Water Heater System". In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2021. DOI: 10.1109/ISGTEurope52324.2021.9640081

5.1 Introduction

In recent years, some multi-agent RL applications have gained attention. For example, Reymond *et al.* [66] show that independent agents at household level perform better than a centralised grid agent. Furthermore, Kazmi *et al.* [30] use model-based RL in a large scale field test in The Netherlands. Their work shows multi-agent settings can accelerate learning of TCL behaviour. Thus, in addition to showing the possibilities of multi-agent settings, Kazmi *et al.*'s work also shows tranfer learning can be used to achieve higher data efficiency in DR applications that use RL. The same conclusion can be drawn from our earlier work. The previous chapter (and Peirelinck *et al.* [56]) show that transfer learning, with domain randomization, can be used to jumpstart RL control performance. All these results are promising, as data inefficiency has until recently been considered to be the greatest drawback of RL methods [62].

In transfer learning one aims to use the knowledge extracted from a source domain or task to increase (learning) performance in a target domain or task [50]. This work aims to further develop transfer learning applications in the DR and smart grid domain. Additionally, it builds upon previous multi-agent settings. Similar to Kazmi *et al.* [30], other agents' experience is used to accelerate training time. But, additionally, this occurs in a multi-agent setting. In contrast to other multi-agent work [66, 30], we consider a two agent system, where one agent is part of the environment of the other. The experiments conducted consider a household equipped with an EWH and a rooftop PV installation, with only a rudimentary forecast. The electricity consumption of the household is charged based on a ToU-pricing mechanism, without a feed-in tariff. There is thus a clear incentive for self-consuming locally generated PV power. A first agent controls the EWH for this purpose, using FQI. Later, a battery, with a second FQI agent, is added to the system. The goal of the battery agent is to further reduce the household electricity bill. The EWH-agent already holds information about the household inflexible load, and the EWH consumption pattern. Therefore, the experiments aim to investigate if it is worthwhile to use this experience to pre-train the battery agent. The household with EWH will thus act as the source domain and the household with EWH and battery will act as the target domain. Furthermore, the experiments aim to show that a scenario where one agent is part of the environment of the other is feasible. In order to do so, we assume that the EWH agent has priority over the battery agent, as the EWH can be considered the least flexible appliance. By imposing this priority, the EWH-agent's state and action can be part of the battery agent's state. The main hypothesis of this work is that it is possible to combine two RL agents, with the same cost-function, and use the experience replay memory of the former, to pre-train the latter.

The second section of this chapter recapitulates the presented control problem as an MDP. The third section presents the experiments and results. The section thereafter concludes this part of the work and explores future research directions.

5.2 Environment Models and Reinforcement Learning Algorithms

Both agents, with their respective state- and action-spaces, are deployed in an MDP. It has been assumed the EWH model is not known a priori and the agent solely relies on its experiences to learn a control policy. In contrast, a very rudimentary battery model is assumed to be available to the agent for offline pre-training. However, none of the exogenous data is assumed available. These data sources can only be read online, during simulation.

This section quickly recapitulates the MDP, models and algorithms used throughout this chapter.

5.2.1 The Markov Decision Process

As typical in residential DR settings, the exact transition probabilities are unknown. The goal of the RL agent is to learn a policy $\pi : \mathcal{X} \to \mathcal{U}$ which can cost-efficiently operate in the environment, given this uncertainty. The following subsections introduce each part of the MDP.

State-space

The agent perceives battery and EWH state as $x^B \in \mathcal{X}^B$ and $x^{\text{EWH}} \in \mathcal{X}^{\text{EWH}}$, respectively. This is only a partial observation of their respective internal state. The observed state is augmented with history $h = 4$, in order to recover the Markov property [70, 23].

Because the states of both environments are used by both agents, they need to be transferable. Therefore, SoC is used to describe the energy content of EWH and battery. The EWH and battery states, as observed by the agent, are given by equation (5.1) and (5.2), respectively.

$$x_t^{\text{EWH}} = [SoC_t^{\text{EWH}}, \ldots, SoC_{t-h}^{\text{EWH}}, R_t^{PV}, t_{\cos}, t_{\sin}] \tag{5.1}$$

$$x_t^{B} = [SoC_t^{B}, \ldots, SoC_{t-h}^{B}, u_t^{\text{EWH}}, SoC_t^{\text{EWH}}, R_t^{PV}, t_{\cos}, t_{\sin}] \tag{5.2}$$

Furthermore, by defining the states like this, it is clear that the EWH has priority over the battery. That is, the EWH-agent always decides its action first, after which the action is included in the battery agent's observed state.

The goal is to minimise the need for forecasting. In order to get a practical scenario, only a forecast of the peak power production of the current day F_t^{PV} is assumed available. We assume a perfect forecast. And thus, the agent only observes this forecast combined with power production and consumption of the previous time step, P_{t-1}^{PV} and P_{t-1}^{m}, respectively. All this is incorporated in R_{PV}, as defined by

$$R_t^{PV} = \frac{P_{t-1}^{PV} - P_{t-1}^{m}}{F_t^{PV}}. \tag{5.3}$$

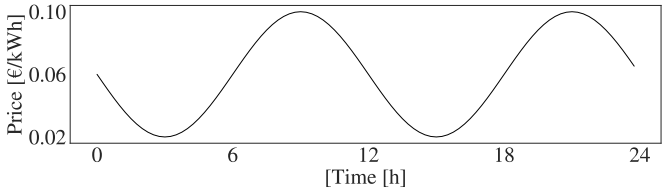


Figure 5.1: Time-of-Use price profile.

Action-space

The action-spaces of both agents differ slightly, as it is impossible to discharge an EWH. Or rather, it can only be discharged by consuming hot water, which is a decision made by the users. Therefore, the EWH-agent has a binary action-space $\mathcal{U}^{\text{EWH}} = \{0, 1\}$ indicating if the heating element should be turned on or off, with rated power P_r^{EWH}.

Additional to charging and turning off, the battery agent can also decide to discharge the battery. The battery action-space therefore equals $\mathcal{U}^B = \{-1, 0, 1\}$.

Cost-function

From the DR setting described earlier, it is clear the agent's main aim is to reduce operating costs and, therefore, increase self-consumption. There is no feed-in tariff present and the consumed electrical energy is charged based on a ToU-pricing mechanism, with price λ_t at time step t. In what follows, a sinusoidal price profile, as shown in Fig. 5.1 [61], has been assumed. The remaining load of the (typical Belgian) household E_t^m and the PV generation E_t^{PV} are assumed uncontrollable to the agents. However, they do influence the cost. The cost at time step t is defined by (5.4).

$$\begin{aligned} c_t &= \lambda_t \max(E_t^{\text{EWH}} + E_t^B + E_t^m - E_t^{PV}, 0) \\ &= \lambda_t \max(E_t^{\text{net}}, 0) \end{aligned} \tag{5.4}$$

5.2.2 Electric Water Heater and Battery Model

Electric Water Heater Model

A two-mass buffer model has been used, as presented in Chapter 2 and by Sinha *et al.* [56, 75]. The model is based on heat-balance equations between the hot layer (denoted by subscript h) and cold layer (subscript c). A hard thermocline has been assumed between both layers. There are thus two temperatures inside the boiler, the hot water layer temperature T_h and the cold layer temperature T_c. The layers have a variable volume V_c and V_h, which always adds up to the buffer volume $V = V_c + V_h = 203\,l$. A Canadian tap-demand profile with 5-minute granularity has been used. The average DHW demand per day equals $189\,l$ [15]. The temperature measurement inside the buffer T_b is located in the middle. At this location, water has a minimal allowed temperature of $T_{\min} = 45°C$ and maximal allowed temperature of $T_{\max} = 55°C$.

Battery Model

The energy based battery model, as presented in Chapter 2, has been used for this use case. The energy balance inside the battery is given by (2.15). The energy content of the battery is limited to $E^B_{\max} = 2.4\,\text{kWh}$. Furthermore, a charge and discharge efficiency of $\eta = 95\,\%$ is assumed.

5.2.3 Reinforcement Learning Algorithm

This chapter aims to show FQI can be used in a multi-agent setting and its performance can be improved by adding transfer learning. Our adapted version of FQI, as presented in Algorithm 2, separates the SDP in $T = 96$ supervised learning problems, one for every time step t, assuming a granularity of 15 minutes. The RL agents considered here use T random forests [6] to approximate each time step's Q-function. Using this Q-function, the EWH- and battery agent's policies are defined by (5.5) and (5.6), respectively.

$$\pi^{\text{EWH}}(x_t, t) = \underset{u}{\operatorname{argmax}}\, Q^{\text{EWH}}_t(x_t, u) \tag{5.5}$$

$$\pi^{B}(x_t, t) = \underset{u}{\operatorname{argmax}}\, Q^{B}_t(x_t, u) \tag{5.6}$$

5.3 Experiments and Results

5.3.1 Experiment Set-up

The experiment consists of three steps. During the first and third step, new data is seen by the respective agent. In the second step, the battery agent is pre-trained using earlier experienced transitions. Fig. 5.2 visually represents the different steps of the experiment.

First, the EWH-agent has been assumed to be in operation for two months. This MDP is visually represented in the top of Figure 5.2. During these two months, the EWH-agent learns a control policy that minimises EWH operating cost, using FQI. As Ruelens *et al.* [70] show, one month of data suffices to learn a near-optimal control policy with FQI. Therefore, after these two months of training, the training process stops and every step t the agent evaluates $\pi^{\text{EWH}}(x^{\text{EWH}}, t)$ to decide upon action u_t^{EWH}, which is then mapped onto a power signal $u_{\text{phys},t}^{\text{EWH}}$ according to (2.17).

Second, after two months, the battery agent has been added to the system. As mentioned, the battery agent's environment thus incorporates battery, EWH and EWH-agent. Each time step, the battery agent waits for the EWH-agent's decision before evaluating its own policy $\pi^B(x^B, t)$. Therefore, u_t^{EWH} can be part of x^B, as given in (5.2).

This set-up has been simulated using two different approaches. In a first, naive, approach, the battery agent has been added without any prior information, *i.e.*, in a complete model-free fashion. This means the middle (gray) step of Fig. 5.2 is omitted. In a second approach, the battery agent is pre-trained using the battery model described earlier, in combination with the transitions that are experienced by the EWH-agent. More precisely, there is one off-line training sweep through the transitions of the EWH-agent, as if the battery agent has already been added and is experiencing these transitions. Using the battery model, rewards can be recalculated as if the battery agent had taken its action. Only after this pre-training phase, visually represented by the middle of Figure 5.2, the battery agent and EWH-agent are operating together and experience unseen state-transitions, as depicted in the bottom of Figure 5.2. Comparing the performance of these two approaches allows us to verify our main hypothesis: pre-training the battery agent leads to cheaper operating cost of the combined system. All experiments have been performed five times, to incorporate variability. Results shown are always the mean over these five runs.

The results have also been compared with Rule-based Control (RBC). This controller always charges both appliances when net power consumption is

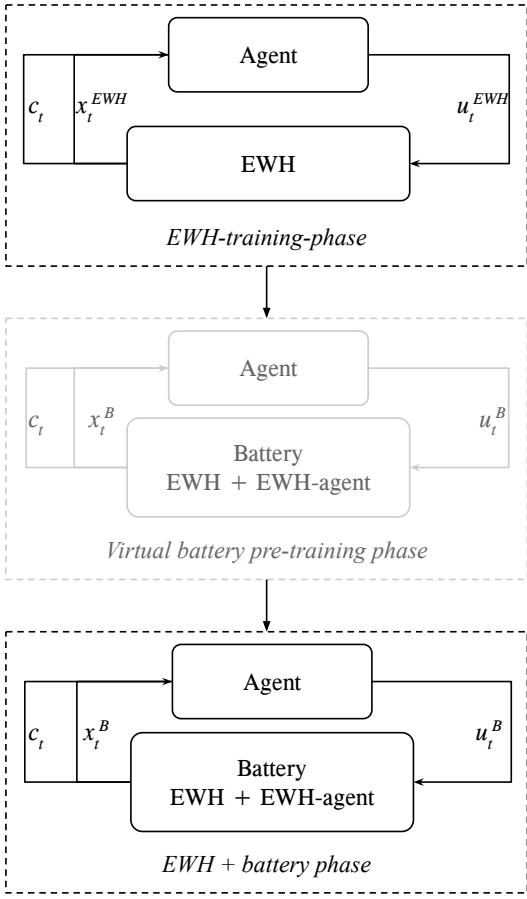


Figure 5.2: Visual representation of the different experiment phases. Top figure: first phase of the experiment, only the EWH-agent is active and learning. Middle figure: second phase of the experiment, the battery agent is learning with the data saved by the EWH-agent during the first phase (this part is omitted in the naive approach). Bottom figure: third phase of the experiment, both agents are active and controlling their respective appliance.

negative, *i.e.*, $E_t^{PV} > E_t^m$. Both backup controllers also internally overwrite this control signal to keep the appliances within operable bounds. A second rule always discharges the battery when $E_t^{PV} < E_t^m$.

5.3.2 Results

Fig. 5.3 shows an overview of the experiment where FQI has been combined with pre-training (FQI+PT). The figure aims to show how the different agents affect overall power consumption. The figure visualizes power consumption of the final three days, *i.e.*, when both agents have been deployed. The grey area depicts net power consumption without battery and EWH power consumption ($P_t^m - P_t^{PV}$). The blue dotted line is net power consumption with EWH power consumption added ($P_t^m + P_t^{\text{EWH}} - P_t^{PV}$). It is clear that the EWH added some major power peaks to the overall profile. The EWH-agent has shifted those peaks to times when overall consumption is rather low. However, due to DHW consumption, this is not always possible. But, for example in the final day shown (day 61), EWH power consumption occurs early morning and late evening, rather than during times with high inflexible load. The full black line depicts overall net power consumption ($P_t^m + P_t^{\text{EWH}} + P_t^B - P_t^{PV}$). The battery has an overall flattening effect to the power profile. For example, early morning of the first shown day, the battery agent has successfully reduced the first power peak as a result of EWH power consumption. Additionally, it has reduced the whole evening inflexible power consumption to zero. It is also clear from this figure that the battery charges during the day, and discharges during the morning and evening.

Fig. 5.4 shows SoC of both energy buffers, together with the price profile λ and, again, net inflexible power consumption. The figure shows that the battery agent charges the battery when net power consumption is negative, as it is only allowed to do so at these times. Contrary to this behaviour, the EWH-agent mainly schedules charging times when prices are low. Intuitively, this can be explained because the hot water consumption of the household occurs mainly early morning and evening. Due to the limited buffer capacity of an EWH, it is not possible to charge many hours in advance.

The following two tables aim to compare the performance of the three studied control paradigms: RBC, FQI and FQI with pre-training (FQI+PT). Table 5.1, compares all cost-related performance metrics of the simulations. It shows FQI+PT is the overall cheapest control method in terms of cost. It is 8.6 % cheaper than FQI and 28.5 % cheaper than RBC. The first column shows that cost reductions starts at the first week of battery agent operation. This confirms the hypothesis that pre-training does allow a jump start. Furthermore, the

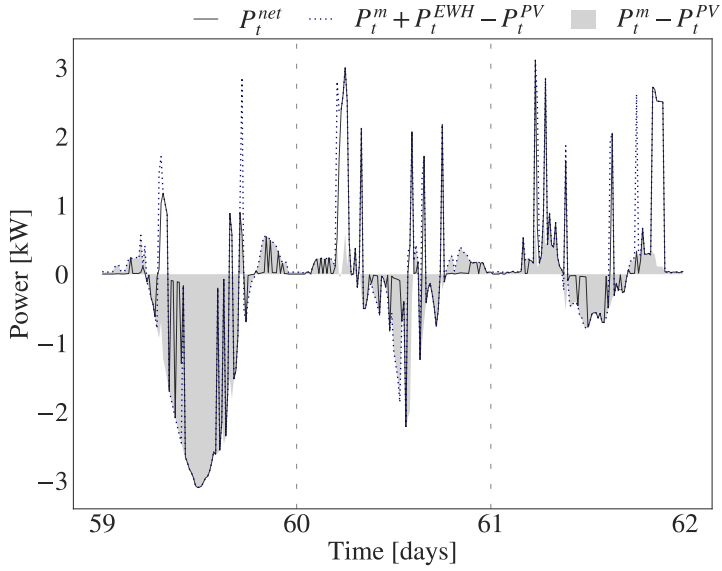


Figure 5.3: Power consumption of three example days (FQI+PT experiment).

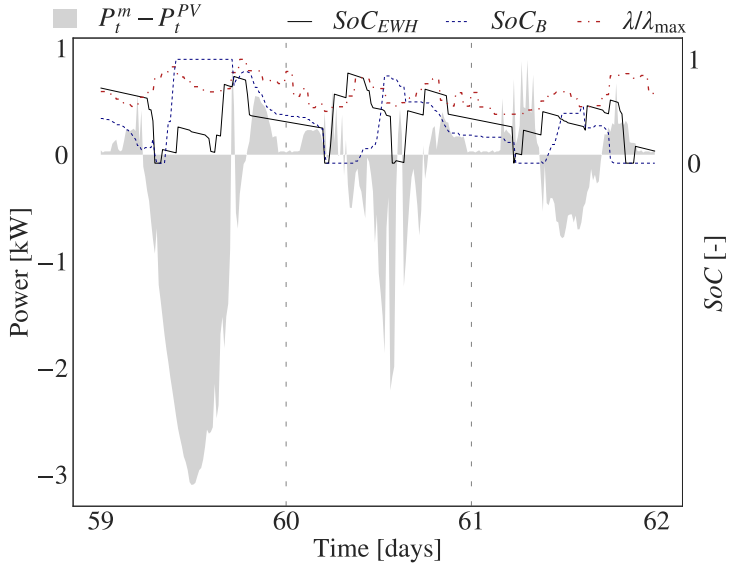


Figure 5.4: SoC and price profile of three example days (FQI+PT experiment).

Table 5.1: Summary of main cost performance indicators.

	$\sum_{t=0}^{7\cdot 96} c_t[€]$	$\sum c_t[€]$
RBC	4.57	34.53
FQI	3.76	27.03
FQI+PT	3.11	24.7

Table 5.2: Summary of main energy consumption performance indicators.

	$\sum E_{\text{net}}$[kWh]	SC-rate	$E_{\text{off-take}}$[kWh]	$E_{\text{off-take}}^{\lambda<\mu_\lambda}$[kWh]
RBC	57.41	0.60	459.41	111.99
FQI	61.83	0.54	516.20	302.80
FQI+PT	64.60	0.57	496.41	311.61

second column shows the total cost. This gives an indication that cost gains are achieved for the whole simulation period.

Table 5.2, shows all energy consumption related performance metrics. The first column gives the net energy consumption over the whole two month period $E_{\text{net}} = \sum_t E_t^{\text{net}}$. The second column shows the household's overall Self-Consumption (SC) rate, *i.e.*, the amount of locally consumed PV production. Since, with RBC almost all consumption is shifted to times with local PV production, it is expected that this performs best, taking only this metric into account. However, from the previous table it is clear that the lowest cost is achieved using FQI+PT. Both RL-agents have thus managed to learn a trade-off between consuming power when electricity prices are low or consuming power when local production is available. The two last columns of Table 5.2 confirm this. $E_{\text{off-take}}$ gives the summed net consumption per quarter, *i.e.*, $E_{\text{off-take}} = \sum_t \max(E_t^{\text{net}}, 0)$. And $E_{\text{off-take}}^{\lambda<\mu_\lambda}$ gives this measurement, but only taking into account quarters with a price lower than the mean price μ_λ. Thus, shifting energy consumption to low-priced hours does increase overall energy consumption (compared to RBC). However, due to *smart* shifting (to quarters with low prices) overall operating costs turns out to be lower. We observe that the total energy consumption of FQI is higher than that of FQI+PT, but consumption during relatively cheap hours is lower. It can be concluded that, by being pre-trained, an RL-agent can better find the balance between keeping additional energy consumption low, while still shifting energy consumption to low-priced quarters.

5.4 Conclusions

In this chapter, FQI has been used in a multi-agent setting. The setting consisted of an EWH- and battery agent, both having the goal of decreasing electricity cost in a ToU-princing scenario. By imposing a strict priority amongst the two agents, it was possible to give valuable information about the EWH agent's behaviour to the battery agent. This allowed to successfully incorporate the EWH agent into the battery agent's environment. Furthermore, transfer learning has been used to jumpstart the performance of the battery agent. Simulations confirm the main hypothesis set out in the introduction of this chapter: it is possible and beneficial to use experience gained by the EWH agent to pre-train the battery agent. Pre-training is performed by re-iterating over the EWH agent's training data, with a battery model. For a combined system of EWH, battery, PV and inflexible load, over a period of two months, our results show a 28.5 % decrease in final operating cost comparing pre-training with RBC and a 8.6 % decrease comparing to a scenario without pre-training. This is especially promising as our pre-training approach holds no additional complexity, but merely uses the already available data sources efficiently. We show that a pre-trained battery agent's control policy better maintains the balance between an overall increase in energy consumption and energy arbitrage.

Future research directions will aim to extend this research to scenarios with real electricity prices, and, potentially, a feed-in tariff. Furthermore, extending this work to multiple agents is also considered. For example, in a fully electrified residential heating system for spatial heating and DHW, there is room for three agents: a DHW buffer-, a heat pump- and a battery agent.

Chapter 6

Incorporating Domain Knowledge in Demand Response Learning Problems

6.1 Introduction

Renewable Energy Sources (RES) are reshaping the energy sector landscape. Due to their decentralised and intermittent nature, market and tariff designs are challenged. In the Flemish region of Belgium the energy regulator (VREG) has recently announced a change of distribution fee design [94]. Previously, Flemish residential electricity distribution fees were energy-based. The rise of residential PV installations and net-metering meant a reduction in income for the Distribution Network Operator (DNO). With the introduction of digital metering, the regulator takes the opportunity to introduce a capacity tariff, starting from 2022 [94]. The regulator motivates its decision by arguing that the main costs of the DNO are capacity-based rather than energy-based [94].

By tying the distribution grid fees to power rather than energy consumption, the VREG encourages consumers to reduce their peak power consumption. This offers an interesting application for DR. TCLs are considered excellent appliances for DR, as they have an inherent energy buffer [59, 55]. In other DR applications with TCLs, RL has shown promising results. For instance, Ruelens *et al.* [70] showed that FQI can reduce energy consumption cost of a heat pump by 19 %, compared to a default controller, in an energy arbitrage

scenario. Mbuwir *et al.* [44] apply the same algorithm for local optimisation in their transactive control framework. Their methodology manages to reduce grid congestion using flexibility available in the microgrid's heat pumps.

Additionally, multiple examples exist of successful EWH control with RL [70, 54, 59, 30, 56]. Reducing peak power consumption can (partly) be accomplished by local PV self-consumption. Self-consumption is a DR application in itself, and earlier work has applied RL for maximising residential self-consumption. Soares *et al.* [77, 78] use a model-based RL algorithm to control residential batteries and heat pumps. In their field test, they achieve a 68 % average self-consumption rate. This means, on average, 68 % of heat pump energy is covered by local PV generation. In a similar field test, using the same model-based RL approach, but only scheduling the EWH heat cycle, De Somer *et al.* [13] manage to increase PV self-consumption with 20 %. The DR application considered in this chapter differs from Soares *et al.* [77, 78] and De Somer *et al.* [13] as the goal is not to maximise self-consumption. Rather, maximising self-consumption is an implicit goal when minimising peak power consumption. In the past, we have already touched upon a capacity tariff scenario [61]. However, the capacity tariff treated here is different, and is as proposed by the Flemish regulator in Belgium, *i.e.*, VREG [94].

State-of-the-art RL has improved over time. Since its introduction, policy gradient methods [81] have gradually gained interest. Compared to value iteration methods, the policy gradient approach uses a function approximator that explicitly represents the policy [81]. To further improve the policy gradient, which updates the approximator of the policy, an estimate of the expected future reward can be used [33]. This is achieved by the introduction of a critic [33]. These actor-critic methods thus combine the advantages of value iteration and policy iteration [33]. PPO [73] is the latest introduced family of actor-critic methods and now widely used in RL research. PPO has proven to work well in different domains [73]. Consequently, we have opted to use PPO in this work.

Here, PPO has been adapted for automated DR when a consumer with an EWH is billed, at least partly, based on quarter hourly peak power consumption. The RL controller's aim is to minimise final energy cost by turning the EWH on or off. Cost can be minimised by avoiding energy consumption when other (inflexible) loads are already using power or by self-consuming locally generated PV power. To achieve this goal, the agent is first pre-trained based on readily available consumption [15, 4] and production [63] data. The main contribution of this work is the modification of PPO to this setting and a state-space design that explicitly takes into account transfer learning. Additionally, the state-space inherently includes the dynamic nature of the environment, *i.e.*, its seasonality. Given that local load and PV forecasts are difficult to obtain, we aim to reduce the need for forecasting. Furthermore, while other work [77, 78, 13, 61] focusses

on either load flattening or self-consumption, we implicitly account for both objectives. We test our approach on data from real households, obtained from a field-test in The Netherlands [13, 20].

A naive implementation of the capacity tariff, as designed by the VREG, results in (very) sparse rewards because the final power cost of Flemish households will be billed once a year, based on the past 12 months' Mean Month Peak (MMP) (as discussed next in Section 6.2). RL is known to be less effective in sparse reward scenarios [41]. Mataric [41] proposes reward shaping for accelerated learning in such sparse reward tasks. The reward shaping methodology can be used to incorporate domain knowledge in the reward function, and thus guide the learner to perform better. Although reward shaping seems promising, altering the reward function can possibly alter the resulting optimal policy [48]. However, PPO does not guarantee optimality. As such, one can never be sure to have found to optimal policy, even without reward shaping [73]. The considered reward function aims to guide the learner to the intuitive goals of the considered control problem, *i.e.*, minimising power peaks and maximising self-consumption. However, final performance is assessed based on the final electricity bill of the considered household.

This chapter is divided in four sections. Section 6.2 gives a more detailed formulation of the capacity tariff design, formulates the MDP and lays out the RL algorithm and its modifications. The chapter then goes on to Section 6.3, presenting the experiments and discussing their results. Finally, Section 6.4 concludes this part of the work and gives some future work directions.

6.2 Problem Formulation

The first part of this section elaborates on the capacity tariff, as designed by the Flemish electricity and gas regulator (VREG). The second part defines the SDP, and formulates it as an MDP. The final part presents the algorithm used to solve the MDP.

6.2.1 Tariff Design

A current Flemish residential electricity bill approximately consists of three parts: (i) the energy cost [€/kWh], (ii) the distribution costs [€/kWh] and (iii) taxes and levies (depends partly on energy or power consumption). In the remainder of this work, we assume this simplified decoupling of the Flemish electricity bill. Traditionally, residential consumers only have a Ferraris meter installed,

which is limited to metering of net energy consumption. The introduction of residential PV installations caused the DNO to see its income reduced, as so-called *prosumers* have relatively low net energy consumption and, as mentioned, distribution costs are energy-based. To compensate for this loss, a *prosumer-tariff* was introduced. This tariff is charged based on the power inverter capacity of the PV installation [€/$\mathrm{kW_{inverter}}$] [94].

With the introduction of digital metering comes the ability to measure electricity consumption and production separately, and have finer grained measurement points. Together with the observation that distribution grid investment cost is mainly tied to grid capacity (and not energy transported), the regulator opted for a capacity-based distribution fee, from 2022 onwards. This approximately results in the second part of the earlier mentioned electricity bill being capacity-based [€/kW] [94].

The peak power to calculate the bill is based on the quarter-hourly measurements of the digital meter. The capacity fee of a residential consumer will be calculated based on the running MMP of the past 12 months. It only takes into account net off-take, *i.e.*, there is no capacity fee based on grid injection. Thus, assuming P_t^m is the quarter-hourly power consumption time-series of the current month m in kW, *i.e.*, the digital meter output, and λ_P is the price per kW in euro, the capacity fee F of a residential consumer is calculated by Eq. (6.2).

$$\mathrm{MMP} = \frac{\sum_{i=m-12}^{m} \max(2.5, \max_t(P_t^i))}{12} \tag{6.1}$$

$$F = \lambda_P \cdot \mathrm{MMP} \tag{6.2}$$

Eq. (6.1) implies a minimal capacity fee based on a monthly peak power consumption of 2.5 kW, as in the tariff design [94]. The main aim of this work is to minimise the final energy bill, *i.e.*, including the parts related to energy consumption and taxes. The total energy bill is calculated by

$$C_{\mathrm{capacity}} = \lambda_E \cdot E + \lambda_P \cdot \mathrm{MMP} + \lambda_{\mathrm{tax}}^E \cdot E + \lambda_{\mathrm{tax}}, \tag{6.3}$$

with λ_E the energy price, E the total energy consumption of considered year, λ_{tax}^E the taxes charged based on energy consumption and λ_{tax} the fixed taxes payable per year. Eq. (6.3) is used to judge agent performance.

In RL, the reward function can be used to direct the agent to optimal parts of the solution space, based on expert knowledge. For example, here we know self-consumption of locally generated PV will be beneficial for both reducing energy consumption cost and reducing the MMP. Furthermore, λ_{tax} is independent of MMP and E. As a consequence, neither equation (6.2) nor equation (6.3) are used as the MDP's reward-function. The following part of this section formulates

the control problem as MDP by presenting the state-space, action-space and reward-function.

6.2.2 Markov Decision Problem

The problem is formulated as a discrete-time MDP with time steps of length $\Delta t = 15$ minutes. The MDP consists of state-space $\mathcal{X}$, action-space $\mathcal{U}$, reward-function $r : (\mathcal{X}, \mathcal{U}) \to \mathbb{R}$ and state-transition probabilities $p(\cdot|x, u)$, given by the EWH model. The agent is unaware of these transition probabilities.

With the setting as mentioned in the previous section, the main objective of this work is to reduce peak power consumption. Intuitively, reducing peak power consumption goes hand in hand with increasing self-consumption.

Our reward-function aims at formalising this intuition. Following the MDP framework, this objective is translated to reward-function (6.4).

$$r_1(x_t, u_t) = \begin{cases} \min(P^c - P_t^{net}, 0) + P_t^{sc} & P_t^{\text{EWH}} \neq 0 \\ 0 & P_t^{\text{EWH}} = 0 \end{cases} \tag{6.4}$$

Where P^c is 2.5 kW, P_t^{EWH} is the electrical power consumed by the EWH, P_t^{net} is the net power consumption and P_t^{sc} is the self-consumed EWH power, at quarter t. Given P_t^D is the household's other inflexible electrical energy demand and P_t^{PV} is the electrical power produced by the PV installation, the net power consumption P_t^{net} and the EWH self-consumption P_t^{sc} are calculated by equations (6.5) and (6.6), respectively.

$$P_t^{net} = P_t^{\text{EWH}} + P_t^D - P_t^{PV} \tag{6.5}$$

$$P_t^{sc} = \min(\max(0, P_t^{PV} - P_t^D), P^r) \tag{6.6}$$

An additional challenge considered here is the aim of reducing the need for extensive local PV and demand forecasts. On top of that, the policy will be applied to different residential buildings and households. Therefore, to facilitate policy transfer, the state-space is designed to be independent of environment parameters. For instance, the learned policy should preferably be independent of inverter capacity, as different households will have different PV installations. Preference for policy independence on environment parameters can be illustrated by imagining a simple policy that turns the EWH on whenever PV power production is above 2 kW. When this policy would be transferred to a different household, this absolute number does not apply anymore. Moreover, even within one single household this number would have to change between seasons.

At time-step $t \in \{0, \ldots, 95\}$, state $x_t \in \mathcal{X}$ is defined by

$$x_t = \{\mu_t^T, \mu_{t-1}^T, \ldots, \mu_{t-3}^T, \Delta_{T_t^b - T_{\min}}, F_{PV}^E, \frac{P_t^{PV}}{F_{PV}^P}, t_{\cos}, t_{\sin}, t\}, \tag{6.7}$$

with μ_t^T the mean temperature inside the buffer at time-step t, $\Delta_{T_b^t - T_{\min}}$ the difference between the sensor measurement and the minimal allowed water temperature, $t_{\cos}$ and $t_{\sin}$ the projection of the time-step on a circle [56]. Two state features are dependent on a forecast of the local PV production: F_{PV}^E and F_{PV}^P. They are defined by equations (6.8) and (6.9), respectively. F_{PV}^E is the forecast of the current day's energy consumption, scaled with an upper bound of the possible energy production given the inverter power P_{inv}. F_{PV}^P is a forecast of the peak power production of that same day. The agent has no (explicit) information on local power demand.

$$F_{PV}^E = \sum_{i=\lfloor t/96 \rfloor}^{\lfloor t/96 \rfloor + 96} \frac{E_i^{PV}}{96/2 \cdot \Delta t \cdot P_{\text{inv}}} \tag{6.8}$$

$$F_{PV}^P = \max_{i=\lfloor t/96 \rfloor}^{\lfloor t/96 \rfloor + 96} \left(\frac{E_i^{PV}}{\Delta t} \right) \tag{6.9}$$

This work considers a binary action-space. Every quarter t, the agent chooses an action $u_t \in \mathcal{U} = \{0, 1\}$, turning the EWH on or off. When temperature constraints are violated, the backup controller overrules the agent's action according to Eq. (2.17).

A two-layer EWH model, similar to earlier work [13, 77, 78, 56] and as presented in Chapter 4, is used as a virtual test-bed.

6.2.3 Algorithm

We use PPO [73] to obtain a stochastic policy, maximising expected total reward, given reward-function (6.4). Both actor and critic are parametrised using a (separate) NN, with parameters θ_{actor} and θ_{critic}, respectively. The update rules have been presented in Chapter 4.

To improve convergence speed and policy transfer capabilities, the actor-NN is tailored to the task at hand, *i.e.*, expert knowledge is incorporated into the actor NN design. The domain knowledge is included in such a way that it does not restrict the applicability to one household or one type of TCL. The actor's NN has been designed based on three main observations:

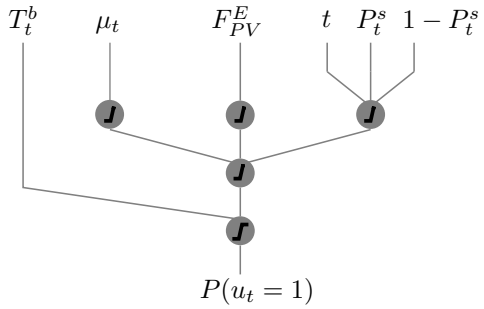


Figure 6.1: Actor sub-NN architecture, with $P_t^s = P_t^{PV}/F_{PV}^P$.

- Time-steps close to each other have a similar policy.
- If the current net consumption is close to the forecasted maximum PV output of the day, it's quite likely beneficial to heat water.
- The backup controller affects the policy and can be incorporated into the actor.

More specifically, the actor is divided in 24 subnetworks, *i.e.*, one for every hour of the day. Each of these networks has 5 layers, of which the first is a feature extraction layer. The subnetwork architecture is shown in Fig. 6.1. The output of subnetwork $\mathcal{T} \in \{0, \ldots, 23\}$, with parameters $\theta_{\mathcal{T}}^{\text{actor}}$, equals the probability of choosing action $u_t = 1$. The final neuron of each subnetwork implements Eq. (2.17), assuring temperature stays within comfort bounds. The actor thus uses only part of the observable state and, so does the critic. The full observable state is given in (6.7). The part used by the actor and critic is defined by (6.10) and (6.11), respectively. The time-step is not omitted in x_t^{actor}, as it is needed to determine if it is better to turn the EWH on at night or not.

$$x_t^{\text{actor}} = \{T_t^b, \mu_t^T, F_{PV}^E, \frac{P_t^{PV}}{F_{PV}^P}, t\} \tag{6.10}$$

$$x_t^{\text{critic}} = \{\mu_t^T, \mu_{t-1}^T, \ldots, \mu_{t-3}^T, \Delta_{T_t^b - T_{\min}}, F_{PV}^E, \frac{P_t^{PV}}{F_{PV}^P}, t_{\cos}, t_{\sin}\} \tag{6.11}$$

The critic's NN has a conventional fully-connected architecture, with two layers of 28 neurons. Apart from the neuron representing the backup controller, every neuron uses a ReLu activation function. All NNs have been implemented in PyTorch [52].

Table 6.1: General metrics of training- and test data, for 1 year.

	DHW cons. [l/day]	$\sum E^D$ [kWh]	$\sum E^{PV}$ [kWh]
Training	188.93	3781.55	3766.76
House 1	57.7	2929.57	7894.09
House 2	116.06	5966.85	8269.54
House 3	33.03	3627.04	7467.20
House 4	29.05	3761.71	8296.90
House 5	185.98	5335.82	7920.31

6.3 Experiments and Results

6.3.1 Experiment Set-up

In earlier chapters we have shown transfer learning increases performance for agents applied in a DR setting [56]. Hence, the task is separated in a pre-training and test phase. The pre-training phase only uses readily available data. Three data streams are needed. First, simulated PV production data is taken from the *ninja* tool developed by Pfenninger and Staffell [63]. Second, electrical load data is generated using *Strobe* [4]. Third, training phase simulations use DHW consumption data from Edwards *et al.* [15]. These three sets contain data of one year and pre-training lasts 15 simulation-years. After the initial pre-training phase, the obtained parameters θ_{actor} and θ_{critic} are used as initial values for the test phase. During the test-phase, which lasts one simulation-year, we use real residential PV, load and DHW data, from five houses, obtained from a field test in The Netherlands [13, 20]. Data is available starting from the first of October. Each experiment is conducted 10 times to account for variability in both phases. Table 6.1 gives some general metrics of the training- and test data. Clearly, a variety of households has been considered. Algorithm 4 shows the complete training pipeline.

Algorithm 4 Pre-training with PPO

1: **Input**: h, γ, λ, ϵ, T
2: Initialise θ^{actor}, θ^{critic}
3: **while** time $<$ 10 years **do**
4: Act according to Algorithm 3, with training household data
5: Use obtained θ^{actor}, θ^{critic}
6: **while** time $<$ 1 year **do**
7: Act according to Algorithm 3, with household i data

The presented approach has been compared with three other control approaches:

Hysteresis Control (HC), Rule-based Control (RBC) and a non-expert version of RL (PPO). The hysteresis controller assures user comfort and turns the EWH on or off according to Eq. (2.17). Like RL, RBC adds an additional layer on top of this hysteresis controller. The implemented rule-based controller turns the EWH on for four hours, at a fixed time t_{RBC}. While the choice of t_{RBC} may affect control performance, its optimal value is unknown beforehand. Therefore, all RBC simulations have been run four times, with $t_{\text{RBC}} \in \{10, 11, 12, 13\}$ hour. The non-expert version of RL also uses PPO as a training algorithm. However, the actor has a more traditional fully connected NN with two layers of 10 neurons each. This allows to confirm if the tailor made actor increases performance.

In the next section we show different result metrics, such as the final energy bill of the considered household, calculated by (6.3). The Flemish regulator has published capacity tariff values λ_P for different DNO regions. We have chosen the average value $\lambda_P = 47.78\,€/\text{kW}$.

6.3.2 Results

This part presents the results of the simulations. We start with a visualisation of the three main control approaches (HC, RBC, (expert) RL). Thereafter, we compare their performance differences more thoroughly. For the sake of simplicity, only expert RL has been considered at the start. It will be referred to as RL from now on. Only at the final detailed comparison of the main metrics, non-expert RL is included.

Fig. 6.2 shows several test-phase days for the three considered control approaches. In each subfigure, the grey area depicts net uncontrollable load, *i.e.*, inflexible load P_t^D minus PV production P_t^{PV}. The EWH power demand P_t^{EWH} for each control approach is depicted by different line-styles. As explained earlier, simulations have been conducted with several values for t_{RBC}. In all subfigures of Fig. 6.2, results of the t_{RBC} value which resulted in the best final RBC performance for the considered house has been shown.

Fig. 6.2a shows the three days immediately succeeding pre-training. At first sight, RBC seems to be a good initial control approach, consuming power when local PV production is high. This is, however, as expected, as the choice for RBC is the result of expert domain knowledge. These three shown days further suggest RL has resulted in similar behaviour as RBC. This confirms the observation that RBC is a rather good initial approach.

Fig. 6.2b shows three winter days, which illustrate how RL further improves upon RBC. On the second and third day, the RL agent turns the EWH on at

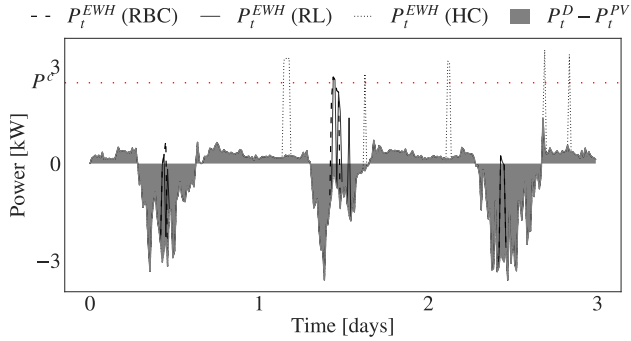
night, thus avoiding a heating cycle in the afternoon (during RBC hours), when PV production is low and consumption high. Clearly, HC performs worse than the other approaches, as it does not take into account net consumption at all.

Finally, Fig. 6.2c presents three spring days. All have quite a lot of PV production and, therefore, clear negative consumption in the afternoon. However, PV production is intermittent and has certain drops during the day. While the RL agent has no forecast of PV production in the next quarter, it has current net consumption as an input. As a result, and contrary to RBC, it temporarily interrupts heating when PV output suddenly drops. This is illustrated by the third day (day 242) of Fig. 6.2c.

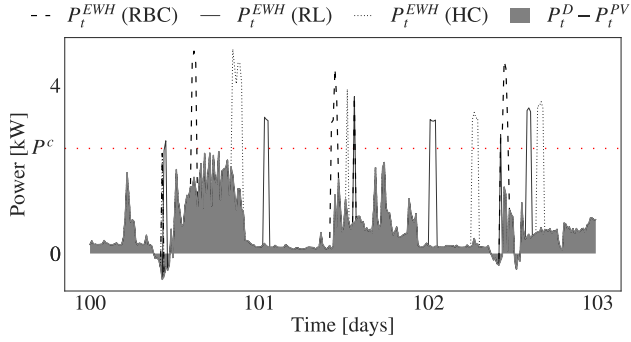
Fig. 6.3 gives a more extensive overview of the results. It presents an entire test-phase simulation year for house 3. The bottom graph shows total PV energy production $\sum E^{PV}$, total inflexible load $\sum E^{D}$ and average DHW consumption for each month. These measures are useful for interpretation of control performance. Second, the middle graph shows self-consumption ratio of each month. For RBC and RL it shows mean and standard deviation of all simulation runs. The self-consumption ratio is the share of total EWH power consumption which has been locally produced by the PV installation. Intuitively, it is clear that this has to be maximised. The graph shows slightly better RL performance, compared to RBC, for all months. Especially in months with low PV production, RL manages to capture more of the scarce local renewable energy for own consumption. Finally, the top figure shows $P^{\max}$, *i.e.*, the month peak, for each month and for all three control approaches, with mean and standard deviation for RBC and RL. Remembering our final goal, $P^{\max}$ should be minimised. RL outperforms the other control approaches for all months, except November, in this household.

In the end, the main goal is reducing the final yearly energy bills of households in Flanders, which are prone to a capacity tariff. Fig. 6.4 shows all interesting metrics for the whole test-phase year and for all houses. The top figure shows the MMP, calculated by (6.1). (Expert) RL outperforms RBC, HC and non-expert RL for each household. More precisely, on average over all houses, RL reduces the MMP by 16.85 % compared to HC, and by 6.84 % compared to RBC. In absolute numbers, this is a reduction of 0.90 kW and 0.33 kW, respectively.

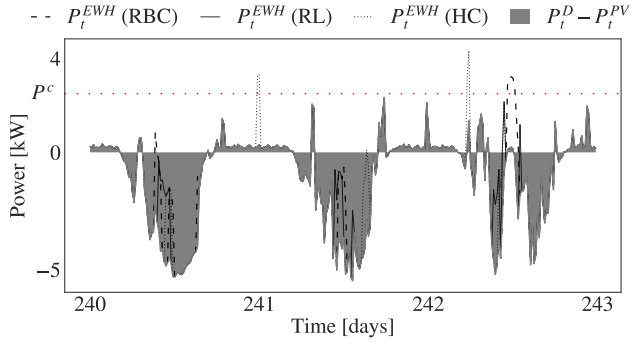
As in Fig. 6.3, the middle graph of Fig. 6.4 shows the EWH's self-consumption ratio. This figure shows that, for each household, RL manages to shift EWH power consumption better to quarters in which local energy production is available. Compared to HC and RBC, expert RL captures 192.34 % and 9.93 % more PV production, respectively. This means that, over the year on average, 495.21 kWh more EWH energy consumption is locally produced, compared to HC, or 57.24 kWh compared to RBC. Of further interest is that expert RL



(a) House 3: 1 - 3 October ($t_{\text{RBC}} = 11\text{h}$).



(b) House 1: 9 - 11 January ($t_{\text{RBC}} = 10\text{h}$).



(c) House 2: 29 - 31 May ($t_{\text{RBC}} = 11\text{h}$).

Figure 6.2: Example days of three controllers and three houses, with best choice for t_{RBC}.

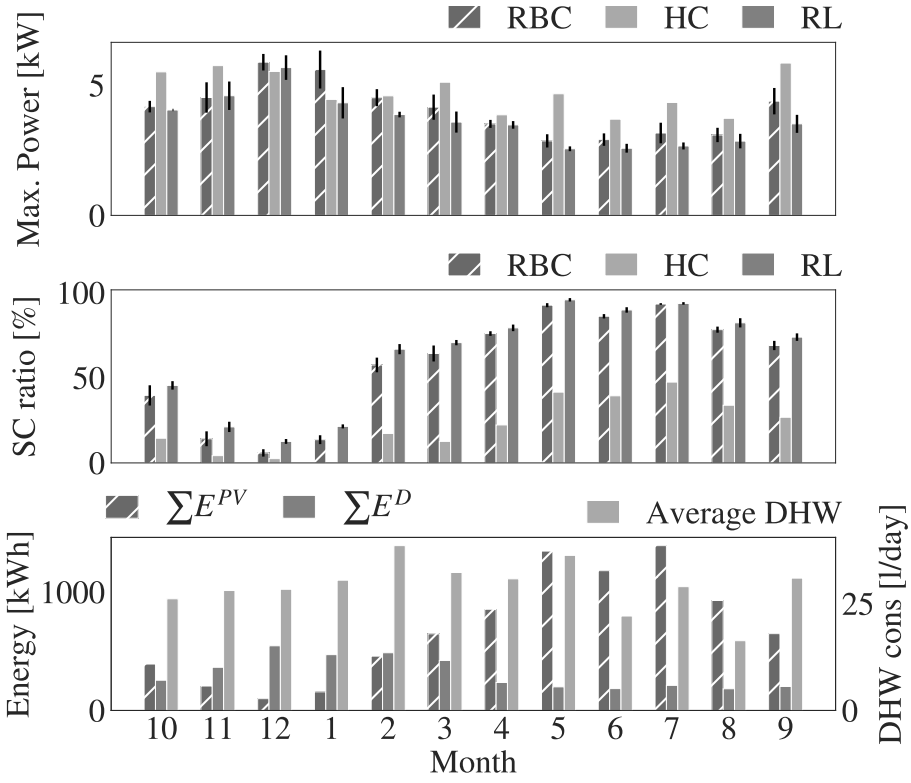


Figure 6.3: Test-year monthly performance metrics (house 4).

manages to greatly reduce variability of the final performance, compared to non-expert RL.

The bottom bar chart shows the final (electrical) energy bill of each household, defined by (6.3). (Expert) RL is 14.51 % cheaper than HC and 4.59 % cheaper than RBC. For these households and the considered capacity cost, this thus results in an average reduction in cost of €78.07 and €22.13 compared to HC and RBC, respectively. Moreover, by incorporating domain knowledge into the RL algorithm, we have managed to reduce costs with 6.68 % or €32.96. Additionally, the performance variability has decreased.

The training-phase is an important step in the design and implementation of this RL set-up for DR. RL is known to be data inefficient [73]. A pre-training-phase mitigates this drawbacks as data is less scarce in this phase. Fig. 6.2a has illustrated that, because of the pre-training-phase, the RL agent performs its

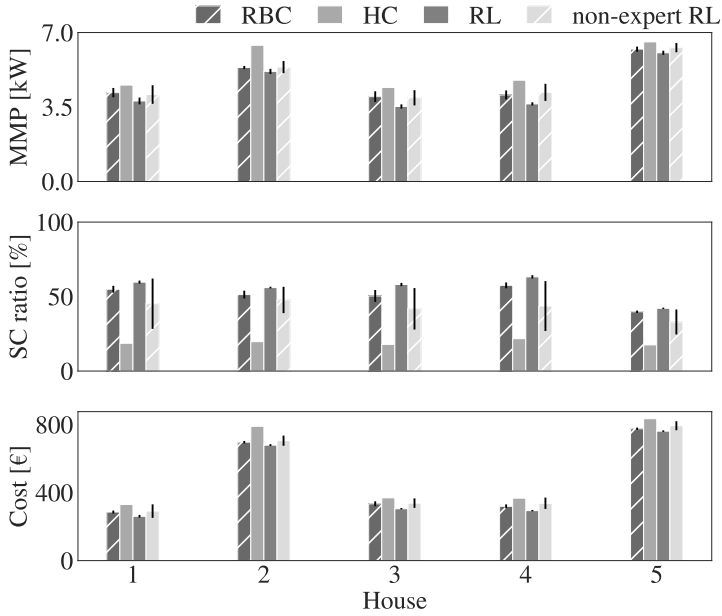


Figure 6.4: Final (test-phase) results for all houses.

task well immediately at the start of the test-phase. Fig. 6.5 aims to inspect if final pre-training performance affects test-phase performance. This figure shows that, although pre-training's final year mean reward varies between simulation runs, test-phase mean reward is always rather similar. This result suggests that, while it is important to pre-train the agent, it might not be necessary to somehow find the *best* pre-trained agent, as the average Pearson correlation coefficient between the mean reward of the final year of pre-training and the mean test-phase reward is 0.23.

6.4 Conclusions

We adapted state-of-art RL, based on PPO, to the DR setting and have applied it for EWH control in a capacity tariff use case. The considered setting is highly relevant in Belgium, as the (Flemish) regulator has decided to introduce a capacity tariff for residential consumers as of 2022. In this scenario, the identified goal was two-fold: (i) reduce the EWH peak power consumption and (ii) increase self-consumption of local rooftop PV production. In our test-phase, we have used real-life data from five houses, all with different consumption

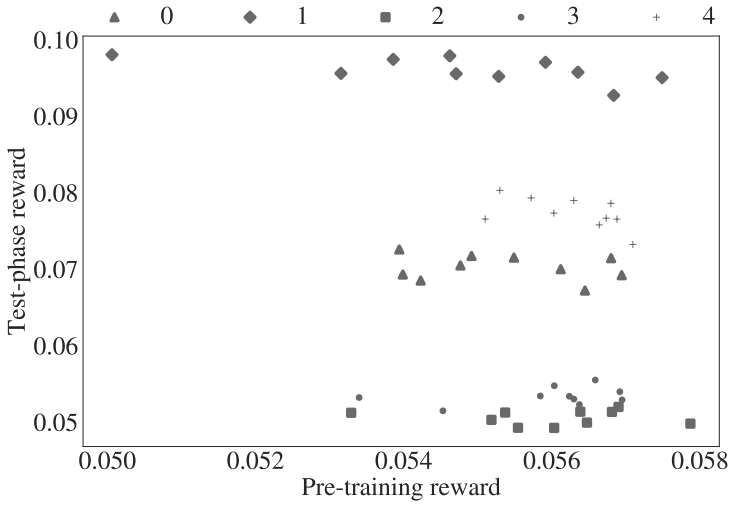


Figure 6.5: Scatter plot of mean pre-training-phase reward versus mean test-phase reward.

patterns, but equipped with the same EWH. Beforehand, the agent had been trained with readily available data. The setting is particularly challenging as the agent has no forecast of residential load and limited knowledge on future PV production. Furthermore, we proposed an extension of PPO where domain knowledge is incorporated into the actor's design. The results indicate that, using this approach, above human-level control performance is achieved.

In our experiments, we compared the RL controller with RBC and HC. The experiments showed, for all considered houses, expert RL outperforms both these two other control approaches. Self-consumption ratio was increased by 192.34 % compared to HC, and by 9.93 % compared to RBC. This resulted in a final energy bill reduction of 14.51 % and 4.59 % compared to HC and RBC, respectively. In absolute numbers, this translates to an average reduction of €78.07 and €22.13, respectively, in the yearly energy bill of the considered households. One should note that, while RBC performs relatively well, it is the result of expert knowledge and the best choice of t_{RBC} differs for each house. Tuning this parameter needs expert knowledge, which comes at a cost. In contrast, once deployed, expert RL automatically adapts to its environment. This has been shown in our experiments by the low correlation coefficient between pre-training and final mean reward.

We believe with the introduction of the capacity tariff there is an opportunity for Flemish residential consumers to adopt smart control approaches for EWHs

(and other TCLs). Aggregators can potentially provide the necessary technology. Therefore, our future work is directed towards incorporating local control for reducing the MMP within an aggregator framework.

Chapter 7

Conclusion

7.1 Overview and Answers to the Research Questions

DR can facilitate the shift towards more sustainable means of generating power for our society. Not only does DR mitigate the challenges related to the intermittent nature of (most) sustainable power sources, it also enables a more efficient operation of the grid and the electric appliances connected to it. We believe DR can facilitate a shift towards a more sustainable future, while also providing more comfort to end-users. Unfortunately, residential DR is challenging. A high degree of automation is requested from the users, as it is impractical to manually monitor different grid signals at all times. Traditional control approaches struggle with the variability of the control problem, the heterogeneous nature of the appliances and the required scalability. RL has proved to help mitigating these challenges. One of the main drawback of RL, however, is its data inefficiency. This work has presented different approaches to improve data efficiency of RL algorithms and has succesfully implemented them in pratical residential DR use cases.

7.1.1 Summary and Discussion

TCLs account for a relatively large portion of a household's energy consumption. At the same time, users consider these appliances mostly as a black-box. They require the service of, for example, hot water. Furthermore, the service (*e.g.*,

heat) TCLs offer and the energy they consume (*e.g.*, electricity) are somewhat decoupled. Meaning that TCLs are a source of demand flexibility. Therefore, these applliances have been identified to be very promising for DR applications. Throughout this work, an EWH with DHW buffer has been used as a guiding example.

Chapter 2 explains how the DR control problem is an SDP, *i.e.*, at discrete time steps a decision about the next control action has to be taken, based on the current state of the environment. Thereafter, this chapter formalises the SDP as an MDP. The MDP framework is used as a guide throughout this work. It allows to incorporate RL into DR settings. By means of the MDP framework, the state space, action space and reward functions have been defined. Furthermore, the particularities with respect to the boundary conditions, *i.e.*, comfort guarantees, have been presented. Throughout this work different EWH models have been used. Some have only been used as a source domain in transfer learning settings, others have been used as a virtual test-bed. All these models have been presented in Chapter 2. Additionally, the three major DR tasks that have been identified through the course of this work, and their practical relevance, are discussed. Finally, the chapter presents the core of the used RL algorithms.

In **Chapter 3** the fundamentals of transfer learning for DR applications have been presented. While the transfer learning space is large, no single taxonomy exists. This makes it difficult for researchers and practicioners to recognise potential applications and to know which methods to use in which settings. Therefore, this chapter proposes a transfer learning taxonomy. Based on an extensive literature review, we propose three categories: transductive transfer learning, inductive transfer learning and transfer learning for model-based RL. Finally, transfer learning applications within DR have been reviewed and categorized. This allows to draw the conclusion that transfer learning has the potential to increase both performance and generalizability of RL agents. Domain knowledge can accelerate training. However, in DR, transfer learning is still in its infancy and requires more research. Most work considers (naive) sharing of NN weights. Moreover, especially transductive learning with feature sharing and inductive learning remain open challennges. Therefore, the remaining chapters can be interpreted as an exploration and feasability study of transfer learning techniques within DR.

Chapter 4 extends DQL and FQI with domain randomization. Domain randomization is a technique used to pre-train RL agents, such that they are able to generalise to unseen dynamics. Using this technique, the experiments show that fine-tuning a pre-trained agent in the target domain is significantly faster than training one from scratch in the target domain. This is the case even when the source domain is not sufficiently accurate for MPC. The RL

agent thus generalises to unseen dynamics, as it outperforms optimal source domain actions applied in the target domain. Furthermore, the results indicate that domain randomization is robust against distribution changes in the source domain, something which is not achieved when naively sharing weights between agents.

A two-agent learning setting has been presented in **Chapter 5**. The setting involves an agent entering the environment. Both agents share features by means of a priority between them. Additionally, transfer learning using the experience replay memory and a virtual pre-training phase has been presented. The proposed approach allows to jumpstart performance of the additional agent, while only using data that has already been gathered by the first agent.

Previous chapters focused on using data sources for transfer learning, be it in the form of simulation models or previously gathered data. **Chapter 6** focuses on incorporating domain knowledge directly into the learning pipeline. Of course, the transfer learning possibilities of different data sources are not neglected. Therefore, the proposed methodology also pre-trains the RL agent with readily available data sources, collected for other purposes. The experiments show an agent that has been pre-trained with average household and climate data can generalise to unseen households. Additionally, domain expert knowledge is included in the PPO actor NN design. This allows to incorporate general rules about the control problem. Furthermore, this additional step only needs to be done once, in the design phase of the agent.

7.1.2 Answering the Research Questions

Every chapter has helped towards answering the research questions defined at the start of this work. These questions are answered below.

1. Which residential DR settings would benefit from RL based control and how are consumers incentivized to participate in these DR programs?

There are a lot of potential DR applications and tasks within a residential setting. *Energy arbitrage*, *self-consumption* and *peak power reduction* have been identified as the most promising. These settings are of practical relevance because they have already been implemented in real electricity markets or will be implemented in the near future. Furthermore, these settings are customer-centric, fitting well with RL. Customers are incentivized via their general electricity bill. Therefore, good performing RL agents can directly decrease the user's costs.

2. How can we design cost-effective and generally applicable methods that benefit maximally from available data *before* the agent is deployed at the consumer's site?

Chapter 4 and 5 both answered this question from a different point of view. First, Chapter 4 looked at an approach to pre-train agents such that they are generally applicable in all types of households thereafter. The proposed approach, using domain randomization, allows to pre-train an agent with a general and simplified buffer model. The same pre-trained agent can then be applied in different households. This results in a cost-effective approach, as the training step is only required once. The results show it is faster to fine-tune this general policy to each individual household, than it is to learn an individual household's policy from scratch. All pre-training can be done off-site. As a result, the impact of RL exploration on the final households is decreased. Users will, therefore, experience relatively good control behaviour from the agent immediately after its installation. Secondly, Chapter 5 considered a two-agent system. After one agent has been active in the environment for quite a while, a second enters. This Chapter then looked at an approach to maximally exploit information which has already been captured from the considered household. It proposed a novel approach to use the experience replay memory of the first agent to pre-train the second (new) agent. As a result, initial performance of the second agent increased. Thus, while Chapter 4 examined a strategy to initialise a general agent which will have the same task in different households, Chapter 5 examined an approach to initialise an agent which will have the same task in the same household, with an additional appliance.

3. If available data is not sufficient to guarantee performance from the start of operation, how can we incorporate domain knowledge in a general way, *i.e.*, without the need for extensive modelling?

Chapter 6 proposes a novel method to incorporate domain expertise in the learning pipeline of a PPO agent. By designing a NN layout, tailored to the task at hand, it is possible to include general behaviour guidelines within the agent's policy. As the domain knowledge can be described in a general way, independent on the household, a domain expert is only required once. Thereafter, the same agent can be applied to different households. The tailor-made agent can then still be pre-trained to even further enhance initial performance. This has also been shown by the experiments. The agent is pre-trained with publicly available data and applied to households with different user behaviour. In all cases, performance gains were observed, compared to a naive PPO agent and compared to RBC.

7.2 Challenges of Transfer and Reinforcement Learning in Demand Response

While many challenges of RL have been mitigated over the past years, many remain. Some of the main challenges, such as data inefficiency, can be mitigated by means of transfer learning. However, transfer learning introduces its own set of challenges. The discussion here starts with some of the remaining challenges for applying RL in DR settings.

First of all, many RL researchers have already pointed out in the past that RL algorithm performance depends on the values of its hyperparameters. While RL manages to reduce the need for DR and modelling experts, it introduces the need for RL experts. In recent years, important steps have been set in the right direction. For example, experience shows that PPO is less dependent on hyperparameters than DQL. Unfortunately, throughout this work it has become clear that in order to adapt and tune RL algorithms to DR contexts, quite some domain expertise from both the RL and energy domain is necessary. It is not likely that transfer learning will change this.

A second challenge lays in the design of the reward function. Chapter 6 has shown that it is not always straightforward to match a practical setting to the MDP framework. Additionally, several goals, such as self-consumption and peak power reduction, should sometimes be achieved at the same time. This creates to need to design tailor-made reward functions, which again requires expert knowledge from both the RL and energy domain.

A third challenge arises due to the dependence on data. This challenge presents itself both for general RL and for transfer learning. Every timestep an RL algorithm requires a certain input from the environment, *i.e.*, the state. In this work, a 15 minute interval has been chosen. Earlier work has indicated that this seems to be an interval that is achievable. Even though the feed forward pass of an RL algorithm is computationally relatively cheap, it is not unlikely that computation or communication performance is too slow to reach this deadline. Algorithms, therefore, need to be robust against data loss. Additionally, the quality of exogenous data used as an input, *e.g.* weather information, might be reflected in the quality of the learned control policy, just as the quality of source domain data might be reflected in the initial control policy. This can thus reduce initial performance in the target domain.

A fourth challenge presents itself in the dimensioning of the considered TCLs. While throughout this work conventional industry rules-of-thumb have been used to guide the sizing of the DHW buffer and its rated power, it has become clear these rules may have to change. When local energy production is cheap and

households start focusing more on harnessing their flexibility, the conventional trade-off between comfort and cost can change.

Finally, within transfer learning it is not clear how close of a match the source and target domain need to be. In the worst case scenario, when both domains are too unrelated, this can result in negative transfer. Fortunately, our work has indicated that, for example by using domain randomization, the performance in the target domain is robust against major distribution changes in the source domain.

7.3 Future opportunities

Throughout the different chapters of this dissertation several future research opportunities have been identified. A more general research direction that has not yet been explored is the effect of these control strategies on the electricity grid. In recent years, more regulation with respect to demand aggregators has been put in place, as the goals of grid operators and aggregators might not necessarily align. Although all considered DR settings in this work are customer-centric and could be achieved behind the meter at the customers's site, it can be interesting to investigate the dynamics of a distribution feeder with a high penetration of DR.

Secondly, as proposed in Chapter 2, future work is necessary to explore the possibility to combine and change different DR objectives. As customers change contract or preference, their trained agents should adapt. Transfer learning techniques for quick adaption to new scenarios can potentially mitigate some of the related challenges.

This dissertation has focussed on adapting, implementing and investigating the feasability of different RL algorithms for different DR applications. In Chapter 4, a method for transfer from simulation to practice has shown promising results. Future work could aim to verify this in a real-life setting and transfer a control policy learned in simulation to a real EWH.

Chapter 5 explored multi-agent learning in a specific setting. In this setting, transfer learning has shown its benefits. The results indicate that RL agents can use each others experience memory to jumpstart their own performance. An interesting future work direction is to explore bigger multi-agent settings. As an initial step, one could explore a household which also includes a heat pump for space heating.

Finally, an aggregator could potentially act as a service provider to a household, providing the presented DR services. At the same time, this aggregator could

then cluster all its customer for other DR purposes, such as frequency control. It remains to be seen how RL agents can balance the goals of the aggregator and the individual household. Furthermore, not all agents necessarily have to act upon a signal from the aggregator. For example, if the cluster contains 1 MW of flexibility and the aggregator only requires 500 kW, only 50 % has to alter its default behaviour. Researchers have only started looking at methods to distribute a signal across a cluster of RL agents.

Bibliography

[1] M. Alshiekh, R. Bloem, R. ¨. Udiger Ehlers, B. Könighofer, S. Niekum, and U. Topcu. "Safe Reinforcement Learning via Shielding". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 18)*. 2018. DOI: 10.5555/3504035.3504361.

[2] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. "Learning Dexterous In-Hand Manipulation". In: (Aug. 2018). arXiv: 1808.00177.

[3] I. Antonopoulos, V. Robu, B. Couraud, D. Kirli, S. Norbu, A. Kiprakis, D. Flynn, S. Elizondo-Gonzalez, and S. Wattam. "Artificial intelligence and machine learning approaches to energy demand-side response: A systematic review". In: *Renewable and Sustainable Energy Reviews* 130 (Sept. 2020), p. 109899. ISSN: 1364-0321. DOI: 10.1016/J.RSER.2020.109899.

[4] R. Baetens, R. De Coninck, F. Jorissen, D. Picard, L. Helsen, and D. Saelens. "Openideas-an open framework for integrated district energy simulations". In: *Proceedings of Building Simulation 2015*. 2015, pp. 347–354.

[5] T. Borsche, F. Oldewurtel, and G. Andersson. "Scenario-based MPC for Energy Schedule Compliance with Demand Response". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 10299–10304. ISSN: 14746670. DOI: 10.3182/20140824-6-ZA-1003.01284.

[6] L. Breiman. "Random forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 08856125. DOI: 10.1023/A:1010933404324.

[7] J. Cao, D. Harrold, Z. Fan, T. Morstyn, D. Healey, and K. Li. "Deep Reinforcement Learning Based Energy Storage Arbitrage With Accurate Lithium-ion Battery Degradation Model". In: *IEEE Transactions on Smart Grid* (2020), pp. 1–1. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.2986333.

[8] J. Cigler, D. Gyalistras, J. Široky, V. Tiet, and L. Ferkl. "Beyond theory: the challenge of implementing model predictive control in buildings". In: *Proceedings of 11th Rehva World Congress, Clima*. Vol. 250. 2013.

[9] G. T. Costanzo, S. Iacovella, F. Ruelens, T. Leurs, and B. J. Claessens. "Experimental analysis of data-driven control for a building heating system". In: *Sustainable Energy, Grids and Networks* 6 (June 2016), pp. 81–90. ISSN: 2352-4677. DOI: 10.1016/j.segan.2016.02.002.

[10] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters. "Sharing Knowledge in Multi-Task Deep Reinforcement Learning". In: *2020 International Conference on Learning Representations*. 2020.

[11] R. D'hulst, A. Delnooz, E. Laes, and D. Six. *Onderzoek naar de tariefstructuur van de periodieke distributienettarieven Finaal rapport*. Tech. rep. 2018. URL: https://www.vreg.be/sites/default/files/20180111_studie_vreg_statusrapport_v11_-_eindrapport.pdf.

[12] R. De Coninck and L. Helsen. "Practical implementation and evaluation of model predictive control for an office building in Brussels". In: *Energy and Buildings* 111 (Jan. 2016), pp. 290–298. ISSN: 03787788. DOI: 10.1016/j.enbuild.2015.11.014.

[13] O. De Somer, A. Soares, T. Kuijpers, K. Vossen, K. Vanthournout, and F. Spiessens. "Using Reinforcement Learning for Demand Response of Domestic Hot Water Buffers: a Real-Life Demonstration". In: *2017 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. 2017, pp. 1–7.

[14] Digitaal Vlaanderen. *Teruglevercontract voor elektriciteit die u zelf produceert en op het distributienet injecteert | Vlaanderen.be*. 2021. URL: https://www.vlaanderen.be/energieleveranciers-en-energiecontracten/teruglevercontract-voor-elektriciteit-die-u-zelf-produceert-en-op-het-distributienet-injecteert (visited on 01/19/2022).

[15] S. Edwards, I. Beausoleil-Morrison, and A. Laperrière. "Representative hot water draw profiles at high temporal resolution for simulating the performance of solar thermal systems". In: *Solar Energy* 111 (Jan. 2015), pp. 43–52. ISSN: 0038092X. DOI: 10.1016/j.solener.2014.10.026.

[16] EPEX SPOT Brussels. *EPEX SPOT Belgium*. 2018. URL: https://www.belpex.be/.

[17] European Council. *EUR-Lex - 32019L0944 - EN - EUR-Lex*. 2019. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32019L0944 (visited on 01/17/2022).

[18] A. A. Farooq, A. Afram, N. Schulz, and F. Janabi-Sharif. "Grey-box modeling of a low pressure electric boiler for domestic hot water system". In: *Applied Thermal Engineering* 84 (June 2015), pp. 257–267. ISSN: 1359-4311. DOI: 10.1016/J.APPLTHERMALENG.2015.03.050.

[19] A. Gammerman, V. Vovk, and V. N. Vapnik. "Learning by Transduction Abstract". In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. 1998, pp. 148–155. DOI: 10.5555/2074094.2074112.

[20] D. van Goch, M. Eulen, S. Lodeweyckx, and C. Caerts. *Rennovates, Flexibility Activated Zero Energy Districts, H2020*. Tech. rep. 2017, pp. 29–31. URL: https://cordis.europa.eu/project/id/680603/results.

[21] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2020. URL: http://www.gurobi.com.

[22] N. Hary, V. Rious, and M. Saguan. "The electricity generation adequacy problem: Assessing dynamic effects of capacity remuneration mechanisms". In: *Energy Policy* 91 (Apr. 2016), pp. 113–127. ISSN: 0301-4215. DOI: 10.1016/J.ENPOL.2015.12.037.

[23] M. Hausknecht and P. Stone. "Deep Recurrent Q-learning For Partially Observable MDPs". In: *Computing Research Repository* (2015). arXiv: 1507.06527.

[24] C. Hesse, J. Schulman, V. Pfau, A. Nichol, O. Klimov, and L. Schiavo. *OpenAI Retro Contest*. 2018. URL: https://openai.com/blog/retro-contest/.

[25] S. Iacovella, F. Ruelens, P. Vingerhoets, B. Claessens, and G. Deconinck. "Cluster Control of Heterogeneous Thermostatically Controlled Loads Using Tracer Devices". In: *IEEE Transactions on Smart Grid* 8.2 (2015), pp. 1–9. DOI: 10.1109/TSG.2015.2483506.

[26] F. Jorissen and L. Helsen. "Towards an Automated Tool Chain for MPC in Multi-zone Buildings". In: *International High Performance Buildings Conference* (Jan. 2016). URL: https://docs.lib.purdue.edu/ihpbc/202.

[27] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, R. Sepassi, G. Tucker, and H. Michalewski. "Model-Based Reinforcement Learning for Atari". In: *The International Conference on Learning Representations (ICLR) 2020*. Mar. 2020. arXiv: 1903.00374.

[28] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. "Physics-informed machine learning". In: *Nature Reviews Physics 2021 3:6* 3.6 (May 2021), pp. 422–440. ISSN: 2522-5820. DOI: 10.1038/S42254-021-00314-5.

[29] H. Kazmi, D. Didden, N. Wiese, and J. Driesen. "Sample efficient reinforcement learning with domain randomization for automated demand response in low-voltage grids". In: *IEEE Journal of Emerging and Selected Topics in Industrial Electronics* (Oct. 2021). ISSN: 2687-9735. DOI: 10.1109/JESTIE.2021.3117119.

[30] H. Kazmi, F. Mehmood, S. Lodeweyckx, and J. Driesen. "Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems". In: *Energy* 144 (2018), pp. 159–168. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.12.019.

[31] H. Kazmi, J. Suykens, and J. Driesen. "Large-Scale Transfer Learning For Data-Driven Modelling Of Hot Water Systems". In: *Proceedings of Building Simulation 2019: 16th Conference of IBPSA*. Vol. 16. IBPSA, May 2020, pp. 2611–2618. DOI: 10.26868/25222708.2019.210352.

[32] S. Kim, R. Mowakeaa, S. J. Kim, and H. Lim. "Building energy management for demand response using kernel lifelong learning". In: *IEEE Access* 8 (2020), pp. 82131–82141. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2991110.

[33] V. R. Konda and J. N. Tsitsiklis. "Actor-Critic Algorithms". In: *Advances in Neural Information Processing Systems 12 (NIPS 1999)*. 1999, pp. 1008–1014.

[34] D. Kong, X. Kong, J. Xiao, J. Zhang, S. Li, and L. Yue. "Dynamic pricing of demand response based on elasticity transfer and reinforcement learning". In: *2019 22nd International Conference on Electrical Machines and Systems, ICEMS 2019*. Institute of Electrical and Electronics Engineers Inc., Aug. 2019. ISBN: 9781728133980. DOI: 10.1109/ICEMS.2019.8921683.

[35] T. Lampe and M. Riedmiller. "Approximate model-assisted Neural Fitted Q-Iteration". In: *Proceedings of the International Joint Conference on Neural Networks*. IEEE, Sept. 2014, pp. 2698–2704. ISBN: 9781479914845. DOI: 10.1109/IJCNN.2014.6889733.

[36] N. Leemput, F. Geth, J. Van Roy, P. Olivella-Rosell, J. Driesen, and A. Sumper. "MV and LV Residential Grid Impact of Combined Slow and Fast Charging of Electric Vehicles". In: *Energies 2015, Vol. 8, Pages 1760-1783* 8.3 (Mar. 2015), pp. 1760–1783. ISSN: 19961073. DOI: 10.3390/EN8031760.

[37] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous control with deep reinforcement learning". In: *4th International Conference on Learning Representations*. Ed. by ICLR. 2016. arXiv: 1509.02971.

[38] M. Liu, S. Peeters, D. S. Callaway, and B. J. Claessens. "Trajectory Tracking With an Aggregation of Domestic Hot Water Heaters: Combining Model-Based and Model-Free Control in a Commercial Deployment". In: *IEEE Transactions on Smart Grid* 10.5 (2019), pp. 5686–5695. ISSN: 1949-3053. DOI: 10.1109/tsg.2018.2890275. arXiv: 1805.04228.

[39] S. Liu and G. P. Henze. "Experimental analysis of simulated reinforcement learning control for active and passive building thermal storage inventory: Part 2: Results and analysis". In: *Energy and Buildings* 38.2 (Feb. 2006), pp. 148–161. ISSN: 0378-7788. DOI: 10.1016/J.ENBUILD.2005.06.001.

[40] Y. Ma. *Model Predictive Control for Energy Efficient Buildings*. Tech. rep. 2012.

[41] M. J. Mataric. "Reward Functions for Accelerated Learning". In: *Machine Learning Proceedings 1994* (Jan. 1994), pp. 181–189. DOI: 10.1016/B978-1-55860-335-6.50030-1.

[42] B. V. Mbuwir, K. Paridari, F. Spiessens, L. Nordstrom, and G. Deconinck. "Transfer learning for operational planning of batteries in commercial buildings". In: *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids, SmartGridComm 2020* (Nov. 2020). DOI: 10.1109/SMARTGRIDCOMM47815.2020.9303016.

[43] B. V. Mbuwir, F. Spiessens, and G. Deconinck. "Benchmarking regression methods for function approximation in reinforcement learning: Heat pump control". In: *Proceedings of 2019 IEEE PES Innovative Smart Grid Technologies Europe, ISGT-Europe 2019* (Sept. 2019). DOI: 10.1109/ISGTEUROPE.2019.8905533.

[44] B. V. Mbuwir, F. Spiessens, and G. Deconinck. "Distributed optimization for scheduling energy flows in community microgrids". In: *Electric Power Systems Research* 187 (Oct. 2020), p. 106479. ISSN: 03787796. DOI: 10.1016/j.epsr.2020.106479.

[45] B. Mbuwir, F. Ruelens, F. Spiessens, and G. Deconinck. "Battery Energy Management in a Microgrid Using Batch Reinforcement Learning". In: *Energies* 10.11 (2017), p. 1846. ISSN: 1996-1073. DOI: 10.3390/en10111846.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing Atari with Deep Reinforcement Learning". In: *NIPS Deep Learning Workshop 2013*. 2013. arXiv: 1312.5602.

[47] E. Mocanu, P. H. Nguyen, W. L. Kling, and M. Gibescu. "Unsupervised energy prediction in a Smart Grid context using reinforcement cross-building transfer learning". In: *Energy and Buildings* 116 (Mar. 2016), pp. 646–655. ISSN: 0378-7788. DOI: 10.1016/J.ENBUILD.2016.01.030.

[48] A. Y. Ng, A. Y. Ng, D. Harada, and S. Russell. "Policy invariance under reward transformations: Theory and application to reward shaping". In: *Proceedings of the 16th International Conference on Machine Learning* (1999), pp. 278–287.

[49] F. Pallonetto, M. De Rosa, F. Milano, and D. P. Finn. "Demand response algorithms for smart-grid ready residential buildings using machine learning models". In: *Applied Energy* 239 (Apr. 2019), pp. 1265–1282. ISSN: 0306-2619. DOI: 10.1016/J.APENERGY.2019.02.020.

[50] S. J. Pan and Q. Yang. "A survey on transfer learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. ISSN: 10414347. DOI: 10.1109/TKDE.2009.191.

[51] K. Paridari, D. Azuatalam, A. C. Chapman, G. Verbič, L. Nordström, D. Azuatalam, A. C. Chapman, and G. Verbič. "A plug-and-play home energy management algorithm using optimization and machine learning techniques". In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018. DOI: 10.1109/SmartGridComm.2018.8587418.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury Google, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. K. Xamla, E. Yang, Z. Devito, M. Raison Nabla, A. Tejani, S. Chilamkurthy, Q. Ai, B. Steiner, L. F. Facebook, J. B. Facebook, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32 (NIPS 2019)*. Curran Associates, Inc., 2019, pp. 8026–8037.

[53] C. Patyn and G. Deconinck. "Electric Water Heater Control Through Informed Fitted Q-Iteration". In: *Proceedings of 2019 IEEE PES Innovative Smart Grid Technologies Europe, ISGT-Europe 2019* (Sept. 2019). DOI: 10.1109/ISGTEUROPE.2019.8905737.

[54] C. Patyn, T. Peirelinck, and G. Deconinck. "Intelligent Electric Water Heater Control with Varying State Information". In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–7. ISBN: 9781538679548. DOI: 10.1109/SmartGridComm.2018.8587453.

[55] C. Patyn, F. Ruelens, and G. Deconinck. "Comparing neural architectures for demand response through model-free reinforcement learning for heat pump control". In: *2018 IEEE International Energy Conference, ENERGYCON 2018*. Institute of Electrical and Electronics Engineers Inc., June 2018, pp. 1–6. ISBN: 9781538636695. DOI: 10.1109/ENERGYCON.2018.8398836.

[56] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Domain Randomization for Demand Response of an Electric Water Heater". In: *IEEE Transactions on Smart Grid* 12.2 (May 2020), pp. 1370–1379. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.3024656.

[57] T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Transfer learning for Demand Response of a Multi-Agent Battery and Electric Water Heater System". In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2021. DOI: 10.1109/ISGTEurope52324.2021.9640081.

[58] T. Peirelinck, H. Kazmi, B. V. Mbuwir, C. Hermans, F. Spiessens, J. Suykens, and G. Deconinck. "Transfer learning in demand response: A review of algorithms for data-efficient modelling and control". In: *Energy and AI* 7 (2022), p. 100126. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2021.100126.

[59] T. Peirelinck, F. Ruelens, and G. Deconinck. "Using reinforcement learning for optimizing heat pump control in a building model in Modelica". In: *2018 IEEE International Energy Conference (ENERGYCON)*. IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-3669-5. DOI: 10.1109/ENERGYCON.2018.8398832.

[60] T. Peirelinck, F. Spiessens, C. Hermans, and G. Deconinck. "Double Q-learning for Demand Response of an Electric Water Heater". In: *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2019. DOI: 10.1109/ISGTEurope.2019.8905776.

[61] T. Peirelinck, F. Spiessens, C. Hermans, and G. Deconinck. "Double Q-learning for Demand Response of an Electric Water Heater". In: *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2019, p. 5.

[62] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *2018 IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 3803–3810. ISBN: 9781538630815. DOI: 10.1109/ICRA.2018.8460528. arXiv: 1710.06537.

[63] S. Pfenninger and I. Staffell. "Long-term patterns of European PV output using 30 years of validated hourly reanalysis and satellite data". In: *Energy* 114 (Nov. 2016), pp. 1251–1265. ISSN: 03605442. DOI: 10.1016/j.energy.2016.08.060.

[64] D. Picard, M. Sourbron, F. Jorissen, J. Cigler, Z. Vá?a, L. Ferkl, and L. Helsen. "Comparison of Model Predictive Control performance using grey-box and white box controller models". In: *International High Performance Buildings Conference* (Jan. 2016). URL: https://docs.lib.purdue.edu/ihpbc/203.

[65] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model predictive control : theory, computation, and design*. 2nd ed. Santa Barbara: Nob Hill Publishing, LLC, 2019, p. 623. ISBN: 0975937731.

[66] M. Reymond, C. Patyn, R. Radulescu, A. Nowé, and G. Deconinck. "Reinforcement Learning for Demand Response of Domestic Household Appliances". In: *Proceedings of the Adaptive Learning Agents Workshop 2018*. 2018, pp. 18–25.

[67] S. Riaz, H. Marzooghi, G. Verbic, A. C. Chapman, and D. J. Hill. "Generic Demand Model Considering the Impact of Prosumers for Future Grid Scenario Analysis". In: *IEEE Transactions on Smart Grid* 10.1 (Jan. 2019), pp. 819–829. ISSN: 19493053. DOI: 10.1109/TSG.2017.2752712. arXiv: 1605.05833.

[68] M. Riedmiller. "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method". In: *European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[69] F. Ruelens, B. J. Claessens, S. Quaiyum, B. De Schutter, R. Babuška, and R. Belmans. "Reinforcement Learning Applied to an Electric Water Heater: From Theory to Practice". In: *IEEE Transactions on Smart Grid* 9.4 (2018), pp. 3792–3800. ISSN: 19493053. DOI: 10.1109/TSG.2016.2640184. arXiv: 1512.00408.

[70] F. Ruelens, B. J. Claessens, S. Vandael, B. De Schutter, R. Babuška, and R. Belmans. "Residential Demand Response of Thermostatically Controlled Loads Using Batch Reinforcement Learning". In: *IEEE Transactions on Smart Grid* 8.5 (Sept. 2017), pp. 2149–2159. ISSN: 1949-3053. DOI: 10.1109/TSG.2016.2517211.

[71] F. Ruelens, B. J. Claessens, P. Vrancx, F. Spiessens, and G. Deconinck. "Direct Load Control of Thermostatically Controlled Loads Based on Sparse Observations Using Deep Reinforcement Learning". In: *arXiv preprint arXiv:1707.08553* (2017).

[72] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. "High-dimensional continuous control using generalized advantage estimation". In: *4th International Conference on Learning Representations, ICLR 2016*. June 2016. arXiv: 1506.02438.

[73] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". In: (July 2017). arXiv: 1707.06347.

[74] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (Oct. 2017), p. 354. ISSN: 1476-4687. DOI: 10.1038/nature24270.

[75] R. Sinha, B. B. Jensen, J. R. Pillai, C. Bojesen, and B. Moller-Jensen. "Modelling of hot water storage tank for electric grid integration and demand response control". In: *2017 52nd International Universities Power Engineering Conference, UPEC 2017*. IEEE, 2017, pp. 1–6. ISBN: 9781538623442. DOI: 10.1109/UPEC.2017.8231964.

[76] R. Sinha, B. B. Jensen, J. R. Pillai, C. Bojesen, and B. Moller-Jensen. "Modelling of hot water storage tank for electric grid integration and demand response control". In: *2017 52nd International Universities Power Engineering Conference, UPEC 2017*. Vol. 2017-Janua. Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1–6. ISBN: 9781538623442. DOI: 10.1109/UPEC.2017.8231964.

[77] A. Soares, O. De Somer, D. Ectors, F. Aben, J. Goyvaerts, M. Broekmans, F. Spiessens, D. Van Goch, and K. Vanthournout. "Distributed Optimization Algorithm for Residential Flexibility Activation-Results from a Field Test". In: *IEEE Transactions on Power Systems* 34.5 (2019), pp. 4119–4127. ISSN: 15580679. DOI: 10.1109/TPWRS.2018.2809440.

[78] A. Soares, D. Geysen, F. Spiessens, D. Ectors, O. De Somer, and K. Vanthournout. "Using reinforcement learning for maximizing residential self-consumption – Results from a field test". In: *Energy and Buildings* 207 (2020). ISSN: 03787788. DOI: 10.1016/j.enbuild.2019.109608.

[79] F. Steinke, P. Wolfrum, and C. Hoffmann. "Grid vs. storage in a 100% renewable Europe". In: *Renewable Energy* 50 (Feb. 2013), pp. 826–832. ISSN: 0960-1481. DOI: 10.1016/J.RENENE.2012.07.044.

[80] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith. "Model Predictive Climate Control of a Swiss Office Building: Implementation, Results, and Cost-Benefit Analysis". In: *IEEE Transactions on Control Systems Technology* 24.1 (Jan. 2016), pp. 1–12. ISSN: 10636536. DOI: 10.1109/TCST.2015.2415411.

[81] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems (NIPS 1999)*. Vol. 12. MIT Press, 2000, pp. 1057–1063.

[82] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. 2017.

[83] Y. Tao, J. Qiu, and S. Lai. "A Hybrid Cloud and Edge Control Strategy for Demand Responses Using Deep Reinforcement Learning and Transfer Learning". In: *IEEE Transactions on Cloud Computing* Early Access (2021). ISSN: 21687161. DOI: 10.1109/TCC.2021.3117580.

[84] M. E. Taylor, N. K. Jong, and P. Stone. "Transferring Instances for Model-Based Reinforcement Learning". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5212 LNAI.PART 2 (2008), pp. 488–505. ISSN: 03029743. DOI: 10.1007/978-3-540-87481-2_32.

[85] M. E. Taylor and P. Stone. "Transfer Learning for Reinforcement Learning Domains: A Survey". In: *Journal of Machine Learning Research* 10 (2009), pp. 1633–1685.

[86] A. Tirinzoni, A. Sessa, M. Pirotta, and M. Restelli. "Importance Weighted Transfer of Samples in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by J. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Dec. 2018, pp. 4936–4945. URL: https://proceedings.mlr.press/v80/tirinzoni18a.html.

[87] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE, Dec. 2017, pp. 23–30. ISBN: 9781538626825. DOI: 10.1109/IROS.2017.8202133. arXiv: 1703.06907.

[88] H. Van Hasselt. "Double Q-learning". In: *Advances in Neural Information Processing Systems 23*. Curran Associates, Inc., 2010, pp. 2613–2621.

[89] H. Van Hasselt, A. Guez, and D. Silver. "Deep reinforcement learning with double Q-Learning". In: *30th AAAI Conference on Artificial Intelligence, AAAI 2016*. AAAI, 2016, pp. 2094–2100. ISBN: 9781577357605. arXiv: 1509.06461.

[90] K. Vanthournout, R. D'hulst, D. Geysen, and G. Jacobs. "A Smart Domestic Hot Water Buffer". In: *IEEE Transactions on Smart Grid* 3.4 (2012), pp. 2121–2127. ISSN: 1949-3053. DOI: 10.1109/TSG.2012.2205591.

[91] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer New York, 2000. DOI: 10.1007/978-1-4757-3264-1_1.

[92] V. N. Vapnik. *Statistical learning theory*. Wiley, 1998, p. 736. ISBN: 978-0-471-03003-4.

[93] J. R. Vázquez-Canteli and Z. Nagy. "Reinforcement learning for demand response: A review of algorithms and modeling techniques". In: *Applied Energy* 235 (Feb. 2019), pp. 1072–1089. ISSN: 0306-2619. DOI: 10.1016/J.APENERGY.2018.11.002.

[94] Vlaamse Regulator Energie en Gas (VREG). *Toekomst nettarieven – capaciteitstarief | VREG*. URL: https://www.vreg.be/nl/toekomst-nettarieven-capaciteitstarief (visited on 09/16/2020).

[95] Vlaamse Regulator Energie en Gas (VREG). *V-Test*. 2021. URL: https://vtest.vreg.be/ (visited on 01/19/2022).

[96] L. Vonrueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, M. Walczak, J. Pfrommer, A. Pick, R. Ramamurthy, J. Garcke, C. Bauckhage, and J. Schuecker. "Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems". In: *IEEE Transactions on Knowledge and Data Engineering* (2021). ISSN: 15582191. DOI: 10.1109/TKDE.2021.3079836. arXiv: 1903.12394.

[97] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang. "A survey of transfer learning". In: *Journal of Big Data* 3.1 (Dec. 2016), pp. 1–40. ISSN: 21961115. DOI: 10.1186/s40537-016-0043-6. URL: https://link-springer-com.kuleuven.ezproxy.kuleuven.be/articles/10.1186/s40537-016-0043-6%20https://link-springer-com.kuleuven.ezproxy.kuleuven.be/article/10.1186/s40537-016-0043-6.

[98] P. Zeng, C. Sheng, and M. Jin. "A learning framework based on weighted knowledge transfer for holiday load forecasting". In: *Journal of Modern Power Systems and Clean Energy* 7.2 (Mar. 2019), pp. 329–339. ISSN: 21965420. DOI: 10.1007/S40565-018-0435-Z.

[99] W. Zhang, L. Deng, and D. Wu. "Overcoming Negative Transfer: A Survey". In: (2020). arXiv: 2009.00909.

[100] Y. Zhang and G. Luo. "Short term power load prediction with knowledge transfer". In: *Information Systems* 53 (Oct. 2015), pp. 161–169. ISSN: 0306-4379. DOI: 10.1016/J.IS.2015.01.005.

[101] W. Zhao, J. P. Queralta, and T. Westerlund. “Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey”. In: *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020* (Dec. 2020), pp. 737–744. DOI: 10.1109/SSCI47803.2020.9308468. arXiv: 2009.13303.

Curriculum vitae

Thijs Peirelinck
born on 8th November 1993, Diest, Belgium.

2017-2022	Doctoral Program in Engineering Science KU Leuven, Leuven, Belgium From 2018, supported by a scholarship from "Vlaams Instituut Technologisch Onderzoek" (VITO)
2016 - 2017	MSc Artificial Intelligence KU Leuven, Leuven, Belgium
2015 - 2016	EIT-KIC MSc Energy for Smart Cities Grenoble INP, Grenoble, France
2014 - 2015	EIT-KIC MSc Energy for Smart Cities KTH Stockholm, Stockholm, Sweden
2011 - 2014	BSc Engineering Science KU Leuven, Leuven, Belgium

List of publications

Journals

- T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Domain Randomization for Demand Response of an Electric Water Heater". In: *IEEE Transactions on Smart Grid* 12.2 (May 2020), pp. 1370–1379. ISSN: 1949-3053. DOI: 10.1109/tsg.2020.3024656
- T. Peirelinck, H. Kazmi, B. V. Mbuwir, C. Hermans, F. Spiessens, J. Suykens, and G. Deconinck. "Transfer learning in demand response: A review of algorithms for data-efficient modelling and control". In: *Energy and AI* 7 (2022), p. 100126. ISSN: 2666-5468. DOI: 10.1016/j.egyai.2021.100126
- *(To be submitted)* T. Peirelinck, C. Hermans, F. Spiessens, G. Deconinck,"Combined Peak Reduction and Self-Consumption Using Proximal Policy Optimization", 2022

Conference Proceedings

- T. Peirelinck, F. Ruelens, and G. Deconinck. "Using reinforcement learning for optimizing heat pump control in a building model in Modelica". In: *2018 IEEE International Energy Conference (ENERGYCON)*. IEEE, 2018, pp. 1–6. ISBN: 978-1-5386-3669-5. DOI: 10.1109/ENERGYCON.2018.8398832
- C. Patyn, T. Peirelinck, and G. Deconinck. "Intelligent Electric Water Heater Control with Varying State Information". In: *2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 2018, pp. 1–7. ISBN: 9781538679548. DOI: 10.1109/SmartGridComm.2018.8587453

- T. Peirelinck, F. Spiessens, C. Hermans, and G. Deconinck. "Double Q-learning for Demand Response of an Electric Water Heater". In: *2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2019. DOI: 10.1109/ISGTEurope.2019.8905776
- T. Peirelinck, C. Hermans, F. Spiessens, and G. Deconinck. "Transfer learning for Demand Response of a Multi-Agent Battery and Electric Water Heater System". In: *2021 IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*. IEEE, 2021. DOI: 10.1109/ISGTEurope52324.2021.9640081

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ELECTA
Kasteelpark Arenberg 10, box 2445
3001 Leuven
thijs.peirelinck@gmx.com
https://www.esat.kuleuven.be/electa